Diogo da Mota Pinto

Licenciado em Ciências da Engenharia Electrotécnica
e de Computadores

Simulação de Sistemas Distribuídos de Manufactura

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

Orientador: Doutor João Rosas,

Professor Assistente, Departamento de Engenharia Electrotécnica,

Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri:

Presidente: Doutor Pedro Miguel Ribeiro Pereira, FCT – UNL

Arguente: Doutor Yves Rybarczyk, FCT - UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Simulação de Sistemas Distribuídos de Manufactura

Aos meus pais e irmão

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao meu orientador, Professor João Rosas, por ter acreditado em mim e me ter aceite como seu orientando. Agradeço-lhe também por toda a disponibilidade e apoio demonstrados durante a realização deste trabalho. À FCT – UNL agradeço o acolhimento e condições oferecidas para a minha formação.

A realização deste trabalho não teria sido possível sem o amor incondicional dos meus pais, que nunca deixaram de acreditar em mim. Por tudo o que me proporcionaram e pelo apoio demonstrado muito obrigado.

Quero também agradecer ao meu irmão, Miguel Pinto, pela intimidade criada ao longo destes anos que me deram força nas alturas mais difíceis.

Agradeço também à minha namorada, Inês Pinheiro, pelo amor e companhia durante estes anos.

Quero também agradecer a toda a minha família que ao longo destes anos estiveram sempre disponíveis: Ana Afonso, Cristiana Pinto, Margarida Pinto, Tânia Afonso, Paula Pinto, Afonso Costa, André Pinto, António Mota, Cláudio Pinto, Eliseu Pinto, Fábio Pinto, Henrique Pinto e Tiago Pinto.

Por fim quero também agradecer a todos os meus amigos pelos bons momentos passados, em especial: Ana Cruz, Joana Urbano, Raquel Melo, Rita São Pedro, António Cavaco, Bruno Caixinha, David Aleixo, Hugo Pereira, João Eusébio, João São Pedro, Manuel Ferro, Marco Alves, Nuno Barreira, Nuno Pinheiro, Pedro Bueno, Pedro Oliveira e Tiago Pereira.

Muito Obrigado!

Resumo

Os sistemas distribuídos de manufactura (SDM) possuem como principais características a capacidade de executarem processos de manufactura de forma distribuída em ambientes heterogéneos, em que os diversos estágios e parceiros envolvidos funcionam de forma autónoma. Dotados de uma boa capacidade de adaptação a mercados em constante mudança, são sistemas complexos e dinâmicos, sendo difícil prever o seu desempenho sem abordagens adequadas, nomeadamente, recorrendo a métodos de simulação. No entanto os pacotes de simulação que existem actualmente, apesar de boa qualidade e bastante flexíveis, ainda se caracterizam por uma limitada capacidade para a simulação de SDM. Por outro lado, o desenvolvimento duma ferramenta de manufactura que combine a simulação ao nível local de processos de manufactura (por exemplo: com robôs e maquinas industriais) e a correspondente ao nível global e distribuído, revela-se significativamente complexo e oneroso. Ao invés de se desenvolver tal sistema, com este trabalho, pretende-se especificar um modelo e respectiva ferramenta que permita a simulação de SDM, recorrendo a produtos comerciais já existentes, mediante a definição duma estratégia de interoperabilidade entre as ferramentas. A simulação em simultâneo, de forma agregada e em tempo real dos diversos níveis estratégicos e locais de um SDM, permite assim uma melhor estimativa do seu desempenho futuro.

Palavras-chave: Sistemas Distribuídos de Manufactura, Simulação, Modelação, Arquitectura Orientada a Serviços, Integração e Interoperabilidade.

Abstract

Distributed manufacturing systems (DMS) have the ability to execute distributed manufacturing processes in heterogeneous environments, wherein the different stages and involved partners operate autonomously. Provided with the ability to adapt to constantly changing markets, they're complex and dynamic systems, being hard to predict its performance without the right approach, such as simulations methods. However, the simulated packages currently in existence are considered to have a limited capacity for DMS simulation despite of its fine quality and flexibility. On the other hand the development of a manufacturing tool that combines the simulation of locally individual processes (for example: with robots or industrial machines) with the simulation of globally distributed processes can be quite complex and costly. In this thesis instead of developing this kind of system, it is intended to specify a model and respective tool that would allow the simulation of DMS, using existing commercial tool by definition of an interoperability strategy between this tools. Simultaneous simulation, grouped and in real time of the different local and strategic levels of a DMS, allow a better evaluation about its future performance.

Keywords: Distributed manufacturing system, Simulation, Modelling, Service-Oriented Architecture, Integration and Interoperability.

Índice de Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

ABB	Asea Brown Boveri
API	Application Programming Interface
BD	Base de Dados
CAD	computer-aided design
CAM	computer-aided manufacturing
CAPI	Controller Application Programming Interface
EMD	Empresas de Manufactura Distribuída
HLA	High Level Architecture
ICT	Information and Communication Technology
IP	Internet Protocol
LP	Logical Processes
PC SDK	PC Software Development Kit
PROSA	Product-Resource-Order-Staff-Achitecture
RTI	Runtime Infrastructure
SDM	Sistemas Distribuídos de Manufactura
SIMNET	SIMulator NETworking
SOA	Services Oriented Architecture

SQL Structured Query Language

TIC Tecnologias da Informação e Comunicação

URL Uniform Resource Locator

VBA Visual Basic for Applications

WCF Windows Communication Foundation

1 Introdução

Num mercado caracterizado por um constante crescimento, globalizado, competitivo e disruptivo, sente-se a necessidade de adoptar novos paradigmas de manufactura, e assim novas formas de gestão, que permitam às empresas serem bem sucedidas, e mesmo sobreviver, em tais mercados. Oscilando entre a necessidade de produção em massa, para diminuir custos, e a necessidade de personalização de produtos, para melhor atender clientes com preferências mais sofisticadas, os sistemas de manufactura têm de acompanhar essa dinâmica de mudança e transformação culminando, por exemplo, nos sistemas distribuídos de manufactura. Como consequência, esta complexidade crescente aumenta a incerteza e o risco dos processos de gestão e decisão estratégicos. Neste contexto, a simulação surge como uma estratégia para se conseguir mitigar tais riscos.

Efectivamente, as empresas recorrem cada vez mais a processos de decisão que incorporem simulação com vista a dar melhor resposta às necessidades dum mercado sofisticado. As ferramentas de simulação permitem percepcionar o desempenho futuro de sistemas e produtos, dando a oportunidade às empresas de melhor adoptarem os seus sistemas e processos, e também, a correcção de erros estratégicos ou presentes nos processos de manufactura, que tornar-se-iam bastante onerosos se detectados em fases posteriores.

Os métodos e ferramentas de simulação têm conseguido satisfazer as necessidades das empresas. Nota-se, no entanto, que cada ferramenta de simulação foi desenvolvida para melhor resolver determinados tipos de problemas. Muitas delas focam-se no nível “microscópico” dos processos de manufactura, outras na simulação dos processos de negócio a níveis mais estratégicos. Existe uma lacuna, que importa satisfazer, para criar-se ferramentas de simulação, que de uma forma mais abrangente, permitam considerar os níveis micro e macro de um sistema de manufactura. Tais ferramentas são necessárias no suporte à gestão de sistemas distribuídos de manufactura conforma ilustrado na Figura 1.1.

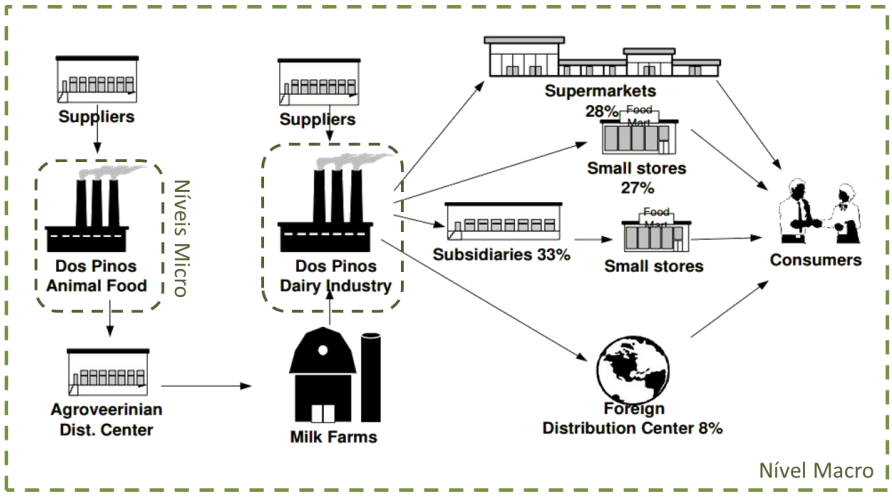


Figura 1.1 – Exemplo de uma Simulação de um SDM

Estes sistemas seriam assim capazes de simular as diversas fases na elaboração de um produto, começando pela entrada de encomendas dos clientes, passando pela manufactura até à fase de entrega. Estes sistemas poderiam considerar os níveis locais (de cada célula ou parceiro) e os níveis globais, macro, numa rede de SDM. Tendo em consideração que tais ferramentas de simulação que permitem simular os diferentes processos de um SDM são difíceis de encontrar, uma das opções passaria pelo seu respectivo desenvolvimento, no entanto, esta opção seria bastante dispendiosa. Outra opção passa pela integração de ferramentas comerciais já existentes, cada uma delas focada nos respectivos níveis locais ou macros dum SDM, como podemos verificar na Figura 1.2. Esta opção permite assim assumir uma perspectiva abrangente a um sistema de manufactura que é menos onerosa, pois é apenas necessário desenvolver um mecanismo adequado de integração de ferramentas de simulação que já satisfazem os requisitos de interoperabilidade.

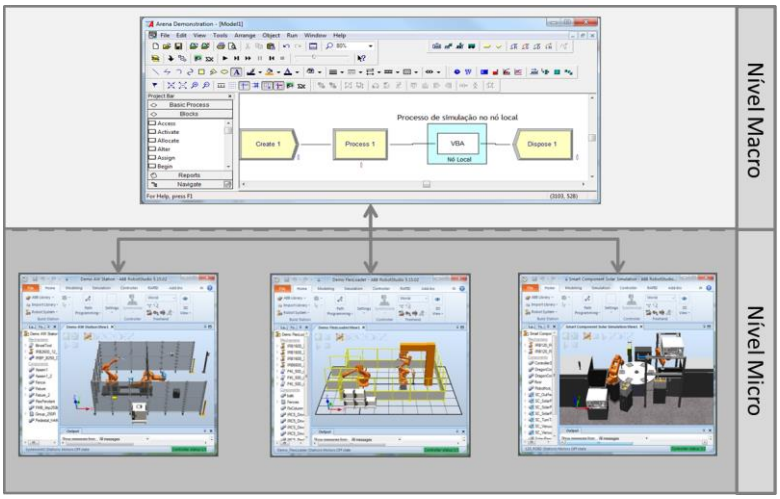


Figura 1.2 – Infra-estrutura de integração para ferramentas de simulação já existentes

Neste trabalho, propõem-se a modelação e desenvolvimento dum mecanismo de integração para ferramentas de simulação já existentes. Este mecanismo vai servir para simular cenários ou problemas de sistemas de produção distribuída numa perspectiva abrangente. Este mecanismo baseia-se na utilização de paradigmas de desenvolvimento de arquitecturas baseadas em serviços, para a referida integração de pacotes de *software* de simulação já disponíveis no mercado.

1.1 Motivação

Nos sistemas de manufactura actuais a modelação e simulação assumem um papel importante na forma como as empresas implementam os seus processos. A vantagem de estabelecer incertezas e de análise prévia das soluções que se pretendem alcançar tornam a simulação uma ferramenta preponderante no sucesso das empresas e dos seus produtos. A adesão a pacotes de *software* de simulação para melhor perceber os processos de negócio, tem vindo a ganhar importância na forma como as empresas aplicam as suas estratégias.

No entanto, a ausência de ferramentas que permitam modelar e simular problemas no âmbito geral de manufactura nos diferentes níveis é ainda uma realidade. Daí, a motivação em criar uma ferramenta que permitisse aliar à simulação dos seus processos de negócio processos de produção, tornando-se assim numa vantagem competitiva para essa ferramenta. Porém, nas tecnologias já existentes, encontram-se algumas lacunas quando se pretende simular níveis diferentes dos processos dos sistemas de manufactura.

Devido à complexidade na criação de uma solução de raiz de uma ferramenta que conciliasse os processos que ocorrem nos diferentes níveis, a solução encontrada consiste em utilizar ferramentas de simulação já existentes, tornando-as capazes de, a partir das actividades individuais, cooperarem entre si num sistema dinâmico e distribuído.

Esta *visão* coloca desafios importantes ao nível de interoperabilidade e integração destas ferramentas, na forma como a troca de informação entre elas possa ser garantida.

O recente desenvolvimento das TIC[1], nomeadamente as arquitecturas orientadas a serviços, permitem lidar com estes desafios na elaboração da solução proposta, e assim criar uma interface baseada na comunicação.

1.2 Metodologia

[1] Tecnologias de Informação e Comunicação

A metodologia de trabalho utilizada nesta dissertação foi inspirada pelo Método Científico descrito em (Schafersman, 1997). O processo de pesquisa para este trabalho, conforme ilustra a Figura 1.3, é composta pelos seguintes passos:

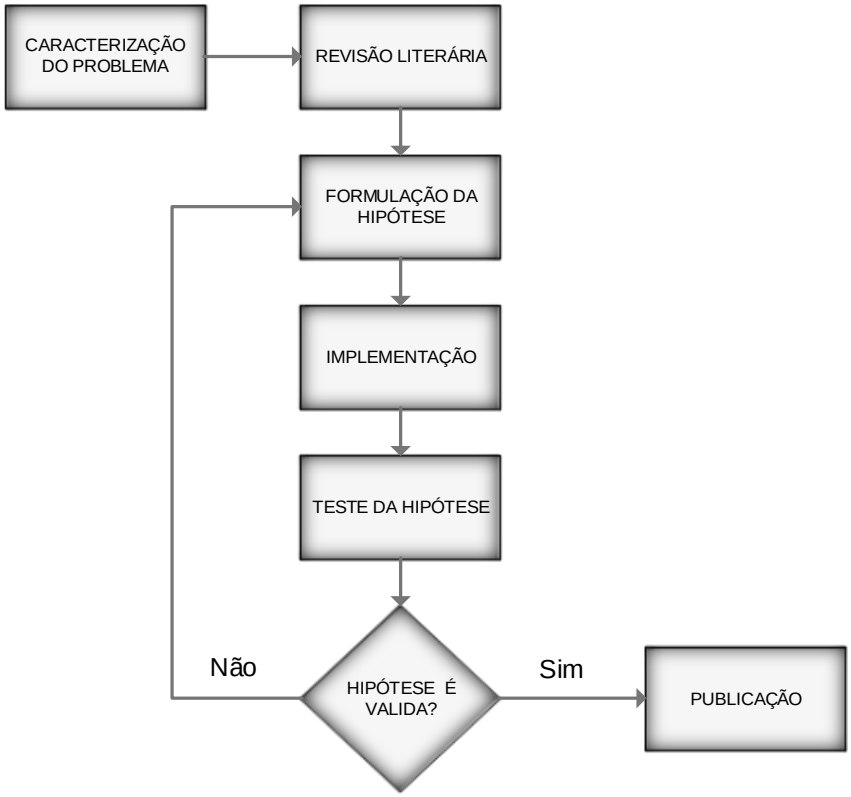


Figura 1.3 – Método de pesquisa adoptado

Caracterização do problema: O primeiro passo é identificar um problema e as respectivas características. O problema identificado foi simular sistemas de manufactura distribuído, numa perspectiva abrangente, considerando os diversos níveis de simulação correspondentes às distintas componentes dos sistemas.

Revisão Literária: Neste ponto são consultados trabalhos efectuados anteriormente relativos ao problema proposto. Com esta revisão literária, pretende-se também identificar questões em aberto que possam ser relevantes para o problema abordado neste trabalho.

Formulação de hipótese: A revisão literária permite um enquadramento mais claro do problema proposto. Daí, resulta uma melhor identificação da questão de pesquisa para a qual será formulada uma hipótese de solução, que será descrita mais à frente.

Implementação: Modelar e desenvolver ferramentas que vão permitir testar e validar a hipótese formulada.

Teste da hipótese: Aplicação de ferramentas de cenários de simulação de forma a testar a sua aplicabilidade e obter resultados que vão ser utilizado para aferir a validade da hipótese.

Análise de resultados: Finalizados os testes, os resultados serão analisados e a proposta é validada. Os resultados são satisfatórios se estes corresponderem com as previsões estabelecidas anteriormente, permitindo validar a hipótese. Caso os resultados obtidos não sejam os esperados deverá ser reformulada uma nova hipótese.

Publicações: A publicação de resultados é feita de acordo com o resultado da experimentação do trabalho de pesquisa.

1.3 Formulação da Hipótese

O primeiro passo consiste em estabelecer uma questão de pesquisa para o problema em estudo.

"Como simular sistemas de manufactura distribuída numa perspectiva abrangente que considere os níveis Macro e Micro destes Sistemas?"

A resposta a esta questão consiste na hipótese de tornar as ferramentas de simulação já existentes colaborativas. Esta abordagem permite assim uma perspectiva mais abrangente dos processos que ocorrem nos diferentes níveis de simulação.

A estratégia de implementação passa pelo desenvolvimento de um mecanismo de integração dessas ferramentas de simulação tornando-as interoperáveis. Tal mecanismo baseia-se no paradigma das arquitecturas orientadas a serviços, conforme será descrito nos capítulos seguintes deste trabalho.

1.4 Esquema da Dissertação

No primeiro capítulo faz-se uma caracterização do problema em questão, nomeadamente, simulação de sistemas distribuídos de manufactura. Depois desta caracterização, descreve-se a abordagem seguida para satisfazer a questão de pesquisa[2] formulada. O capítulo dois é dedicado ao levantamento do estado da arte, que engloba o estado corrente das áreas relevantes em estudo, nomeadamente os sistemas distribuídos de manufactura, a simulação destes sistemas, a robótica e as arquitecturas orientadas a serviços.

[2] *Research questions*

No capítulo três, descreve-se o processo seguido na implementação da abordagem proposta, começando pela fase de modelação até á implementação.

No capítulo quatro são apresentados os testes realizados no sistema assim como os resultados obtidos nas diferentes fases de implementação. No capítulo final são apresentadas as conclusões sobre a hipótese apresentada e as propostas para trabalho futuro.

2 Estado da Arte

Neste capítulo foi realizada uma investigação para caracterizar o estado da arte acerca dos sistemas distribuídos de manufactura e a simulação destes sistemas.

2.1 Sistemas de manufactura

A noção de sistema distribuído de manufactura torna-se mais clara se primeiro se introduzir a noção de sistema de manufactura (tradicional), sendo este o propósito desta secção.

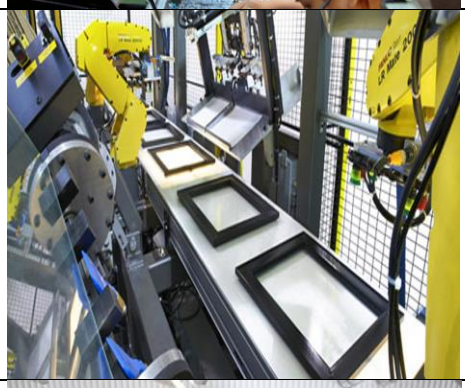
Um sistema de manufactura pode ser visto como uma colecção de equipamentos e recursos humanos, com o objectivo de processar ou proceder a operações de montagens de material, peças ou conjuntos de peças. É num sistema de manufactura que o trabalho efetuado acrescenta valor às peças e produtos. Um sistema de manufactura é tipicamente constituído por:

- Máquinas de manufactura e respectivas ferramentas;
- Mecanismo que permitam manusear material e dispositivos de posicionamento;
- Sistema informático para coordenar e controlar estes mecanismos;
- Trabalhadores para operarem e supervisionarem o sistema;

A Tabela 2.1 ilustra algumas classes de sistemas de manufactura. Esta tabela foi realizada tendo como base a informação contida em (Groover, 2007).

Tabela 2.1: Classes dos sistemas de manufactura.

Tipo de Sistema	Ilustração
Estações de células únicas: Um trabalhador a laborar numa única máquina que opera em ciclos semiautomáticos. (URASTUN, 2003)	

Grupos de Produção: Um trabalhador supervisiona um conjunto de máquinas semiautomáticas. (Research, 2011)	
Linhas de Manufactura Manuais: Linhas de manufactura manuais são grupos de estações de trabalho onde as operações são realizadas por humanos numa linha automática. (PCSTATS, 2014)	
Linhas de Transferência Automática: As linhas de produção consistem em estações de trabalho automatizadas que melhoram o desempenho das operações. (systems, 2014)	
Sistemas de Montagem Automáticos Constituídos por dispositivos automáticos programados para executarem sequências fixas e repetitivas de um produto específico. (DPN, 2014)	

Células de Manufactura Executam operações para um grupo de produtos idênticos que têm finalidades diferentes. (Carnegie Mellon University, 2010)	
Sistemas de Manufactura flexível Sistemas automatizados capazes de produzir um grupo de produtos diferentes. (KUKA, 2014)	

No passado recente, com a necessidade de aumentar a produtividade, a indústria começou a adoptar meios de produção com uma flexibilidade crescente, assentes na utilização do computador para controlar os processos fabris. Essa tendência conduziu também a uma crescente adopção de robôs industriais, capazes de desempenhar uma grande variedade de tarefas de forma mais flexível e a custos mais baixos.

Um robô industrial é um braço mecânico, controlado por um computador, que consiste num conjunto de elos com juntas, dotado duma "mão" para pegar em ferramentas utilizadas em processos de manufactura. Consegue operar dentro de um volume de trabalho estabelecido, utilizando um sistema de coordenadas cartesiana. A Figura 2.1 ilustra um robô articulado e um robô móvel.

Figura 2.1 – Exemplos de robô articulado e de robô móvel

A programação destes robôs é feita através de linguagem de programação especialmente criada para esse efeito, por exemplo, que consideram as características de um robô, nomeadamente, os movimentos, leitura de sensores, etc.

O robô participa nos processos de manufactura, sendo capazes de comunicar com outras máquinas e de se adaptarem a processos de manufactura cada vez mais flexíveis e complexos. Tanto os manipuladores como os robôs móveis são considerados elementos essenciais nos sistemas de manufactura actuais, incluindo os distribuídos. A Figura 2.2 ilustra alguns exemplos dos robôs que podem ser encontrados nos sistemas robóticos. Um dos elementos visíveis é o controlador, que para além de coordenar o funcionamento do robô, consegue comunicar em rede com os outros recursos de um sistema de produção distribuída.

Figura 2.2 – Linha de montagem com controlador local

2.2 Sistemas distribuídos de manufactura

Ao longo do último século, o mundo da manufactura vêm-se alterando principalmente no ramo da manufactura distribuída. Empresas de manufactura distribuída, EMD, também conhecidas como empresas virtuais de manufactura, caracterizam-se por operarem em ambientes geograficamente distribuídos que partilham informação entre si. As EMD são normalmente organizações temporárias onde várias empresas colaboram na produção de uma determinada linha de produto. Desta colaboração são partilhados conhecimentos e produtos e a experiencia de cada empresa aumenta levando a novas oportunidades de negócio. Normalmente, os compromissos estabelecidos são de curto prazo e são independentes das actividades individuais que cada uma executa, acabando por não ter qualquer tipo de influência nos processos internos de cada empresa (Saad, Perera, & Wickramarachchi, 2003).

Devido à sua natureza e ao ambiente que as rodeia, as EMD são compostas por sistemas complexos e heterogéneos onde a manufactura tradicional apresenta uma capacidade de adaptação baixa e incapaz de reagir a sistemas tão dinâmicos.

Este fenómeno levou ao desenvolvimento de arquitecturas de manufactura distribuída que consigam lidar com sistemas mais complexos e dinâmicos. Novos conceitos de manufactura foram introduzidos nos últimos anos para colmatar esses problemas. A Tabela 2.2 ilustra alguns desses paradigmas, tendo como base a informação contida em (Chituc & Restivo, 2009).

Tabela 2.2: Paradigmas de manufactura

Sistemas de Manufactura	Focos	Forças	Fraquezas	Abordagens
Produção em massa	Redução dos custos de produção; Utilização total das máquinas de produção	Utilização total dos recursos disponíveis	Inflexíveis	
Manufactura integrada com computadores	Utilização das ferramentas de suporte dos computadores	Utilização de computadores para suportar diferentes actividades	Incapazes de acompanhar necessidades actuais	
Flexível	Produção de diversos tipos de produto num só sistema	Grande diversidade de produtos	Custo elevado devido a funções de *software* desnecessárias	
Ágil	Capacidade de se ajustar rapidamente a novas necessidades externas	Integrável; Personalizável; Modulável	Custo elevado devido a funções de *software* desnecessárias	
Inteligente	Sistemas com capacidade de decisão			Agentes
Holónica		Preserva a estrutura hierárquica das entidades; entidades com estruturas semelhantes	Ponto central de controlo	Agentes, Redes de Petri
Biónica	Comportamentos autónomos e espontâneos, evolução	Adaptação; flexíveis;	Desenvolvimentos iniciais mais extensos	Agentes
wise	Habilidade para resolver problemas	Adaptação; Capacidade de evolução do sistema	Desenvolvimentos iniciais mais extensos	*Cloud computing*, ICT

Num contexto de manufactura distribuída, o objectivo é tornar os sistemas de manufactura autónomos e colaborativos, com a capacidade de poderem responder de forma mais eficiente a alterações dos mercados globais. A ideia é ter um sistema capaz de integrar *design* e desenvolvimento técnico, sistemas de produção e processos de negócio numa rede de trabalho distribuída que opera localmente e de forma independente. Este conceito é idealizado a partir de serviços orientados e ciclos de aprendizagem onde os requisitos estão associados a capacidades e possibilidades futuras. Toda a informação proveniente deste processo é armazenada e pode ser reutilizada em processos futuros (Nylund & Andersson, 2010).

Adicionalmente, produtos complexos são raramente produzidos por uma única companhia ou num único local. Normalmente as componentes mais básicas que compõem esses produtos são originárias de diferentes fabricantes, sendo apenas o processo de montagem executado num local comum. Como os armazéns possuem capacidades de armazenamento limitadas as entregas só acontecem quando a sua utilização está garantida, tornando a comunicação e organização entre empresas crucial. É portanto importante que os simuladores sejam capazes de incluir na simulação de modelos os processos dos seus fornecedores, pois desta forma consegue-se percepcionar os efeitos duma falha com origem nos fornecedores.

2.2.1 Exemplos de SDM

Multi-site manufacturing company

Uma empresa que funciona com um sistema *multi-site* possui toda a informação distribuída por *sites* individuais (Figura 2.3). Os *sites* podem ser configurados consoante a necessidade da empresa, permitindo a integração entre os sistemas que utiliza. A informação encontra-se deste modo organizada e distribuída para toda a empresa, facilitando a forma como são geridos os processos internos. Esta abordagem permite realizar alterações nestes processos ou realizar *updates* ao sistema de forma mais eficiente e a custos mais baixos, pois as alterações são realizadas num único ponto (SOFTWARE, 2014).

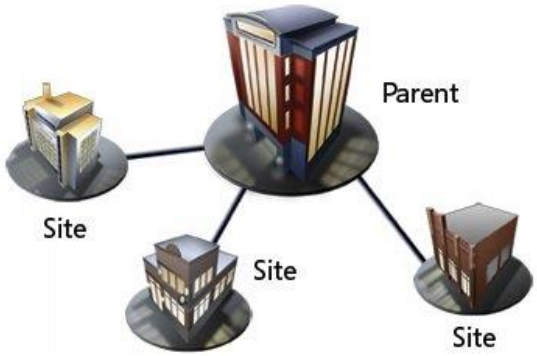


Figura 2.3 – Exemplo de "Multi-site manufacturing company"(LLC, 2011)

Cadeia de Fornecimento (*Supply-Chain*)

Uma cadeia de fornecimento é um tipo de sistema que interliga todas as fases do ambiente de manufactura, desde o fornecimento global até à venda dos produtos aos utilizadores. As diferentes etapas de passagem do produto são tidas em conta e trabalhadas no sistema. Normalmente este processo divide-se em duas fases: Fase de produção, onde o material é recebido, trabalho e criado um produto, e a fase de entrega, que são os processos por onde passa o produto até chegar ao consumidor final. A Figura 2.4 ilustra um exemplo comum de uma cadeia de fornecimento (Camarinha-Matos, Afsarmanesh, Galeano, & Molina, 2009).

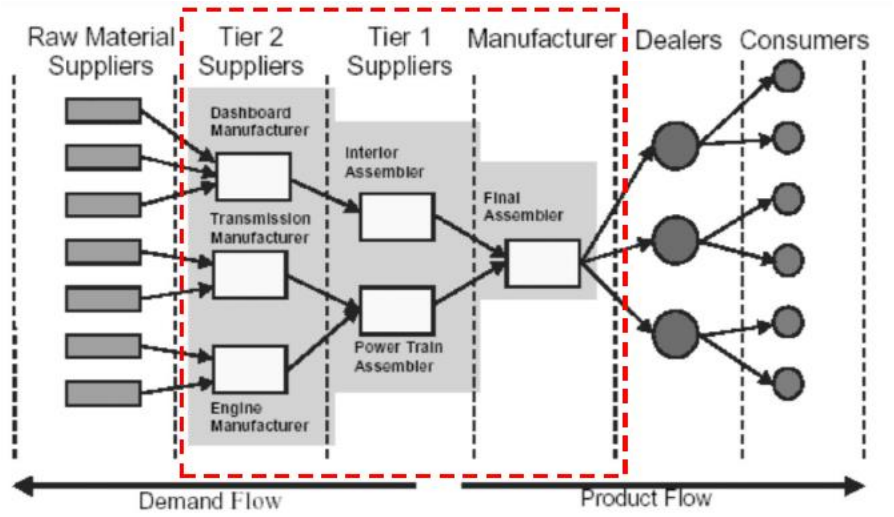


Figura 2.4 – Exemplo das fases de produção numa cadeia de fornecimento

2.3 Aspectos de Simulação

2.3.1 Conceito de simulação

De acordo com (Groover, 2007), a utilização de computadores para imitar ou simular as operações dos diferentes tipos de realidade ou processos nas mais diversas áreas é bastante comum. O conjunto de processos organizados e relacionados entre si é designado de sistema. Através de suposições sobre a forma de operar dos sistemas é possível prever o seu comportamento. Estas suposições aparecem normalmente sobre a forma de relações matemáticas ou lógicas às quais são chamadas de modelos. O modelo é usado para compreender e estudar os comportamentos e as formas de operar do sistema.

No caso das relações que compõem o modelo serem simples, é possível usar métodos matemáticos (algébricos, cálculos ou probabilidades) para obter informação exacta sobre o sistema. Estes métodos são chamados de soluções analíticas. Contudo, grande parte dos sistemas

existentes são demasiado complexos para permitirem uma análise analítica, devendo ser estudados a partir de simulações. Numa simulação, os computadores são utilizados para avaliar numericamente um modelo, sendo recolhida informação de maneira a fazer uma estimativa das características que se desejam atingir.

Consideremos uma empresa de manufactura que deseja renovar o seu ambiente fabril, mas que não tem a certeza se a melhoria nos ganhos na produção justificam o custo de investimento. Certamente que não é economicamente vantajoso partir directamente para a alteração dos seus processos de manufactura e caso o resultado não seja o esperado, ter que remover todas as alterações que tenham sido realizadas. Porém, se cada uma das situações for simulada previamente, poder-se-á ter acesso a informação que permita prever quais as situações alternativas que são mais vantajosas.

Existem diversas áreas onde a aplicação de métodos de decisão baseados em simulação se torna numa mais-valia. A lista que se segue apresenta algumas dessas situações mais comuns (Groover, 2007):

- Projectar e analisar um sistema de manufactura;
- Avaliar sistemas militares, tais como armas e processos logísticos;
- Determinar requisitos de um *hardware* e os protocolos de uma rede de comunicação;
- Projectar sistemas de transporte;
- Reengenharia de processos de negócio;
- Analisar linhas de produção.

A simulação surge como uma das técnicas de investigação operacional mais utilizadas nos dias que correm. Um dos indicadores desta tendência é a adesão de um grande número de investigadores nas conferências que vão decorrendo ao longo do ano, em especial a "*Winter Simulation Conference*" que por ano atrai entre 600 a 800 pessoas no seu evento (Conference, 2014).

Existem no entanto alguns factores que limitam o uso e a utilidade das simulações. Normalmente, os modelos são utilizados para estudar sistema de grandes dimensões e altamente complexos dificultando a construção de programas capazes de simular tais ambientes. No entanto, *ferramentas* recentes providenciam de forma automática os recursos necessários para a construção desses modelos. Outro obstáculo que com o passar dos anos tem sido ultrapassado, devido à evolução tecnológica nos computadores, é que sistemas com níveis tão elevados de complexidade dependem de capacidades de processamento também elevadas. Por último, a utilização de simuladores padece ainda de alguma credibilidade. A ideia que a simulação é meramente um exercício de programação para descobertas científicas, coloca de lado toda a informação precisa que pode ser utilizada na implementação de um sistema real.

Um aspecto intimamente associado à simulação é a respectiva modelação dos sistemas, artefactos ou fenómenos que se pretende simular.

2.3.2 Sistemas, Modelos e Simulação

Um sistema define-se como um conjunto de elementos (humanos, máquinas, etc) capazes de interagir entre eles de forma lógica. No entanto, na prática um sistema não pode ser definido de forma tão linear, uma vez que depende da situação onde se insere. Os sistemas são constituídos por entidades, subsistemas de funcionalidades independentes, e por estados, que caracterizam a situação em que se encontra o sistema. Os estados descrevem, normalmente, um momento relativo aos objectivos do sistema.

Um sistema pode ser caracterizado de dois tipos: discreto e contínuo. Num sistema discreto as variáveis são independentes do tempo e alteram-se de forma instantânea dependendo exclusivamente de acções. Por exemplo, o número de produtos presentes num armazém muda quando é feita uma reposição ou venda. No caso dos sistemas contínuos as variáveis alteram-se ao longo do tempo. Variáveis como velocidade ou aceleração estão dependentes do tempo e são alteradas de forma contínua. A Figura 2.5 ilustra as diferentes formas possíveis de estudo de um sistema.

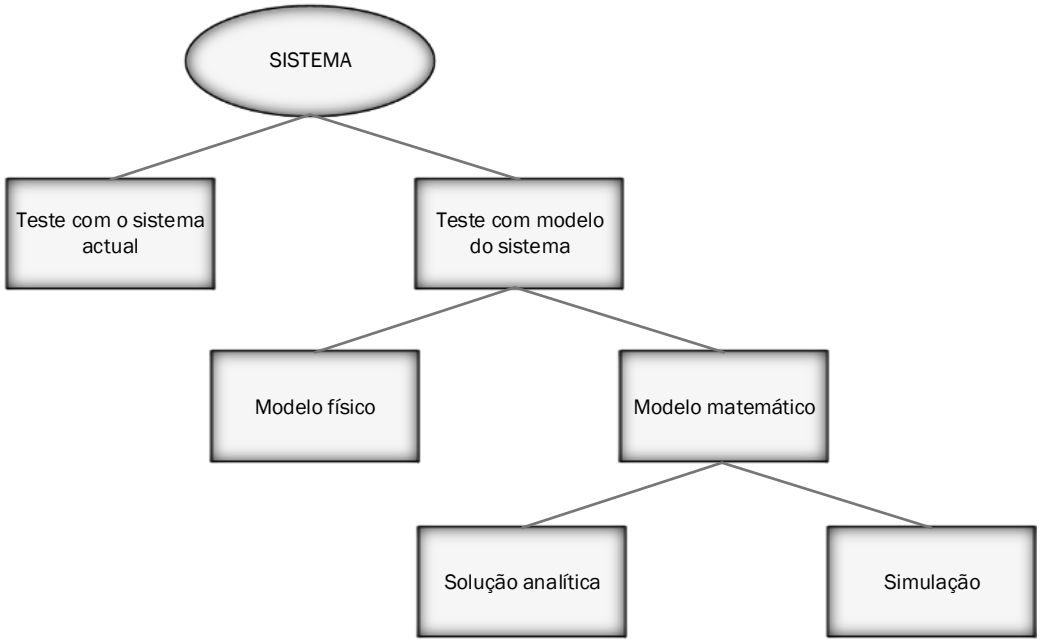


Figura 2.5 – Formas de estudar ou caracterizar um sistema

Teste com o sistema actual ou Teste com modelo do sistema: caso seja possível executar alterações no sistema sem que este sofra problemas ou o seu desempenho seja prejudicado, então a realização dos testes no sistema real é a melhor solução. No entanto, estes casos são

bastante raros, pois uma interrupção de um sistema é normalmente dispendiosa. Por esta razão, é normalmente necessário construir um modelo que espelhe as funcionalidades do sistema real.

Modelo físico ou Modelo matemático: nos modelos físicos, as imitações destinam-se às condições que envolvem um determinado sistema. O ambiente onde o sistema se insere é moldado para atingir um determinado conjunto de condições. Este tipo de modelos é pouco utilizado na análise de sistemas, uma vez que a maioria são matemáticos. Em modelos matemáticos um sistema é representado a partir de relações lógicas e quantitativas que permitem avaliar a forma como o modelo reage e como o correspondente sistema irá reagir no futuro.

Solução analítica ou Simulação: uma solução analítica caracteriza um modelo simples que a partir de métodos matemáticos (álgebra, probabilidade, cálculo) permite obter informações ou resultados exactos acerca do modelo ou sistema. Quando os resultados podem ser obtidos de forma eficiente, a solução analítica é vantajosa por disponibilizar resultados mais precisos. No entanto, para sistemas muito complexos a sua utilização acaba por ser bastante dispendiosa, levando à sua simulação. A simulação avalia numericamente um modelo onde os *inputs* são recolhidos e é realizada uma estimativa do valor real de certa característica.

Os modelos de simulação podem ainda ser classificados como determinísticos, estocásticos, estáticos e dinâmicos.

Modelo estocástico: Estes modelos possuem variáveis aleatórias de entrada o que lhes possibilita também saídas aleatórias. São normalmente utilizados quando pelo menos uma das características operacionais depende de uma função de probabilidade. Os resultados devem ser tratados como estimativas do sistema real.

Modelo determinísticos: caracterizam-se por serem modelos que não permitem variáveis aleatórias e as suas operações são relações exactas. São computacionalmente menos exigentes que os modelos estocásticos.

Modelo estático e Modelo contínuo: São definidos de acordo com as mesmas considerações que definem um sistema discreto ou contínuo e têm grande utilidade para análises mistas.

2.3.3 Simulação de sistemas de eventos discretos

A simulação de sistemas de eventos discretos é utilizada na análise de sistemas onde o estado das variáveis é alterado através da ocorrência de eventos ao longo do tempo. A ocorrência destes eventos define-se como acontecimentos instantâneos que alteram o estado do sistema. Os modelos são analisados por métodos numéricos aplicando procedimentos computacionais para executar os modelos matemáticos. A informação gerada pelo sistema é guardada, o que

posteriormente permite uma análise e estimativa de desempenho do sistema real (Diaz & Behr, 2010).

As componentes e organização que caracterizam uma simulação de sistemas de eventos discretos são:

Estado do sistema – Conjunto de variáveis necessárias para descrever o sistema num determinado instante.

Clock – variável que representa o tempo simulado.

Lista de eventos – lista ordenada de forma lógica que contém a sequência de eventos futuros.

Contadores estatísticos – variáveis utilizadas para guardar informação estatística sobre o desempenho do sistema.

Inicialização dos procedimentos – subprograma que inicia a simulação.

Rotina controlo de tempo – subprograma que a partir da lista de eventos determina o próximo evento e avança o *clock* para o tempo em que este irá ocorrer.

Rotina controlo de evento – subprograma que carrega as informações sobre o estado do sistema quando determinado evento ocorre.

Bibliotecas – conjunto de subprogramas usados para gerar observações aleatórias como uma probabilidade associada.

Relatórios – subprograma que gera um relatório onde os resultados das análises são apresentados de forma clara e concisa.

Main program – subprograma que controla o sistema consoante o estado em que este se encontra.

As relações lógicas entre cada rotina são mostradas na Figura 2.6.

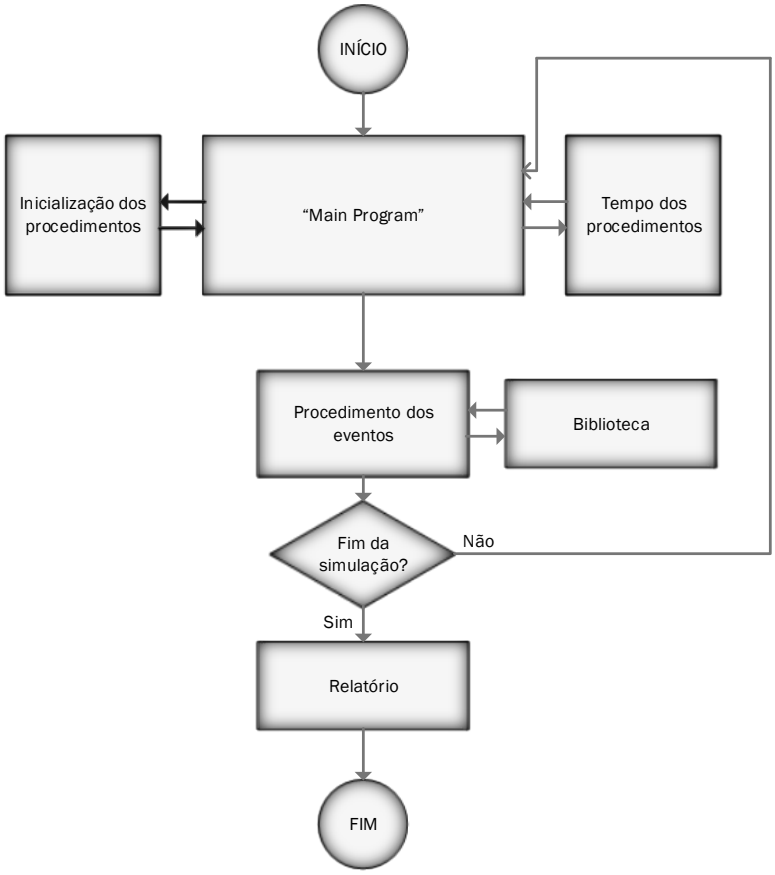


Figura 2.6 – Controlo do fluxo de mensagens da simulação de sistemas de eventos discretos

A simulação começa no tempo 0, com o programa *main* a invocar o procedimento de inicialização onde o *clock* é colocado a 0 e o estado do sistema e a lista de eventos são inicializados. Ao voltar ao programa *main* os tempos de rotina são invocados para determinar qual o evento que está a decorrer. O evento seguinte i é determinado e o *clock* é colocado no tempo referente a esse evento, voltando o controlador ao programa *main*. Os procedimentos do evento i são invocados e ocorrem três actividades: (1) o estado do sistema é actualizado, (2) as informações estatísticas são guardadas e (3) os tempos para eventos futuros são gerados e adicionados à lista de eventos. Normalmente são geradas observações aleatórias de probabilidade que irão influenciar os tempos de eventos futuros. No final deste processo verifica-se se a simulação pode ser terminada. Caso a simulação termine, é gerado um relatório com os resultados da simulação, caso contrário o controlador volta ao programa *main* e o ciclo volta a ocorrer.

2.3.4 Simulação paralela e simulação distribuída

O processo ilustrado na Figura 2.6, refere-se a simulação de processos sequenciais. Um processo alternativo é ter diferentes computadores distribuídos por uma rede local a realizar partes independentes do modelo de simulação e a interagir entre si de forma paralela ou distribuída.

Simulação paralela

A simulação paralela de eventos discretos preocupa-se em ter um modelo de simulação relacionado com as capacidades de processamento do computador. Dividindo as acções que se pretende executar em diferentes computadores é possível obter tempos de execução significativamente reduzidos. Este tipo de simulação tem grande utilidade quando é necessário tomar decisões em tempo real onde cada segundo influencia a produtividade das acções que se pretendem tomar (Groover, 2007).

Para desenvolver uma simulação em paralelo, o modelo é decomposto em processos lógicos (LP[3]). Os LPs individuais são atribuídos a processadores diferentes capazes de comunicar entre si a partir de mensagens "*timestamp*". No caso dos sistemas de manufactura, os meios de produção são modelados como uma rede de sistemas onde os eventos se encontram em filas de espera e cada computador representa uma estação de trabalho. Quando um trabalho termina numa estação, uma mensagem deve ser enviada para a estação seguinte a informar que o trabalho terminou.

Um problema crucial na simulação paralela é garantir que os eventos no modelo global são processados na sequência de tempos correcta. Por exemplo, se um trabalho chega a uma estação para ocupar o lugar do trabalho que está ainda no processo de finalização, então é importante garantir um mecanismo capaz de sincronizar os processos de substituição. Se cada LP processar todos os seu eventos numa ordem temporal, chamado de restrição de causalidade local, então pode ser mostrado que a simulação paralela produz exactamente os mesmos resultados que uma simulação que ocorra de forma sequencial num único computador.

Cada LP pode ser visto como um modelo de simulação para eventos discretos onde as suas variáveis, lista de eventos e tempos de simulação estão alocadas localmente.

Simulação distribuída

A simulação distribuída é utilizada para criar um modelo de simulação composto por dois ou mais simuladores individuais ligados a uma rede LAN ou WAN. Enquanto a simulação paralela

[3] LP – Logical processes

direcciona as suas pesquisas para o desempenho das simulações, a simulação distribuída concentra-se na interoperabilidade e na reutilização de simulações.

O interesse por este tipo de simulação derivou inicialmente dos esforços do *US Department of Defense* para desenvolver tecnologias que ligassem simuladores e Humanos. Para este propósito foi desenvolvida a tecnologia SIMNET[4], que demonstrou a viabilidade deste conceito ao criar simulações distribuídas para criar ambientes virtuais para treino militar dos soldados. Isso levou ao desenvolvimento de um conjunto de padrões para garantir a interoperabilidade entre simuladores conhecido como "Distributed Interactive Simulation". Após anos de trabalho coordenados pelo *Defense Modeling and Simulation Office* em cooperação com o IEEE, a arquitectura HLA[5] foi definida como a norma internacional do IEEE em 2000.

O HLA define uma arquitectura de *software* e não uma implementação específica nem a utilização de qualquer software ou linguagem de programação em particular. É um padrão voltado para a interoperabilidade entre simulações heterogéneas e para a reutilização de simulações. Desta forma, sistemas de simulação já existentes podem ser utilizados para criar novos sistemas com propósitos diferentes, tornando-se uma vantagem na evolução e custo de desenvolvimento de pacotes de *software* de simulação (Uygun, Öztemel, & Kubat, 2009).

A ideia principal do HLA é separar as funcionalidades específicas de cada simulador através de uma infra-estrutura de propósito geral, a "Runtime Infraestructure" (RTI). Para conseguir comunicar com o HLA e, entre si, os simuladores devem utilizar esta RTI, que serve de interface entre as estruturas específicas de cada simulador.

Cada simulador que se encontre conectado à RTI forma uma Federação. Por sua vez um federado é um participante de uma simulação distribuída baseada no padrão HLA. A Figura 2.7 ilustra um exemplo de uma federação.

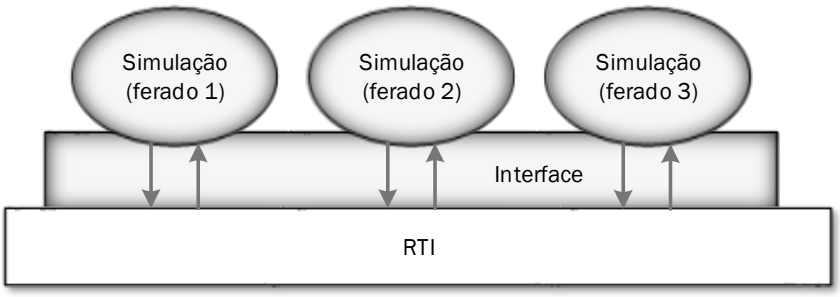


Figura 2.7 – Visão funcional de uma federação HLA (DE NEGREIROS, 2014)

[4] SIMNET – SIMulator NETworking
[5] High Level Architecture

Conforme é possível observar na Figura 2.7, uma federação HLA é dividida em três componentes. A primeira componente são as simulações ou características dos federados. A segunda é o RTI, que disponibiliza um conjunto de serviços de propósito geral que suporta a integração entre federados. Por último, a terceira componente á a interface. Esta permite a interacção entre o RTI e os federados.

2.4 Simulação de Sistemas distribuídos de manufactura

A simulação distribuída combina tecnologias de computação distribuída com as técnicas tradicionais de simulação sequencial. Embora apresentasse algumas desvantagens quando surgiu, tais como baixa velocidade de comunicação e falta de largura de banda de rede, a sua popularidade aumentou nos últimos anos devido à disponibilidade de estações de trabalhos de alto desempenho e de baixo custo e às melhorias nas tecnologias de rede. Em simulações distribuídas, o sistema simulado é decomposto num conjunto de subsistemas que são simulados em estações de trabalho individuais interligadas.

Em (Bagrodia, 1996) os sistemas simulados são vistos como um conjunto de modelos de simulação sequencial, que comunicam entre eles a partir de mensagens (*timestamp*). Um sistema de simulação sincronizado garante que cada modelo de simulação individual processe as mensagens que chegam numa ordem definida e não na ordem real.

Este requisito é referido como restrição de causalidade local. Para satisfazer esta restrição, vários protocolos de sincronização têm sido propostos na literatura. Estes protocolos podem ser amplamente classificados como conservadores ou optimistas (Fugimoto, 1990). Abordagens conservadoras impõem estritamente a restrição de causalidade local e garantem que cada modelo só processa eventos numa ordem *timestamp* não-decrescente. Pelo contrário, abordagens optimistas permitem que ocorram quebras na restrição de causalidade local, sendo capazes de detectar a falha e recuperar através do recuo até ao ponto em que a quebra ocorreu, voltando a processar novamente os eventos numa ordem com horário definido (Saad et al., 2003).

Os modelos de simulação distribuídos geograficamente comunicam entre si através da troca de mensagens, o que geralmente é conseguido através de *middleware*. *Middleware* é uma camada de *software* que contém um conjunto de serviços que permitem que vários processos, a correr em uma ou mais máquinas, possam interagir entre si através da rede. Com base em importantes normas ou produtos disponíveis, a *middleware* pode ser dividida em várias categorias como *Sockets, Remote Procedure Calls* (RPC), *Remote Method Invocation* (RMI), *Distributed Object Component Model* (DCOM) e *Common Object Broker Request Architecture* (CORBA) (Tari and Bukhres 2001). Durante os últimos anos, ferramentas baseadas em DCOM e CORBA

têm ganho alguma vantagem em relação aos seus concorrentes. Mais recentemente, estes modelos começaram a beneficiar das arquitecturas orientadas a serviços *web*.

A Simulação de SDM pode ser considerada como a simulação de manufactura que é composta por múltiplos processos de *software* que são executados independentemente em diferentes locais com a capacidade de interagir uns com os outros (Uygun et al., 2009). Estes processos têm a capacidade de simular linhas de produção completas como peças individuais de máquinas industriais (McLean, Riddick, & Lee, 2005). Embora a área da manufactura seja a maior área de aplicação da simulação, a simulação sequencial tradicional sozinha pode não ser suficiente para a simulação de empresas de distribuição de manufactura altamente complexas (Law & McComas, 1999).

O uso da simulação distribuída permite que cada simulador individual:

- Esconda qualquer informação do utilizador na sua implementação
- Simule vários sistemas de manufactura em diferentes graus de abstracção
- Ligue modelos de simulação desenvolvidos através de diferentes pacotes de *software* de simulação
- Tire proveito de poder de processamento adicional
- Tenha acesso simultâneo a execução de modelos de simulação em diferentes locais
- Reutilize modelos de simulação existentes com pequenas modificações (McLean et al., 2005), (McLean & Riddick, 2000), (Saad et al., 2003).

A simulação de sistemas distribuídos pode assim ser dividida em três níveis: Macro, Meso e Micro. Estes termos são utilizados em diferentes áreas científicas, como por exemplo na Gestão de Recursos Humanos. O nível Macro destina-se a processos estratégicos ou organizacionais, enquanto os níveis micro caracterizam grupos pequenos ou máquinas individuais de identidade partilhada. O nível meso aparece entre os dois níveis referidos, sendo composto por conjuntos de indivíduos do nível Micro e por fazer parte dos elementos do nível Macro.

A modelação manufactura num sistema de níveis é normalmente referido como a simulação de nível macro em que o comportamento de um sistema de manufactura global é analisado. O nível micro é usado em simulações individuais, como processos de maquinaria ou robots. Existe assim uma brecha entre os níveis, tendo em conta que o sistema macro não pode ser explicado com actividades do nível micro. Aplicar o nível meso ao contexto da modelação e simulação estabelece a ponte entre estes níveis. Um estudo de nível meso foca-se então na forma que um grupo de actividades de nível micro se comporta num ambiente de nível macro. A Tabela 2.3 ilustra brevemente os níveis micro, meso e macro no contexto da modelação e simulação.

Tabela 2.3 – Descrição dos níveis macro, meso e micro na Simulação de sistemas de manufactura

Nível de modulação e simulação	Descrição
Nível Micro	Unidades de manufactura, máquinas individuais, métodos de produção e peças. As áreas de simulação são por exemplo CAD[6] e CAM[7].
Nível Meso	Etapas de produção, áreas físicas focadas na cooperação entre as unidades de manufactura. Exemplos típicos são as células ou estações de trabalho locais compostas por um ou mais controladores e vários robôs.
Nível Macro	Etapas de manufactura. A informação sobre os processos dos materiais ao longo das diferentes etapas de manufactura. Por exemplo uma cadeia de fornecimento de um produto.

A Figura 2.8 apresenta os níveis macro, meso e micro de modelação e simulação. Os *inputs* para o nível macro são, por exemplo, os pedidos dos clientes, que consistem em alterar volumes e variações em partes de uma ou mais famílias de peças, e os outputs são, por exemplo, prazos de entrega e fiabilidade aos pedidos dos clientes. A etapa da manufactura representa o nível meso, que usa trabalhadores e lida com material durante a etapa da manufactura. A unidade de processamento é um exemplo de uma característica de nível micro, que pode ser vista como um processo individual em que cada característica especifica da peça é realizada usando certos métodos de maquinaria.

Figura 2.8 – Ilustração dos Níveis Macro, Meso e Micro

A interacção entre níveis pode ser implementada sob a forma de serviços. A solicitação de um serviço no nível macro é o pedido de um cliente. O modelo de nível macro envia um pedido de

[6] *computer-aided design*
[7] *computer-aided manufacturing*

serviço para o nível meso. O pedido do cliente é dividido em partes individuais, e o pedido de serviço é um lote de ordens de uma ou mais partes. A ordem de lotes pode ser uma combinação de vários pedidos de nível macro, dependendo da natureza da etapa de manufactura do nível meso. A resposta do nível meso pode incluir, por exemplo, horários de trabalho, o tempo que o serviço durará e o custo do serviço. Se houver mais do que um possível fornecedor no nível meso, a decisão de qual será seleccionado é tomada no nível macro para cada pedido de serviço independentemente. Do nível meso para o nível micro, o serviço é dividido de forma semelhante em múltiplos subserviços. O pedido de serviço é para uma peça de trabalho que inclua as características necessárias para ser produzido. À semelhança do que acontece no nível macro, o fornecedor é decidido no nível meso se existirem várias possibilidades (Nylund & Andersson, 2010).

2.5 Etapas para o estudo de uma simulação

Depois de um estudo detalhado sobre simulação de eventos discretos é pertinente identificar os passos necessários de *design* e análise de um sistema complexo que se pretende simular. A Figura 2.9 ilustra as etapas que decorrem num projecto de modelação e simulação dum determinado sistema.

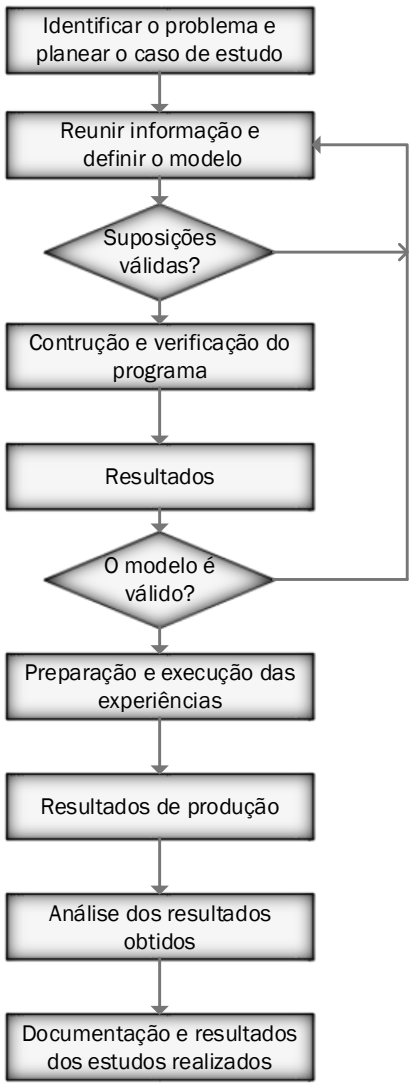


Figura 2.9 – Etapas de um caso de estudo de uma simulação

2.6 Exemplos de pacotes de *software* de Simulação

Para que se possa transmitir uma clara impressão sobre a situação corrente em termos de simulação, importa referir alguns produtos comerciais que são bastante utilizadas, tanto a nível industrial como científico. Focar-se-á apenas um deles mais em detalhe, sendo a ferramenta utilizada na implementação da solução proposta.

A Tabela 2.4 ilustra alguns dos pacotes de *software* de simulação mais utilizados:

Tabela 2.4 – Características de algumas ferramentas de simulação.

Ferramentas	Características
Arena	Baseia-se na linguagem de programação SIMAN; Permite a implementação de modelos discretos e contínuos; Construção de modelos é feita a partir de blocos lógicos; Permite a construção de modelos sem qualquer linha de código; É compatível com ferramentas do Windows tais como: Word, Excel e CAD (Kelton, Sadowski, & Sadowski, 2002).
Extend	Linguagem de simulação do tipo orientada a objectos; Permite implementar modelos discretos e contínuos; Existência de uma biblioteca de modelos que podem ser utilizados ou ajustados às necessidades dos utilizadores; Os blocos dividem-se em blocos-código, blocos programáveis, e blocos hierárquicos, constituídos num conjunto de blocos (dos Santos Lopes, 2008).
AutoMod	Capacidade de trabalhar dados físicos e lógicos em simultâneo; Processamento lógico detalhado; Os recursos são definidos pelas suas características e tempos de processamento; Permite detalhar aspectos como velocidade aceleração e pontos de controlo remoto (Materials, 2013)
ProModel	Sistemas muito flexíveis permitindo construir sistemas logicamente complexos; Bons recursos de análise estatística e interface simples; Principais elementos: • Locais: elementos fixos onde as operações se desenrolam • Entidades: elementos móveis dos locais • Chegadas: definição da chegada de entidades ao sistema • Processos: tempos de operação, recursos lógicos, movimentações • Recursos: elementos da operação • Rede de encaminhamento: trajecto onde se movimentam os recursos (Harrell, Ghosh, & Bowden, 2000)

Arena

O *software* Arena tem como tecnologia de base incorporada a linguagem de simulação SIMAN e combina a facilidade de utilização dos simuladores de alto nível com a flexibilidade das linguagens de simulação e linguagens de programação genéricas (Kelton et al., 2002). Funciona no sistema operativo Windows e é compatível com outros pacotes de *software*, tais como Word, Excel e ferramentas CAD. Permite ainda simular sistemas discretos e contínuos.

A construção de modelos baseia-se em ambientes gráficos e é feita a partir de blocos lógicos acabando por não necessitar de linhas de código.

Utiliza uma arquitectura hierárquica na modulação, onde módulos são definidos a partir de outros módulos. Assim, os processos são desenvolvidos e verificados, podendo ser posteriormente utilizados para desenvolver módulos de processos de níveis hierarquicamente superiores.

O ambiente de simulação do Arena fornece aos utilizadores ferramentas de: Simulação, análises de *inputs*, *outputs* e processos, gestão de cenários, fontes de dados externos e animação (Marques, 2007). A Figura 2.10 apresenta um modelo simulado no Arena.

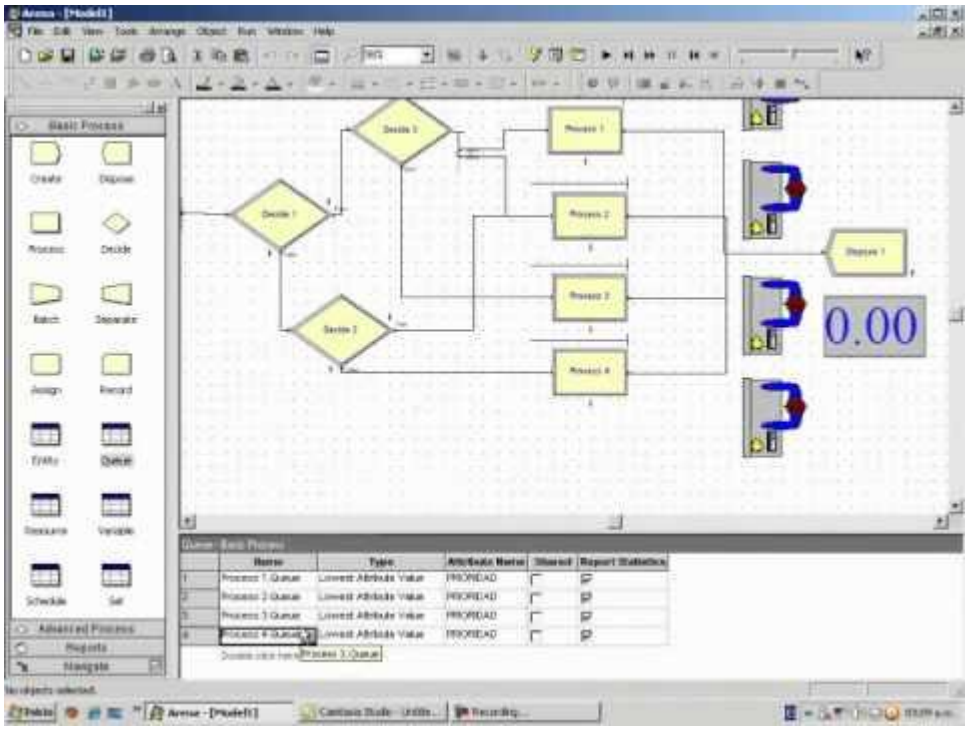


Figura 2.10 – Simulação utilizando o *software* ARENA

3 Desenvolvimento da Infra-estrutura de Simulação Distribuída

3.1 Identificação e caracterização do problema

Neste capítulo pretende-se caracterizar os principais problemas e contextualizar os aspectos mais importantes na solução do problema proposto.

A ausência de ferramentas que permitam modelar e simular problemas no âmbito dos SDM, despertou a necessidade da criação de uma ferramenta que permitisse a simulação agregada dos seus processos de negócio e respectivos processos de manufactura. A maioria dos pacotes de simulação actuais apresentam algumas lacunas, nomeadamente quando se pretende simular processos que decorrem em níveis distintos duma empresa.

Como a criação de uma ferramenta de raiz é um processo bastante oneroso, a solução encontrada passa por combinar várias ferramentas já existentes que em conjunto seriam capazes de simular os diversos níveis funcionais de um SDM.

Para satisfazer tal necessidade, foram encontrados alguns desafios ao nível da integração e interoperabilidade entre os diversos pacotes de simulação. No entanto, o recente desenvolvimento das TIC permitem lidar com estes desafios. Quando se procede à simulação de um sistema, pretende-se também criar uma ilustração o mais próxima possível do ambiente que se está a imitar, para que, dos processos realizados e dos dados adquiridos, possam ser feitas previsões sobre o comportamento do sistema real. Dado que os modelos simulados se referem a SDM, isso implica que a própria infra-estrutura de simulação seja distribuída, assim como, se as suas componentes são heterogéneas, os nós locais de simulação são também heterogéneos, permitindo uma melhor adaptação entre a infra-estrutura de simulação e as distintas componentes dos modelos simulados.

São inúmeros os casos de sistemas heterogéneos, onde os seus controladores estão distribuídos localmente e funcionam de forma independente. No entanto, quando os simuladores se encontram em redes de trabalho diferentes, a forma como os processos são coordenados é ainda um obstáculo. Admitindo que por cada rede local existe um controlador capaz de coordenar as estações de trabalho distribuídas na mesma rede, é importante garantir uma forma de comunicação entre esses controladores.

Um outro obstáculo bastante comum em sistemas de simulação é a diferença entre a natureza dos processos que se pretende simular. Quando se pretende simular as diferentes etapas do ciclo do produto, existem fases que decorrem em níveis diferentes e que possuem características próprias onde a utilização de ferramentas de simulação já adaptadas a cada nível podem melhorar a qualidade da simulação. A Figura 3.1 demonstra os problemas identificados e caracterizados dentro de uma abordagem referenciada no capítulo anterior em relação aos níveis macro, meso e micro.

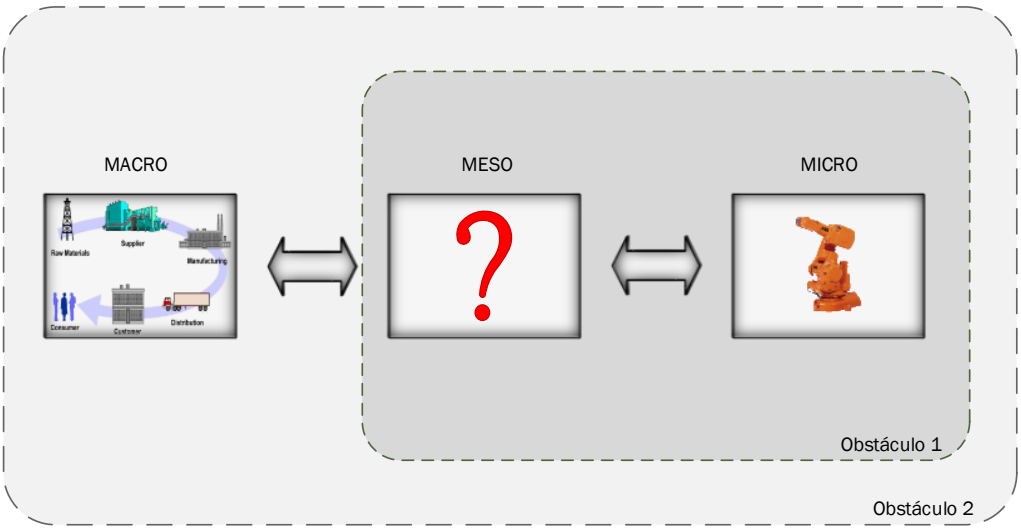


Figura 3.1 – Identificação do problema

3.2 Requisitos do sistema

> *Os requisitos de um sistema são fundamentalmente uma descrição das funções e restrições, que permitem a compreensão do problema do sistema que se pretende solucionar. Os requisitos podem guiar tanto o processo de desenvolvimento de software quanto o processo de aquisição. (Sommerville, Melnikoff, Arakaki, & de Andrade Barbosa, 2003)*

3.2.1 Requisitos funcionais

Os requisitos funcionais são declarações de funções do sistema, da forma como o sistema reage a necessidades externas e comportamentos em determinadas situações. Descrevem assim as funcionalidades ou serviços que o sistema fornece. De acordo com o referido anteriormente os requisitos funcionais identificados foram os seguintes:

Requisitos de Sistema de simulação

Tabela 3.1 – Requisitos funcionais do sistema de simulação

RF	Descrição
RF_s1	Simular o comportamento de processos de manufactura locais
RF_s2	Simular o comportamento de nós de manufactura
RF_s3	Permitir a simulação de distintos cenários de SDM
RF_s4	Incorporar simultaneamente os níveis locais e globais no mesmo cenário de simulação

Requisitos de Serviços

Os serviços que o sistema está apto a fornecer estão relacionados aos produtos. Os serviços são processos necessários quando se pretende simular a produção de dado produto, podendo ser compostos por determinado número de processos ou então por outros serviços, acabando numa combinação de serviços. A Tabela 3.2 identifica os requisitos funcionais.

Tabela 3.2 – Requisitos funcionais dos serviços

RF	Descrição
RF_pc1	Oferecer serviços ao cliente
RF_pc2	Processar informações sobre as BD
RF_pc3	Conhecer todos os nós de simulação
RF_pc4	Funcionar de forma distribuída
RF_pc5	Armazenar informação sobre as características dos produtos numa BD

Requisitos de nós de simulação local

Os nós de simulação encontram-se ligados a uma rede local onde é permitida a troca de informação entre eles. Cada nó é composto por um controlador remoto e por um simulador local. Na Tabela 3.3 estão os requisitos funcionais dos nós locais.

Tabela 3.3 – Requisitos funcionais dos nós de simulação locais

RF	Descrição
RF_nc1	Identificar os nós que fabricam determinado produto
RF_nc2	Identificar os produtos que são produzidos em cada nó
RF_nc3	Identificar as tarefas associadas à manufactura de determinado produto

3.2.2 Requisitos não funcionais

Os requisitos não funcionais são aqueles que não dizem respeito directamente às funções específicas fornecidas pelo sistema (Sommerville et al., 2003). Ou seja, ao invés de estabelecer comportamentos ou funções particulares de um sistema, especificam os critérios para avaliar a

qualidade e o funcionamento de um sistema. Enquanto um requisito funcional determina o que um sistema deve fazer, um requisito não funcional determina como um sistema deve ser feito.

Escalabilidade e adaptabilidade: o sistema deverá funcionar adequadamente, mesmo se a quantidade de nós aumentar, ou seja, o número de nós não poderá afectar do desempenho do sistema e este deverá permitir a expansão do sistema.

Resiliência: No caso de um dos nós deixar de funcionar, ou não responder, o sistema deverá ser capaz de manter um desempenho adequado, continuando os restantes nós a funcionar com normalidade. Este requisito não se aplica ao nó central.

Interoperabilidade: Cada nó de uma simulação deverá conseguir interagir com os restantes nós, mesmo que sejam de natureza distintas ou estejam localizados remotamente.

Similaridade: o sistema deverá ser capaz de realizar simulações de ambientes e ferramentas reais. Deverá ter em atenção as necessidades e características do sistema real para que este possa ser simulado.

Eficiência e Usabilidade: O sistema deverá devolver resultados correctos e num tempo minimamente aceitável. O comportamento deverá ser estável e previsível ao utilizador.

3.3 Enquadramento da Solução

A satisfação dos requisitos funcionais formulados previamente implica a construção de um mecanismo composto por componentes dedicadas e à execução das funcionalidades correspondentes aos requisitos funcionais formulados. Desta forma, o mecanismo que se destina ao controlo remoto dos nós locais deverá possuir uma arquitectura que inclua as componentes de simulação de nós locais, uma componente de coordenação centralizada (*engine*) e as componentes de suporte à interacção com os utilizadores finais.

É necessária a existência de um núcleo central capaz de gerir as diferentes actividades do sistema e de comunicar com os restantes nós simulação. Cada nó de simulação irá receber mensagens vindas do nó central e, remotamente, irá controlar o programa que foi implementado na ferramenta de simulação. A troca de mensagens é realizada a partir de uma rede local onde todos os nós se encontram ligados como mostra a Figura 3.2.

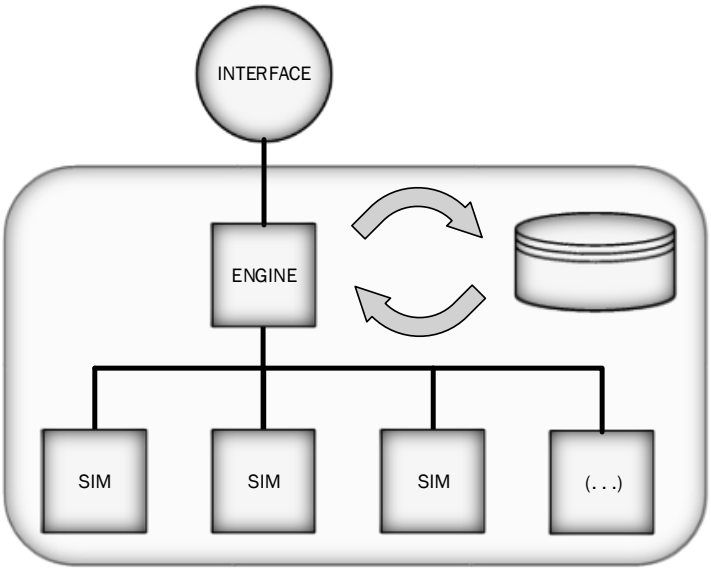


Figura 3.2 – Procedimentos para a implementação de um controlador local

O e*ngine*, ou nó central, define a arquitectura do modelo. A lista de produtos que o sistema é capaz de simular encontra-se guardada numa base de dados. Esta lista é fornecida ao utilizador onde poderão ser escolhidos os produtos que pretende simular. Após a escolha do produto, a base de dados é consultada novamente e são recolhidas as informações sobre as tarefas necessárias para a simulação da produção do produto escolhido. De forma sequencial, as mensagens serão enviadas aos diferentes nós de simulação. O *engine* funciona como cliente dos nós de simulação, onde ao enviar um pedido aguarda *feedback* por parte dos servidores. Este processo é ilustrado na Figura 3.3.

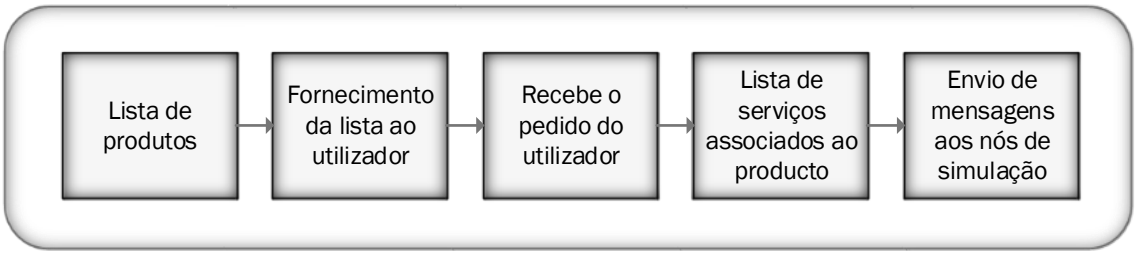


Figura 3.3 – Definição do *Engine*

Com o nó central a funcionar como cliente, os nós de simulação locais são servidores, prestando serviços ao cliente quando este faz a sua solicitação. Conforme mostra a Figura 3.4, em cada nó encontra-se a ferramenta de simulação específica, com os respectivos programas, e um controlador remoto. A ferramenta tem a capacidade de produzir um determinado número de tarefas de manufactura de uma estação de trabalho, que pode ser composta por um ou mais robôs. As tarefas são definidas na linguagem própria da ferramenta. No caso do controlador, este espera por pedidos vindos do cliente e quando solicitado, manda executar o programa

embutido no *software* de simulação. Quando a ferramenta termina a simulação, o controlador remoto é avisado da conclusão da simulação por parte da ferramenta.

Figura 3.4 – Definição do nó de simulação

Quando o nó local terminar a execução de um pedido, deverá informar o cliente da conclusão, sendo enviada uma mensagem de fim da produção por parte daquele simulador. A Figura 3.5 apresenta o diagrama de sequência da solução proposta, que ilustra a funcionalidade aqui descrita.

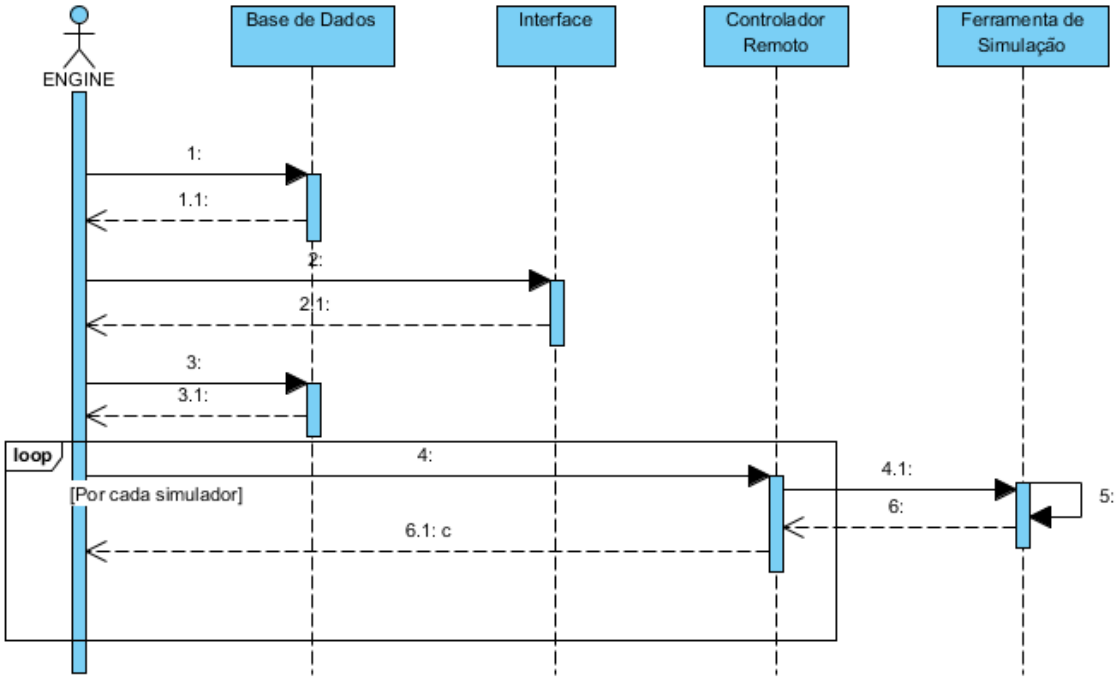


Figura 3.5 – Diagrama de sequência e satisfação dos pedidos para simulação

Para realizar a simulação global do sistema, a arquitectura anterior é reutilizada, realizando-se pequenas alterações na parte do cliente, para que o utilizador possa controlar o sistema numa

perspectiva mais estratégica, e assim utilizar outros tipos de processos. Desta forma a arquitectura deverá utilizar tipos de processos do nível estratégico e, quando necessário, controlar processos externos mais detalhados, ou seja nós locais, e utilizar os resultados obtidos dessas fontes externas nas estimativas finais. Os nós podem operar na rede local de trabalho ou encontrarem-se geograficamente distribuídos. A Figura 3.6 ilustra a arquitectura referida da solução global.

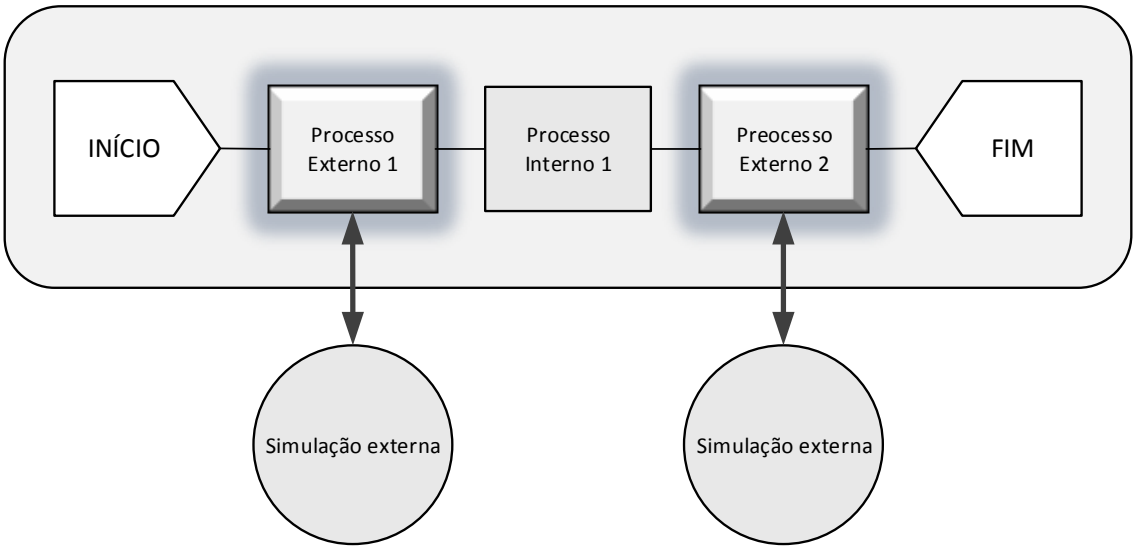


Figura 3.6 – Procedimento da solução proposta a um nível estratégico

3.4 Implementação da solução

Conforme referido anteriormente, para se chegar à solução proposta, é necessário garantir um número de funcionalidades nas diferentes etapas da implementação. Partes destas etapas funcionam de forma independente das restantes, processando as suas funções localmente. De seguida serão então apresentados os métodos utilizados na implementação referida.

Como referido em algumas abordagens estudadas anteriormente, o sistema pode ser visto como um conjunto de subsistemas cooperativos, que quando integrados formam o sistema global descrito. Deste modo a descrição da implementação dividir-se-á em três partes:

- Nós de simulação: Onde se encontra a ferramenta de simulação;
- Infra-estrutura de comunicação: a forma como a troca de mensagens entre nós se realiza.
- *Engine* ou nó central: responsável pela coordenação do sistema;

Em cada parte serão explicadas as ferramentas e tecnologias utilizadas na implementação da solução.

3.4.1 Nós de simulação

Os nós de simulação são constituídos por computadores ligados à rede que possuem uma ferramenta capaz de simular uma estação de trabalho. As acções que se pretendem executar são programadas directamente na ferramenta de simulação e são controladas remotamente por um controlador, conforme ilustra a Figura 3.4. As ferramentas utilizadas foram: o RobotStudio, como Software de simulação disponibilizada pela ABB e o Visual Studio 2012 utilizado para a criação destes controladores remotos (ABB, 2012).

A ABB fornece ferramentas constituídas por um conjunto de bibliotecas que permitem aos programadores criar interfaces para manipular controladores personalizados. Estas aplicações podem funcionar como clientes locais ou remotos. Tendo em conta os interesses desta dissertação optou-se por utilizar a plataforma PC SDK[8], que permite criar um cliente remoto, possibilitando controlar diferentes robôs a partir de uma estação e a sua integração em sistemas de natureza diferente.

Esta plataforma utiliza bibliotecas que estão organizadas por domínios. Estas bibliotecas, que são suportadas pela *framework* .NET, são importadas para a ferramenta Microsoft Visual Studio, onde é criada a aplicação que permite a interacção do utilizador com o simulador, recorrendo à linguagem C#. A Figura 3.7 demonstra a arquitectura da plataforma PC SDK. Esta comunicação é conseguida através da utilização das classes referidas, com a particularidade do simulador estar programado em linguagem RAPID.

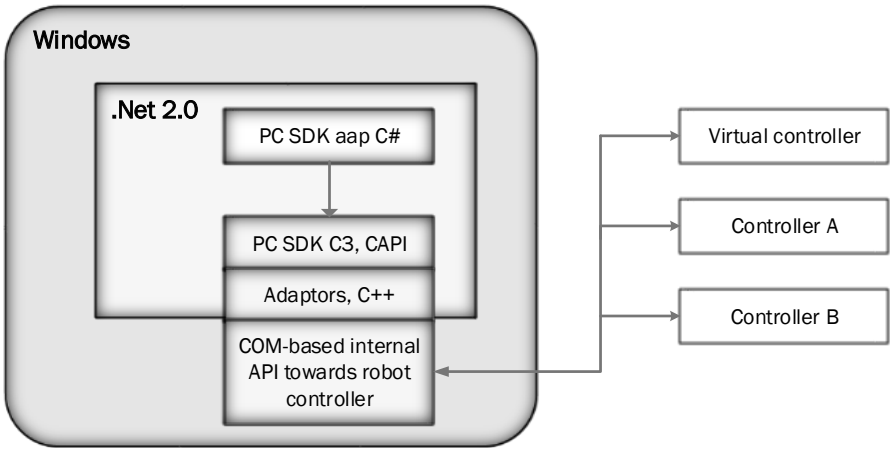


Figura 3.7 Arquitectura PC SDK

[8] *PC Software Development Kit*

O modelo de classes utilizadas para aceder às funcionalidades do controlador estão organizadas por domínios como mostra a Figura 3.8.

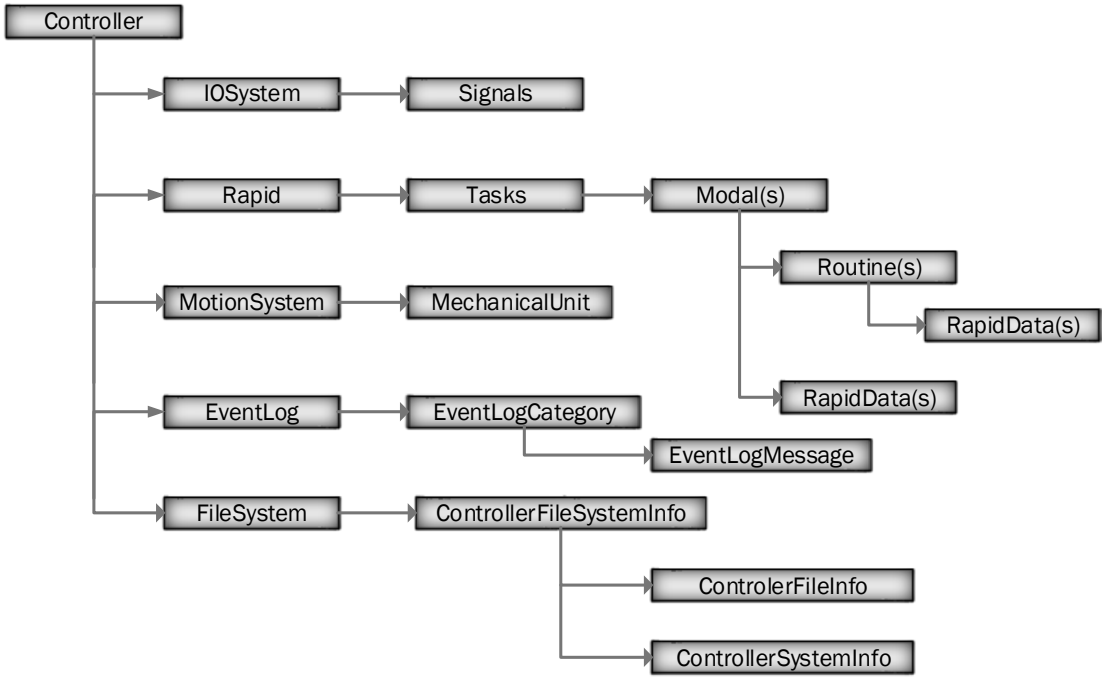


Figura 3.8 - Aplicação do controlador (CAPI[9]) (Pires, 2005)

3.4.2 Comunicação

A comunicação é um aspecto fundamental para a implementação da solução. A troca de informações entre os nós é indispensável para garantir a cooperação entre eles e a interoperabilidade do sistema. É então importante criar uma ponte entre o *engine* e os nós de simulação, a fim de conseguir que a troca de informações seja possível. Para atingir este objectivo, os computadores têm que se encontrar ligados à mesma rede de trabalho para poderem ser encontrados a partir do IP.

Para construir uma forma de comunicação entre os nós controladores, foram utilizadas as bibliotecas WCF[10] da Microsoft, contidas na .NET *framework* 3.5, que se baseia numa arquitectura orientada a serviços (SOA). A arquitectura SOA, que fundamentalmente é um paradigma utilizado para organizar serviços em competências distribuídas que estão sobre o domínio de ferramentas diferentes, permite transformar aplicações fixas em componentes de *software* flexíveis, possibilitando a interoperabilidade entre diferentes tecnologias (Cibraro, Claeys, Cozzolino, & Grabner, 2010).

[9] *Controller API*
[10] *Windows Communication Foundation*

A troca de mensagens é realizada no padrão mais comum de *request/reply*. O cliente solicita informações sobre um serviço através do envio de uma mensagem e aguarda pela resposta do servidor como é ilustrado na Figura 3.9. Este padrão é muito utilizado para organizar a comunicação entre aplicações.

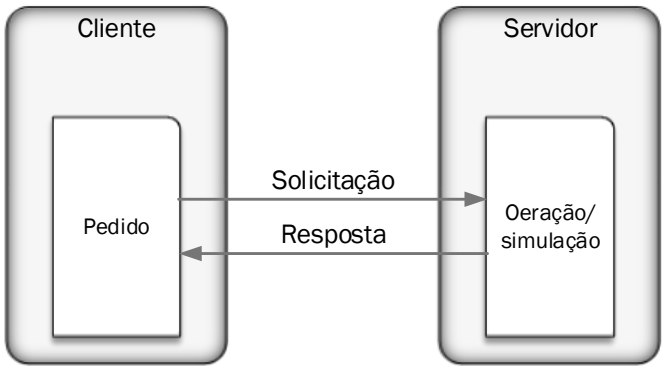


Figura 3.9 – Padrão de mensagens Solicitação/resposta

A ferramenta WCF, para além do que foi dito anteriormente, caracteriza-se ainda pela extensibilidade dos serviços.

3.4.3 *Engine* ou nó central

O *engine* é o nó central que coordena todo o sistema, funcionando como o "cérebro" da solução. É responsável por disponibilizar informações ao utilizador das acções que o sistema realiza e satisfazer as suas necessidades. As informações referentes às capacidades do sistema encontram-se armazenadas numa base de dados em SQL. A base de dados possui duas lista de Produtos e Processos, respectivamente, que associa as actividades necessárias à produção de um produto. O esquema da base de dados é ilustrado na Figura 3.10.

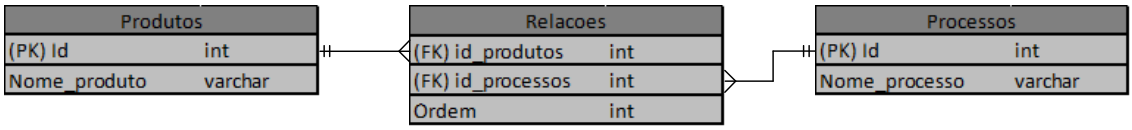


Figura 3.10 – Esquema da base de dados das actividades do sistema

Na tabela Produtos são inseridos os produtos que o sistema pode simular. A tabela processos são acções que caracterizam cada simulador local. Uma vez que o os simuladores locais são programados localmente, as suas acções estão definidas previamente e são guardadas na base de dados. Na tabela relações, os processos são associados aos produtos e vice-versa. Ao associar um processo a um produto é também guardada a ordem em que este processo acontece, tornando a simulação sequencial, de acordo com o planeamento de fabrico do produto.

A interacção com o utilizador é feita a partir de uma interface onde os produtos existentes são carregados e fornecidos ao utilizador. Depois de escolhido o produto, é consultada novamente a base de dados e são enviadas mensagens sequenciais para os nós locais. O *Engine* funciona como cliente dos restantes nós de simulação, que ao fazer um pedido espera que o nó de simulação confirme a execução dos processos.

Simulação do nível global

Na simulação do nível global pretende-se integrar os níveis macro e micro, permitindo simular em simultâneo processos característicos de cada nível num só cenário. É então possível combinar simulações das componentes estratégicas com as componentes de manufactura, tornando o *engine* um conjunto de processos de um nível macro.

O modelo de simulação macro é especificado dentro do *software de simulação* Arena. Os processos que possui são acções definidas no nível meso, onde no final da simulação podem ser obtidos resultados sobre a execução de cada processo. Em alguns desses processos, existem elementos que ficarão associados aos nós locais onde a simulação da manufactura ocorrerá. A Figura 3.11 ilustra um cenário de simulação que inclui o Arena para o modelo de simulação estratégico e os consequentes processos, do nível meso, que desencadeiam as simulações nos nós locais.

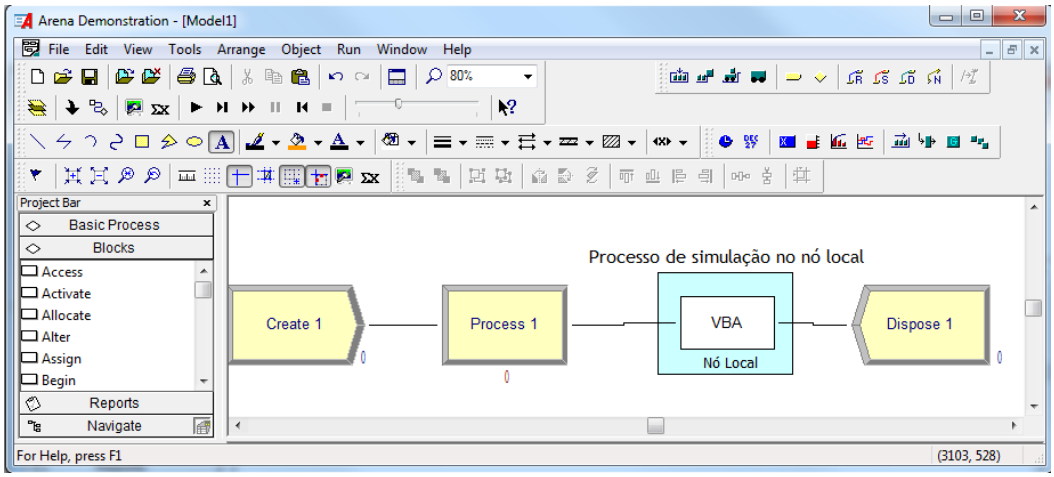


Figura 3.11 – Simulação do nível estratégico na ferramenta Arena

Obtém-se assim um sistema de simulação distribuída que engloba simultaneamente os diferentes níveis dum SDM.

4 Validação

> *O processo de verificação refere-se ao conjunto de tarefas que garantem que o software implementa correctamente as funções para o qual este foi desenvolvido. A validação refere-se ao conjunto de tarefas que avaliam se o software satisfaz os requisitos do cliente (Pressman, 2011).*

A validação da solução é composta por diferentes partes. Inicialmente, os processos implementados são testados em separado para perceber se o comportamento corresponde ao resultado esperado. Depois da validação individual, é realizada uma validação ao sistema global, onde são feitas comparações com simulações individuais realizadas inicialmente. Como foram encontrados dois obstáculos a validação dividir-se-á também em duas partes, apresentando as soluções para cada um deles.

4.1 Nós de simulação

Nos nós de simulação é importante comparar os *outputs* obtidos por um controlador remoto com os resultados obtidos com uma simulação realizada internamente na ferramenta de simulação. A ferramenta de simulação utilizada foi o RobotStudio, sendo que a estação trabalho utilizada foi a mesma em ambos os casos.

Foi implementado um controlador remoto em linguagem C# na ferramenta da Microsoft Visual Studio, onde foram utilizadas as seguintes bibliotecas da ABB:

- ABB.Robotics;
- ABB.Robotics.Controllers;
- ABB.Robotics.Controllers.Discovery
- ABB.Robotics.Controllers.RapidDomain.

As figuras que se seguem ilustram o controlador referido e a forma de utilização do mesmo. Como podemos verificar na Figura 4.1, o controlador procura os sistemas que se encontram ligados na rede de trabalho local, criando uma lista com as características referentes a cada um. O código referente à implementação do controlador pode ser consultado em anexo (Anexo 2).

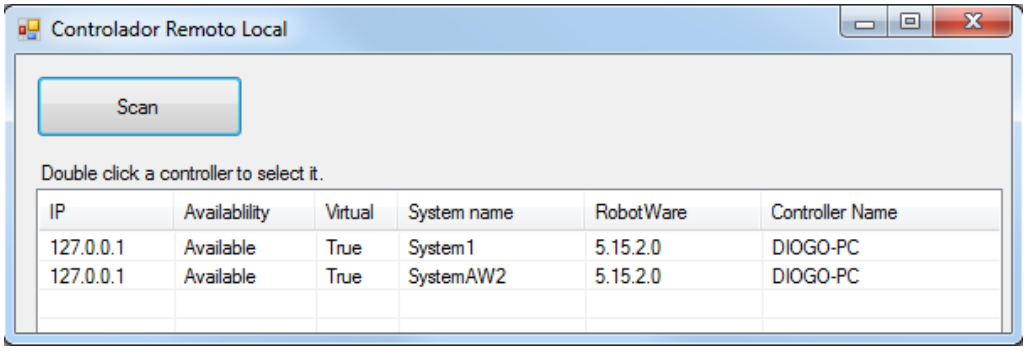


Figura 4.1 - *Scan* dos sistemas de simulação na rede de trabalho local

Cada um destes sistemas representa uma estação de trabalho concreta. Por exemplo, o sistema "System 1" representa um nó que simula um processo de manufactura como ilustra a Figura 4.2. O sistema "SystemAW2" representa o sistema ilustrado na Figura 4.3.

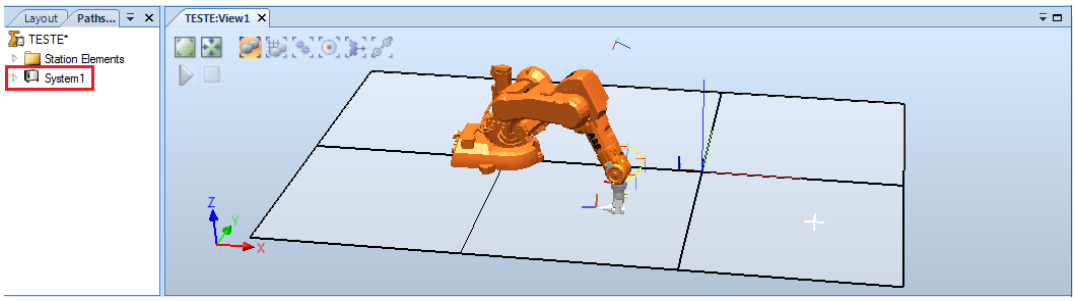


Figura 4.2 – Simulação da estação de trabalho do sistema "System 1"

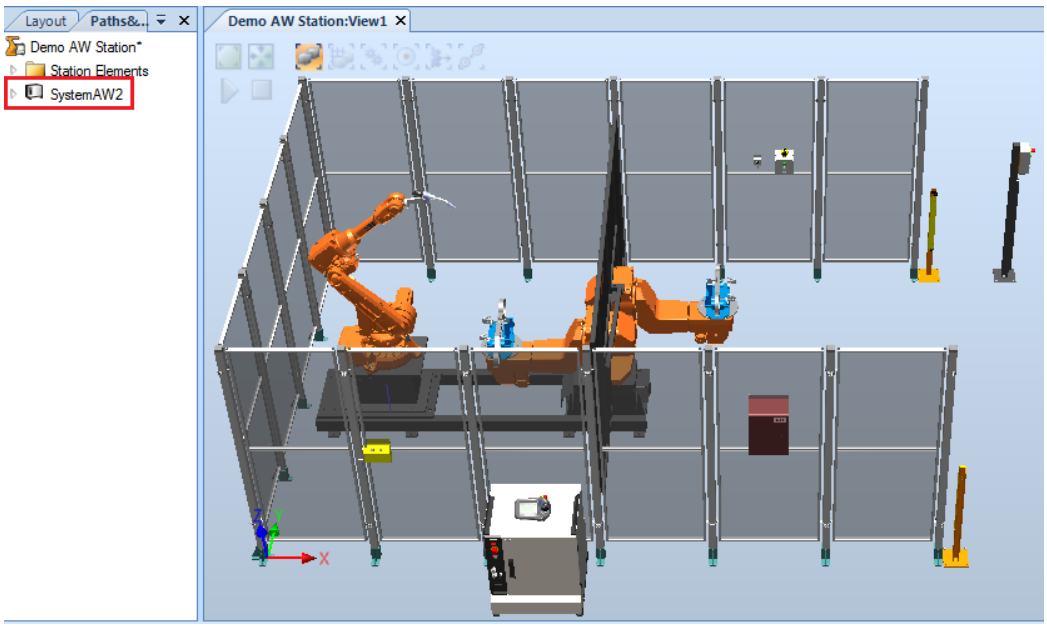


Figura 4.3 – Simulação da estação de trabalho do sistema "SystemAW2"

Na lista disponibilizada o utilizador pode seleccionar o sistema que pretende simular. Quando a opção “start production” é seleccionada o controlador inicia a simulação do respectivo sistema como ilustra a Figura 4.4.

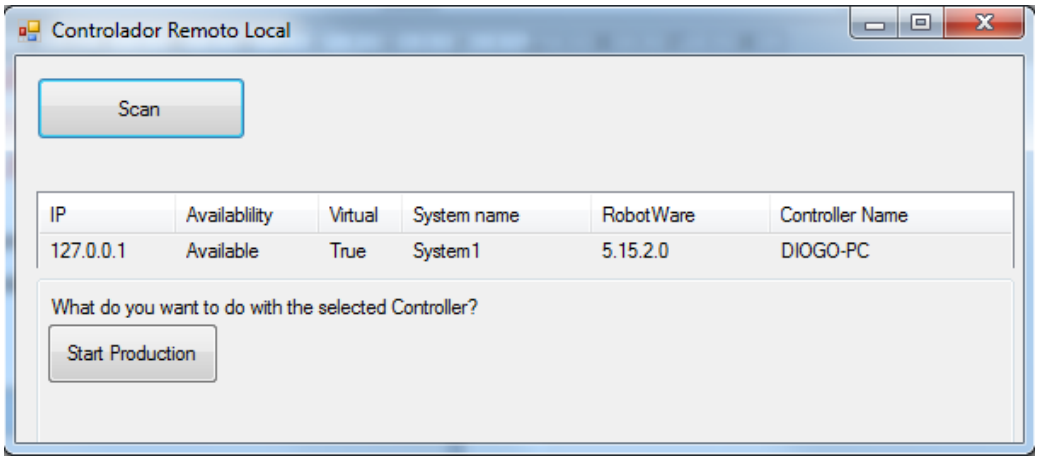


Figura 4.4 – Iniciar produção do sistema de simulação seleccionado

Enquanto a simulação decorre o controlador verifica o estado em que esta se encontra e, quando terminada, o utilizador é informado sobre a sua conclusão (Figura 4.5). Esta informação é obtida através da função RapidExecutionStatus, que avalia o estado em que o sistema se encontra.

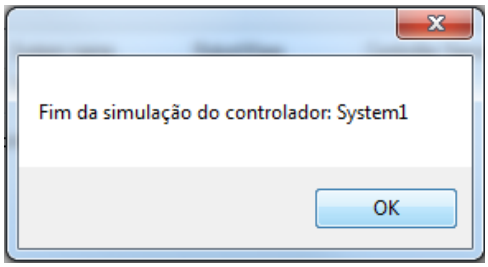


Figura 4.5 – Mensagem de aviso do fim da simulação

Na Tabela 4.1 as mensagens de output que se obtêm na ferramenta RobotStudio executadas de forma local e remota podem ser comparadas.

Tabela 4.1: Comparação entre as mensagens de output da execução local e execução remota

Controlo local	Output Show messages from: All messages System1 (Station): Program pointer has been reset — 20-09-2014 03:04:59 — Event Log System1 (Station): Regain start — 20-09-2014 03:04:59 — Event Log System1 (Station): Regain ready — 20-09-2014 03:04:59 — Event Log System1 (Station): Program started — 20-09-2014 03:04:59 — Event Log System1 (Station): Corner path failure — 20-09-2014 03:05:00 — Event Log System1 (Station): Program stopped — 20-09-2014 03:05:01 — Event Log
Controlo remoto	Output Show messages from: All messages System1 (Station): Regain start — 20-09-2014 03:02:09 — Event Log System1 (Station): Regain ready — 20-09-2014 03:02:09 — Event Log System1 (Station): Program restarted — 20-09-2014 03:02:09 — Event Log System1 (Station): Corner path failure — 20-09-2014 03:02:11 — Event Log System1 (Station): Program stopped — 20-09-2014 03:02:11 — Event Log

Realizando as simulações em ambos os casos, os resultados da simulação obtidos são os mesmos como é possível verificar pelo output de mensagens obtido em cada caso. O código RAPID utilizado na implementação do sistema pode ser consultado em anexo (Anexo 1).

4.2 A componente WCF

A componente WCF é uma arquitectura orientada a serviços utilizada na troca de mensagens entre o cliente e o servidor. A partir dela é possível aceder remotamente a controladores que não se encontrem ligados na mesma rede de trabalho. Na Figura 4.6, é possível observar o servidor remoto a aguardar solicitações vindas do cliente. Quando as mensagens são recebidas o servidor processa a mensagem e devolve uma resposta ao cliente.

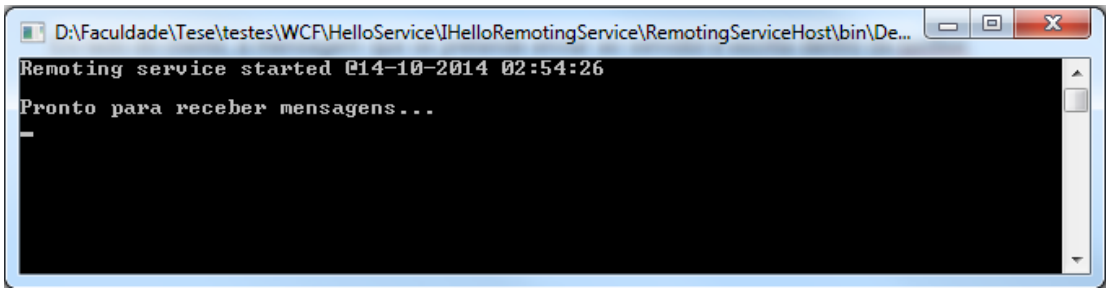


Figura 4.6 – Servidor iniciado

Do lado do cliente, a mensagem que se pretende enviar ao servidor é escrita dentro da *textbox*. Depois de enviada a mensagem o servidor executa os processos programados e devolve a resposta ao cliente como é ilustrado na Figura 4.7.

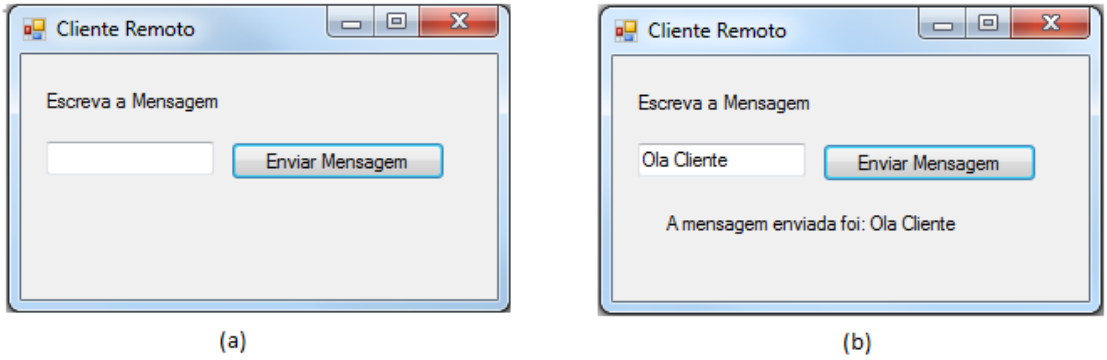


Figura 4.7 – (a) Envio de uma mensagem ao servidor; (b) Recepção da mensagem enviada

4.3 Integração do RobotStudio e a componente WCF

A implementação do controlador remoto e da componente WCF foram realizadas e testadas individualmente. Como foi possível comprovar, através dos testes apresentados anteriormente, o resultado individual alcançado é o esperado. É importante agora integrar as duas partes, para que um cliente possa controlar as simulações que um nó local permite executar remotamente.

Utilizando os exemplos antes referidos, foram realizadas algumas alterações nas "*forms*" existentes e foi assim inserida uma base de dados onde são inseridos os controladores que se encontram disponíveis na rede local para realizar a simulação. Os controladores que se encontram activos são então colocados numa *ComboBox*. À solução RemotingService foi adicionada a classe Robot_Studio onde são processadas as funções para controlar remotamente a estação criada na ferramenta RobotStudio. Na Figura 4.8 é possível ver as alterações realizadas assim como o output da simulação local.

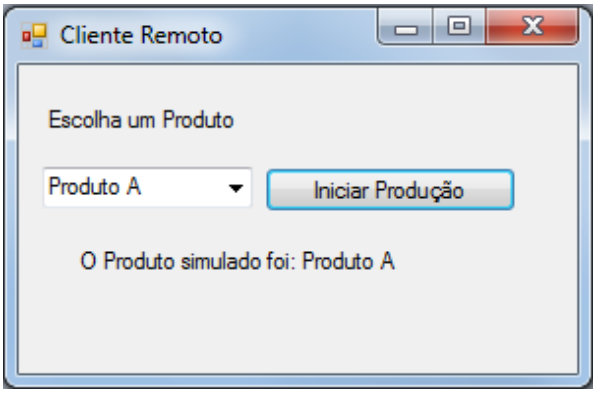


Figura 4.8 – Escolha da estação sistema de simulação local

Como é possível observar na Figura 4.9 o *output* das mensagens obtido, que resulta da interacção entre esta duas componentes, volta a ser igual ao caso em que se realizou a simulação individual na ferramenta de simulação, confirmando assim, que o sistema integrado é válido.

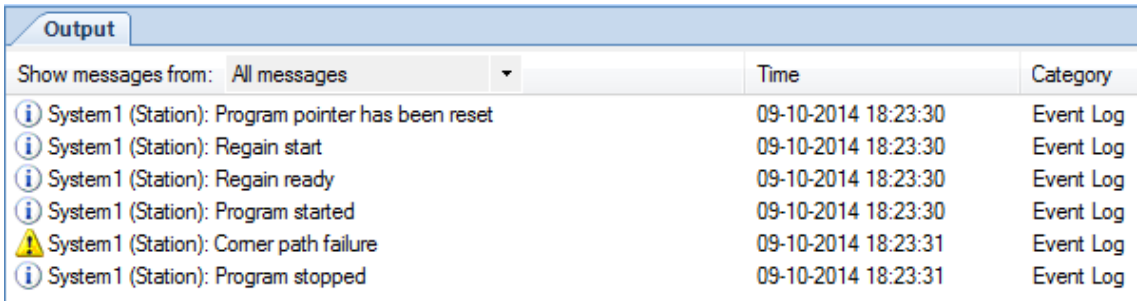


Figura 4.9 Output de mensagens no RobotStudio controlado remotamente por um cliente

Para a implementação futura com outras ferramentas é importante poder utilizar este serviço a partir de outros meios que não uma aplicação. Por esse motivo o serviço foi ajustado e pode ser utlizado a partir do *browser*. A Figura 4.10 mostra a utilização desse mesmo serviço. Neste caso, e para efeitos meramente ilustrativos, o servidor devolve o dobro do valor que o cliente lhe enviou.

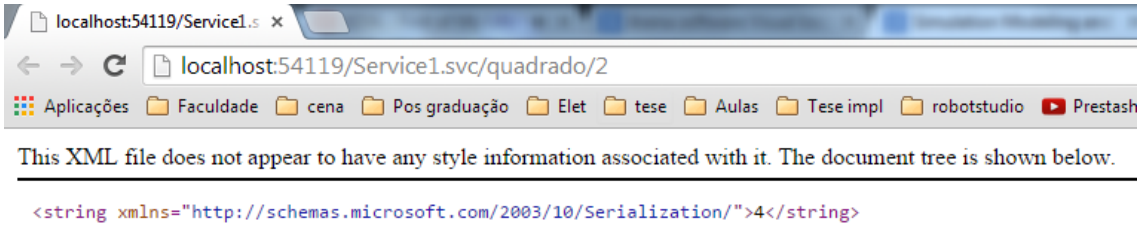


Figura 4.10 – Serviço WCF da função dobro

4.4 Arena e WCF

Para a simulação dos processos a um nível mais estratégico foi utilizado o *software* de simulação Arena. O Arena, como foi dito anteriormente, permite a utilização de bibliotecas VBA[11], que foram utilizadas para fazer a integração deste programa com outros sistemas. Como teste, o programa criado utiliza o serviço *web* ilustrado na Figura 4.10, onde se obtém o dobro do valor da variável que damos de entrada. A Figura 4.11 ilustra a implementação realizada.

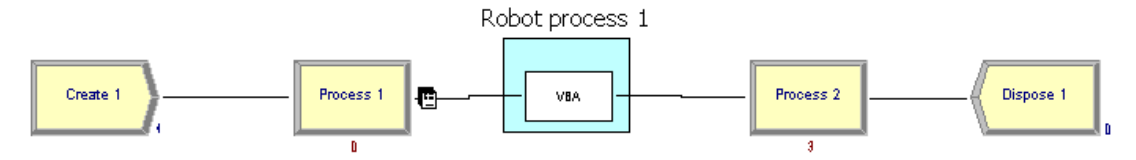


Figura 4.11 – Exemplo da simulação de processos no Arena

É possível verificar que o processo *Robot process 1* é diferente dos restantes. Nesse processo, quando chega um pacote este é processado numa lógica de programação diferente dos restantes. É accionado um programa em VBA onde é enviado um valor ao servidor. Como resposta o servidor irá retornar o dobro do valor que recebeu. Na Figura 4.12 podemos verificar que o valor recebido é o dobro do valor enviado. Deste modo, garante-se a integração individual entre estes sistemas.

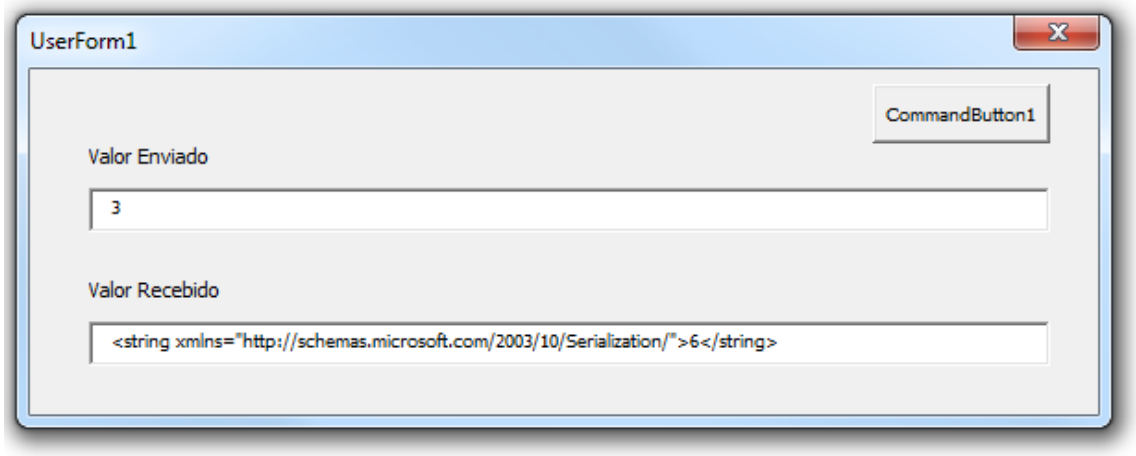


Figura 4.12 – Exemplo da troca de mensagens a partir do Arena

[11] Visual Basic for Applications

O código em VBA utilizado para na implementação deste sistema pode ser consultado em anexo. Este mecanismo de interoperabilidade entre o Arena e serviços *web* é agora utilizado na interacção com os nós de simulação locais, conforme descrito na secção seguinte.

4.5 Integração completa do Sistema

Como foi comprovado anteriormente, as diferentes partes foram testadas de forma individual apresentando sempre um desempenho positivo sobre o resultado espectável. Para chegar à solução proposta é necessário integrar as diferentes fases num só sistema e aplicá-lo a um caso típico de simulação.

Decidiu-se então utilizar um caso fabril onde são produzidas peças com uma probabilidade de defeito. Ao longo do seu processo de manufactura, as matérias-primas chegam à linha de montagem e são trabalhadas em máquinas locais para serem produzidas. Depois de criadas, são sujeitas a uma avaliação sobre a sua qualidade. Se apresentarem a qualidade desejada a peça em causa não sofrerá mais alterações sendo enviada para uma fase seguinte. Caso apresente defeito a peça é sujeita a uma correcção. Se o defeito for corrigido, então a peça volta à linha de montagem para o processo se repetir, caso contrário a peça é excluída. A Figura 4.13 mostra a implementação deste sistema no software Arena. É possível visualizar pacotes diferentes na linha de produção. Os quadrados a preto simbolizam pacotes virgens acabados de chegar à linha de produção e as bolas a vermelho são pacotes que apresentaram defeitos de fabrico e foram corrigidos.

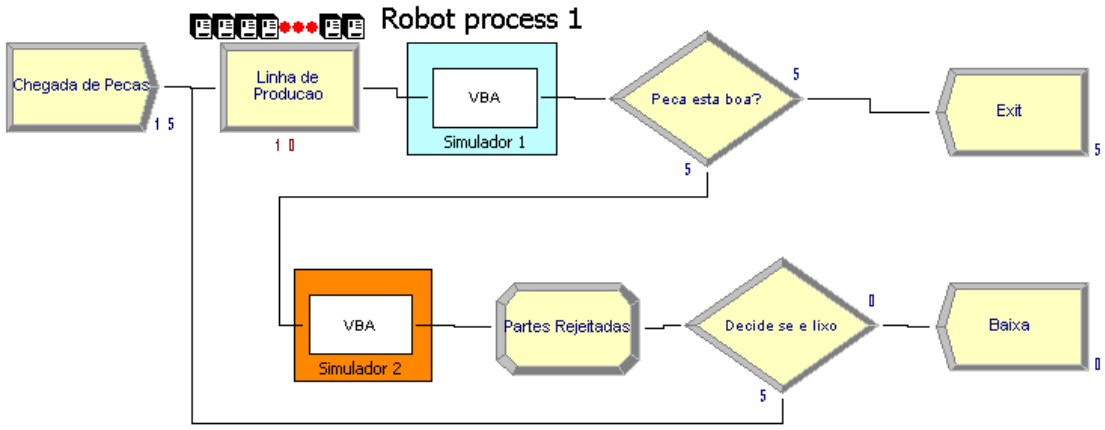


Figura 4.13 – Simulação integral no Arena

Os blocos simulador 1 e simulador 2 utilizam processos externos controlados remotamente a partir do serviço *web* implementado. Nos nós locais existem vários simuladores a utilizar a ferramenta Robotstudio ligados na mesma rede de trabalho. Dependendo do pedido que é envia-

do pelo cliente, o nó local decide qual a estação que deve simular. Quando terminada uma mensagem é enviada de volta para o cliente informando-o que a simulação ao nível local terminou. Os servidores podem estar local ou geograficamente distribuído, necessitando apenas de estar conectados à *World Wide Web.* As figuras seguintes demonstram os resultados finais obtidos em cada parte da simulação integral do sistema.

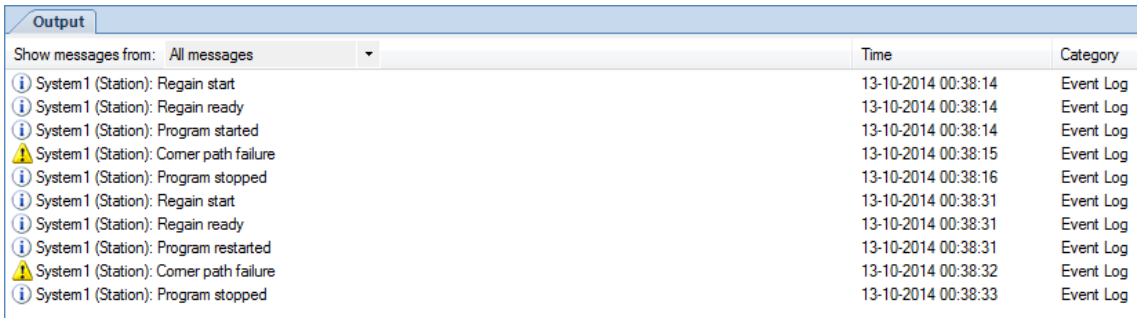


Figura 4.14 – Output de mensagens na ferramenta RobotStudio

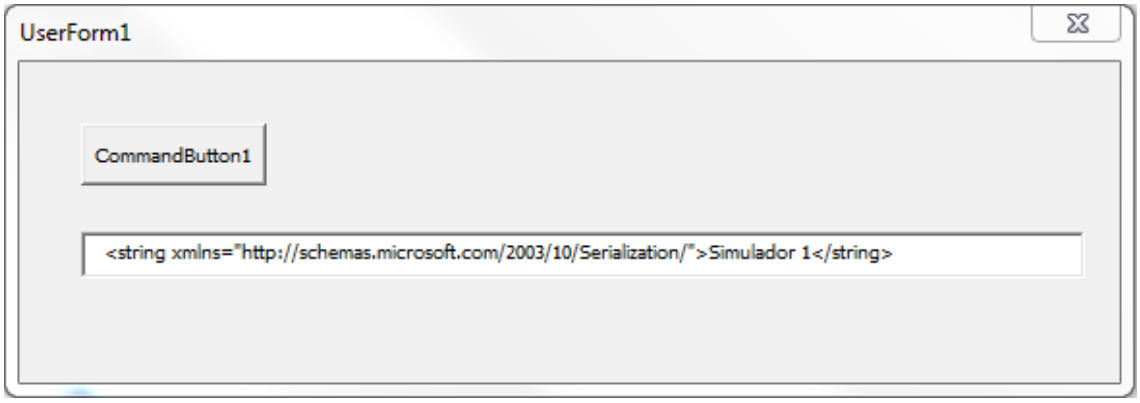


Figura 4.15 – Mensagem recebida no Arena após ordem de simulação remota

Comparando estes resultados com as simulações individuais realizadas é possível verificar que a integração das diferentes partes foi atingida, obtendo-se assim o sistema distribuído e cooperativo pretendido. O código criado para o “Simulador 1” e “Simulador 2” pode ser consultado em anexo (Anexo 3).

5 Conclusões

5.1 Síntese do trabalho efectuado

Conforme foi observado ao longo deste trabalho, num sistema distribuído existem processos que ocorrem em simultâneo e em distintos níveis. A ideia de conseguir simular tais sistemas de uma forma que se aproxime da realidade é por diversos motivos vantajosa. No entanto, como foi possível verificar, as ferramentas de simulação actuais não conseguem fornecer este tipo de serviço com a qualidade desejada para os diferentes níveis. Se realizamos a simulação de um nível estratégico de uma cadeia de produção, o que acontece realmente em cada no local pode ser simulado apenas de forma abstracta, isto é, o processo é uma estimativa de tempo ou probabilidade de um acontecimento, perdendo-se deste modo a qualidade da simulação que acontece a nível local, e de todas características que o sistema real local apresenta.

Surgiu assim a possibilidade de criação de um nível macro ou estratégico que utilize os dados de simuladores locais, num nível mais microscópico e aproximados da realidade, retirando a simulação local de um nível hipotético. Este trabalho focou-se então nas necessidades de integração entre os sistemas de simulação já existentes dos diferentes níveis, fornecendo uma interface capaz de comunicar remotamente com ambas as partes que se pretende simular.

Sendo que o principal problema encontrado consistiu nas distintas características entre os simuladores de cada nível, considerou-se importante elaborar um sistema que permitisse que pacotes de simulação distintos pudessem ser combinados e interoperáveis num ambiente heterogéneo. Este sistema seria então capaz de controlar as acções de cada simulador remotamente. A partir de troca de mensagens entre o nó central e os nós de simulação possibilitou-se a cooperação e distribuição de serviços entre eles.

A abordagem seguida possibilitou desenvolver uma ferramenta de simulação que permite simular problemas no âmbito dos SDM que consideram os processos que decorrem nos vários níveis estratégicos, nomeadamente, os níveis macro, meso e micro. Uma vez que os nós de simulação se encontravam distribuídos pela rede, existiram dois problemas que foram necessários colmatar para satisfazer as necessidades do sistema: (1) Uma forma de controlar remotamente uma ferramenta de simulação e (2) a troca de mensagens entre computadores. Foram utilizadas três ferramentas para fazer o sistema satisfazer as necessidades da hipótese:

- PC SDK – Faz o controlo remoto local. É a ferramenta utilizada para integrar o RobotStudio e o Visual Studio. A partir das bibliotecas fornecidas pela ABB, a comunicação entre a estação de trabalho, programada em RAPID, e um controlador externo é possível, e o controlo remoto acontece.
- WCF – possibilitou a troca de mensagens entre computadores a partir de uma arquitectura servidor/cliente, onde o servidor aguarda por solicitações vindas por parte do cliente.
- VBA – possibilitou aceder a um serviço *web* criado, que a partir do seu URL foi possível enviar mensagens aos nós locais distribuídos.

Para chegar à solução proposta, foi necessário, a realização de um número de passos que, quando integrados, implementam um modelo de simulação proposto. As componentes da ferramenta implementada foram estudadas e trabalhadas individualmente, para comprovar que serviam os propósitos da sua escolha e posteriormente foram integradas. As componentes dividem-se assim:

- *Engine*: que gere todos os processos do sistema. Serve de interface com o utilizador e comunica os pedidos ao restante sistema. A informação sobre as capacidades do sistema é guardada pelo *engine* para que o utilizador consiga seleccionar pedidos específicos.
- Nós de simulação: encontram implementadas as soluções para o *engine*. Eles produzem na ferramenta RobotStudio simulações de estações de trabalho onde a execução dos processos é controlada remotamente por uma aplicação criada na ferramenta Visual Studio.
- Arena: simulação dos níveis estratégicos, ou macros, dum SDM. Um processo dum nível estratégico pode estar ligado a processos do nível micro, conforme descrito nos capítulos anteriores.

A partir da implementação e validação podemos observar que a hipótese satisfaz o problema identificado tanto aos níveis locais como aos níveis estratégicos.

5.2 Resultados obtidos

Ao longo deste trabalho obtiveram-se os seguintes resultados:

- Desenvolvimento duma plataforma para simulação de sistemas de manufactura distribuída;

- Esta plataforma possui nós locais que conseguem simular processos de manufactura nesses mesmos nós;
- Utilizou-se a linguagem RAPID para especificar os processos que ocorrem nos níveis mais globais;
- Desenvolvimento do mecanismo de interoperabilidade baseado no WCF, que permite que pacotes de simulação heterogéneos possam interagir e participar na simulação de problemas de simulação numa perspectiva mais abrangente que inclua os diversos níveis dum SDM. No caso em concreto foi criado um caso real de produção de um produto no Arena, onde alguns processos são realizados nos nós locais que executam modelos de simulação dentro do RobotStudio.

Feita a verificação e validação da ferramenta desenvolvida, constatou-se que os resultados obtidos permitem concluir que os requisitos funcionais estão satisfeitos. Perante estes resultados, conclui-se que a hipótese de pesquisa formulada foi verificada. Com a verificação da hipótese, podemos também concluir que a questão de pesquisa colocada (no capítulo 1) ficou satisfatoriamente respondida.

5.3 Trabalho Futuro

Como trabalho futuro, seria interessante continuar a desenvolver a ferramenta de forma que esta consiga integrar qualquer pacote de simulação que possua funcionalidades básicas de interoperabilidade. Esta integração deverá ser feita de forma transparente e com o mínimo esforço. Para isso, seria apenas necessário que cada novo pacote de simulação utilizasse um conjunto padrão de serviços *web*, a pré-estabelecer na ferramenta de simulação desenvolvida, que permita:

- Adição de novos pacotes de simulação;
- Adição de novos nós de simulação;
- Troca de mensagens entre os diversos pacotes.

Mercê dos novos paradigmas de arquitectura distribuídas, actualmente em voga, poderia ser interessante transportar a ferramenta para um ambiente *Cloud*, para assim se conseguir lidar com os aspectos de escalabilidade que ocorrem quando se pretendem simular modelos para SDM complexos ou de grandes dimensões.

6 Bibliografia

ABB. (2012). Application manual - PC SDK (Robotics, Trans.) RobotWare 5.14.

Bagrodia, R. L. (1996). Perils and pitfalls of parallel discrete-event simulation. Paper presented at the Proceedings of the 28th conference on Winter simulation.

Camarinha-Matos, L. M., Afsarmanesh, H., Galeano, N., & Molina, A. (2009). Collaborative networked organizations–Concepts and practice in manufacturing enterprises. Computers & Industrial Engineering, 57(1), 46-60.

Carnegie Mellon University, R. I. (2010). from http://www.ri.cmu.edu/research_guide/manufacturing.html

Chituc, C.-M., & Restivo, F. J. (2009). Challenges and trends in distributed manufacturing systems: are wise engineering systems the ultimate answer. Paper presented at the Second International Symposium on Engineering Systems MIT, Cambridge, Massachusetts.

Cibraro, P., Claeys, K., Cozzolino, F., & Grabner, J. (2010). Professional WCF 4: Windows Communication Foundation with. NET 4: John Wiley & Sons.

Conference, W. S. (2014). from http://www.wintersim.org/2014/pastconf.html

DE NEGREIROS, Â. L. V. (2014). Desenvolvimento e Avaliação de Simulação Distribuída para Projeto de Sistemas Embarcados com Ptolemy.

Diaz, R., & Behr, J. G. (2010). DISCRETE-EVENT SIMULATION. MODELING AND SIMULATION FUNDAMENTALS, 57.

dos Santos Lopes, H. (2008). MODELAGEM E SIMULAÇÃO COMO FERRAMENTAS AO DIAGNÓSTICO OPERACIONAL DE SISTEMAS: ESTUDO APLICADO AO TRANSPORTE DE MINÉRIO DE FERRO NA HIDROVIA DO ARAGUAIA-TOCANTINS. Universidade Federal do Ceará.

DPN, D. P. N. (2014).). Retrieved from http://www.dpncanada.com/Factory-Automation/News/Automated-assembly-lines-from-KUKA-Systems-outfit-Canadas-largest-solar-panel-plant.html

Groover, M. P. (2007). Automation, production systems, and computer-integrated manufacturing: Prentice Hall Press.

Harrell, C., Ghosh, B. K., & Bowden, R. (2000). Simulation using promodel: McGraw-Hill.

Kelton, W. D., Sadowski, R. P., & Sadowski, D. A. (2002). Simulation with ARENA (Vol. 3): McGraw-Hill New York.

KUKA. (2014). from http://www.kuka-systems.com/NR/exeres/4A50C9C8-4549-4918-92EE-08A5F0AB03C7

Law, A. M., & McComas, M. G. (1999). Simulation of manufacturing systems. Paper presented at the Proceedings of the 19th conference on Winter simulation.

LLC, A. B. S. (2011). from http://www.advanced-business-systems.com/Solutions/ERPAccounting/FunctionalAreas/MultiSiteInternational.aspx

Marques, P. J. (2007). Simulação de um Sistema Automático de Logística Interna para a Indústria de Calçado. Tese de mestrado. Faculdade de Engenharia da Universidade do Porto.

Materials, A. (2013). Automod. from http://www.appliedmaterials.com/global-services/automation-software/automod

McLean, C., & Riddick, F. (2000). Simulation in the international IMS MISSION project: the IMS MISSION architecture for distributed manufacturing simulation. Paper presented at the Proceedings of the 32nd conference on Winter simulation.

McLean, C., Riddick, F., & Lee, Y. T. (2005). An architecture and interfaces for distributed manufacturing simulation. Simulation, 81(1), 15-32.

Nylund, H., & Andersson, P. H. (2010). Simulation of service-oriented and distributed manufacturing systems. Robotics and Computer-Integrated Manufacturing, 26(6), 622-628.

PCSTATS. (2014). from http://www.pcstats.com/articleview.cfm

Pires, J. N. (2005). Robot-by-voice: experiments on commanding an industrial robot using the human voice. Industrial Robot: An International Journal, 32(6), 505-511.

Pressman, R. S. (2011). Engenharia de software: McGraw Hill Brasil.

Research, D. W. (2011). from http://www.cwrl.utexas.edu/~bump/603A11/w/Vulture/website/7/7coetzee2.htm

Saad, S. M., Perera, T., & Wickramarachchi, R. (2003). Simulation of distributed manufacturing enterprises: a new approach. Paper presented at the Simulation Conference, 2003. Proceedings of the 2003 Winter.

SOFTWARE, C. (2014). from http://www.cimx.com/industries/enterprise-solutions

Sommerville, I., Melnikoff, S. S. S., Arakaki, R., & de Andrade Barbosa, E. (2003). Engenharia de software (Vol. 6): Addison Wesley.

systems, M. e. (2014). from http://www.mwes.com/automation/assembly-line-automation.html

URASTUN, M. I. P. L. (2003). from http://www.urastun.com/assemblies.html

Uygun, Ö., Öztemel, E., & Kubat, C. (2009). Scenario based distributed manufacturing simulation using HLA technologies. Information Sciences, 179(10), 1533-1541.

Anexo 1

Exemplo de Código RAPID – Implementação do sistema "System 1".

```
MODULE Module1
	CONST robtarget Target_10:=[[-
296.381396819,229.139497184,0],[0.009290827,0.052690896,-
0.998209472,0.026743233],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
	CONST robtarget Target_30:=[[-
186.786670257,100.559113883,0],[0.009290827,0.052690896,-
0.998209472,0.026743233],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
	CONST robtarget Target_50:=[[-203.664599944,-
95.436400033,0],[0.009290827,0.052690896,-0.998209472,0.026743233],[-1,-1,-
1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
	CONST robtarget Target_70:=[[-277.239795257,-
186.576658822,0],[0.009290827,0.052690896,-0.998209472,0.026743233],[-1,-1,-
1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
PROC Path_10()
	MoveL Target_10,v1000,z100,tool0\WObj:=Workobject_1;
	MoveL Target_30,v1000,z100,tool0\WObj:=Workobject_1;
	MoveL Target_50,v1000,z100,tool0\WObj:=Workobject_1;
	MoveL Target_70,v1000,z100,tool0\WObj:=Workobject_1;
ENDPROC
PROC main()
	Path_10;
ENDPROC

ENDMODULE
```

```
MODULE CalibData
	PERS tooldata
Pen_TCP:=[TRUE,[[110,0,140],[0.707106781,0,0.707106781,0]],[1,[10.7435139,0,140
],[1,0,0,0],0,0,0]];
	TASK PERS wobjdata Workob-
ject_1:=[FALSE,TRUE,"",[[868.044222761,0,0],[1,0,0,0]],[[0,0,0],[1,0,0,0]]];
ENDMODULE
```

Anexo 2

Classe RobotStudio – Código da Implementação do controlador remoto em linguagem C#.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using ABB.Robotics;
using ABB.Robotics.Controllers;
using ABB.Robotics.Controllers.Discovery;
using ABB.Robotics.Controllers.RapidDomain;
using System.Collections;

namespace Robo_classe
{
    class Robotstudio
    {
            scanner = new NetworkScanner();
            this.scanner.Scan();
            ControllerInfoCollection controllers = scanner.Controllers;
            x = Convert.ToInt32(valor);
            foreach (ControllerInfo controllerInfo in controllers)
            {
                lista.Add(controllerInfo.IPAddress.ToString());
                lista.Add(controllerInfo.Availability.ToString());
                lista.Add(controllerInfo.IsVirtual.ToString());
                lista.Add(controllerInfo.SystemName);
                lista.Add(controllerInfo.Version.ToString());
                lista.Add(controllerInfo.ControllerName);
                this.controller = ControllerFactory.CreateFrom(controllerInfo);
                this.controller.Logon(UserInfo.DefaultUser);

                system_name = controllerInfo.SystemName;

                if (system_name == "System1")
                {
                    StartProduction();
                }
                if ( system_name == "SystemAW2")
                {
                    StartProduction();
                }
            }
            return system_name;
            }

private void StartProduction()
        {

            try
```

```
            {
                if (controller.OperatingMode == ControllerOperatingMode.Auto)
                {
                    tasks = controller.Rapid.GetTasks();
                    using (Mastership m = Mastership.Request(controller.Rapid))
                    {
                        tasks[0].Start();
                        Console.WriteLine(DateTime.Now.Second);
                        tempo1 = DateTime.Now.Second;
                        while (controller.Rapid.ExecutionStatus !=
ABB.Robotics.Controllers.RapidDomain.ExecutionStatus.Stopped)
                        {
                            System.Threading.Thread.Sleep(1000);
                            tempo2 = DateTime.Now.Second;
                        }
                    }
                }
                else
                {
                    Console.WriteLine("Automatic mode is required to start execu-
tion from a remote client.");
                }
            }
            catch (System.InvalidOperationException ex)
            {
                Console.WriteLine("Mastership is held by another client. " +
ex.Message);
            }
            catch (System.Exception ex)
            {
                Console.WriteLine("Unexpected error occurred: " + ex.Message);
            }
        }


        private void loop()
        {

            Console.WriteLine("Fim da simulação do controlador: " + system_name);
            Console.WriteLine(tempo1);
            Console.WriteLine(tempo2);
            Console.WriteLine(tempo2 - tempo1);

            // Mandar mensagem para o cliente
        }


    }
}
```

Anexo 3

Código em VBA do bloco Simulador 1, utilizado na integração do arena com o serviço web.

Arena objects:

```
Private Sub ModelLogic_DocumentOpen()

End Sub


Private Sub ModelLogic_RunBegin()

  Module1.aaa = 1

  UserForm1.Show

End Sub


Private Sub ModelLogic_RunEnd()

  UserForm1.Hide

End Sub


Private Sub VBA_Block_3_Fire()

Module1.aaa = Module1.aaa + 2

   UserForm1.TextBox1.Text =
Moule1.httpget("http://localhost:54119/Service1.svc/quadrado/" & aaa)

End Sub
```

```
Private Sub VBA_Block_4_Fire()

Module1.aaa = Module1.aaa + 1

UserForm1.TextBox1.Text =
Moule1.httpget("http://localhost:54119/Service1.svc/quadrado/" & aaa)

Module1.aaa = Module1.aaa + 1

End Sub
```

Modules

```
Function httpget(sURL) As String

Dim obj As Object

Set obj = CreateObject("Microsoft.XMLHTTP")

obj.Open "GET", sURL, False

obj.send

httpget = obj.responseText

End Function
```