

Towards an agent-based manufacturing control strategy: a simulation study

Guilherme Cordeiro Lopes

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. João Carlos Prata dos Reis
Prof. Susana Margarida da Silva Vieira

Examination Committee

Chairperson: Prof. Paulo Jorge Coelho Ramalho Oliveira
Supervisor: Prof. João Carlos Prata dos Reis
Members of the Committee: Prof. João Manuel Gouveia de Figueiredo
Prof. Carlos Baptista Carneira

November 2018

Aos meus pais,
aos meus avós.

Acknowledgments

To Professor João Reis and Susana Vieira, for all the guidance, support and availability throughout the thesis.

To my family for always supporting and encouraging me to go further. A special thank you to my parents and grandparents, to whom I dedicate this thesis, for all they have given me so far. They are the base support of my life.

To Daniela, all these years always by my side, giving me the strength to overcome the hardest moments and the joy of sharing the happiest ones.

To all my friends who helped turning this journey into the best years of my life.

Resumo

Com o advento de tecnologias emergentes como a Internet das Coisas, redes de sensores sem fios, computação em nuvem, Internet móvel e Inteligência Artificial, e a sua introdução no mundo da indústria, um novo paradigma chegou: a quarta revolução industrial. Com o objectivo de usar estas novas tecnologias para alcançar a fábrica inteligente, flexível e reconfigurável do futuro, capaz de produzir produtos customizados e em pequenos lotes de forma eficiente e rentável, alguns sistemas avançados de controlo de manufactura têm sido propostos. Nesta tese, um sistema de manufactura baseado em agentes computacionais inteligentes foi modelado e testado em simulação com o objectivo de estudar a sua aplicabilidade para sistema de fabrico da fábrica do futuro.

Um sistema de manufactura flexível foi projetado, com o funcionamento de acordo com um conhecido sistema de referência, e um sistema multi-agente composto por 5 tipos de agentes reativos foi modelado para controlar o seu funcionamento. Os agentes foram modelados usando redes de Petri e a comunicação entre agentes foi definida através da combinação de Protocolos de Comunicação FIPA.

O sistema foi simulado em cenários estáticos e dinâmicos, tendo a sua performance sido validada sempre que possível por comparação com resultados de outras abordagens no mesmo sistema de referência. O comportamento reativo exibido pelo sistema é comparável com o das outras abordagens, tendo apresentado uma performance melhor que outro sistema para um dos cenários estáticos e uma performance semelhante para o outro. O sistema respondeu com sucesso a todas as perturbações dinâmicas simuladas.

Testes experimentais foram realizados para começar a revelar o caminho no que toca à integração de hardware e implementação de agentes num sistema de produção real.

Palavras-chave: Sistemas baseados em agentes, Controlo de manufactura, Sistemas Multi-agente, Estudo de simulação, Comunicação entre agentes, Sistema de manufactura flexível.

Abstract

With the advent of emerging technologies like Big Data, Internet of Things, wireless sensor networks, cloud computing, mobile Internet and Artificial Intelligence, and their introduction to the manufacturing environment, a new paradigm arrived to the industry world: the fourth industrial revolution. With the aim of taking advantage of the new technologies to achieve the smart, flexible and reconfigurable factory of the future, capable of producing customized and small-lot products efficiently and profitably, some advanced manufacturing control systems have been proposed. In this thesis, a manufacturing system based on intelligent computational agents was designed and tested through simulation with a view to study its applicability as a production system for the factory of the future.

A flexible manufacturing system was designed, and its operation was modelled according to a known benchmark. A Multi-Agent System composed of 5 types of reactive agents was designed to control its operation. The agents were modelled using Petri nets and agent communications were defined through the combination of FIPA Interaction Protocols.

The system was simulated under the conditions of static and dynamic scenarios, having its performance validated whenever possible by comparison with results from other approaches in the same benchmark. The reactive behaviour performance exhibited by the system was comparable with other approaches, having presented a better performance than other system for one static scenario and a similar performance for the other. The system successfully responded to all dynamic perturbations simulated.

Experimental tests were performed to start disclosing the way in terms of hardware integration and agent implementation in a real production system.

Keywords: Agent-based systems, Manufacturing control, Multi-Agent Systems, Simulation study, Agent communication, Flexible manufacturing system.

Contents

Acknowledgments	v
Resumo	vii
Abstract	ix
List of Tables	xiii
List of Figures	xv
Nomenclature	xvii
1 Introduction	1
1.1 Multi-Agent Systems in Manufacturing	3
1.1.1 Protocols and Standards	4
1.1.2 Platforms, Frameworks and Simulators	5
1.1.3 Approaches and implementations	6
1.2 Contributions	9
1.3 Thesis Outline	10
2 Manufacturing System	11
2.1 Benchmarking	11
2.2 AIP-PRIMECA Flexible Manufacturing System	13
2.2.1 Products	13
2.2.2 Machines	14
2.2.3 Conveyor System	15
2.2.4 Operation	15
2.3 Proposed IAL Manufacturing System	17
3 Multi-Agent System Architecture	20
3.1 FIPA Specifications	20
3.2 Agents	24
3.3 Agent Communication	29
3.4 Decision Rules	31
4 Simulation	32
4.1 Simulation Layout	32

4.2	Static Scenarios	36
4.3	Dynamic Scenarios	36
5	Experimental Implementation	38
5.1	Hardware integration	38
5.2	Agent Development Platform	42
5.3	Contract Net Protocol Example	44
6	Results and Discussion	48
6.1	Static Simulation	48
6.1.1	Static Scenario C0	48
6.1.2	Static Scenario B0	51
6.1.3	Static Scenario with loading order from MILP	53
6.2	Dynamic Simulation	54
6.2.1	Dynamic Scenario #PS5	55
6.2.2	Dynamic Scenario #PS7	56
6.2.3	Dynamic Scenario #PS9	57
6.2.4	Dynamic Scenario #PS10	59
6.3	Experimental implementations	60
6.3.1	Connection to PLC	60
6.3.2	JADE Example	62
7	Conclusions	64
7.1	Future Work	66
	Bibliography	67
A	Source Code	73
B	MILP formulation	79

List of Tables

1.1	Comparison of selected Platforms, Frameworks and Simulators.	6
2.1	Scenario, description and related thesis/papers of each possible benchmark.	12
2.2	Production sequence for each type of job.	14
2.3	Manufacturing operations processing times.	15
2.4	Analogy between the AIP-PRIMECA FMS and the IAL Manufacturing System.	18
2.5	Transportation times between machines (in seconds).	19
3.1	FIPA ACL Message Elements.	22
3.2	FIPA ACL Communicative Acts.	22
3.3	Communication protocols.	29
4.1	Agents representation in the simulation.	33
4.2	Static Scenarios data.	36
4.3	Dynamic Scenarios data.	37
5.1	Interface Technologies.	40
5.2	PLC variables.	41
6.1	Workstation utilization.	50
6.2	Makespan for several control approaches under B0 conditions.	52
6.3	Summary of all the simulations performed.	63

List of Figures

1.1	Application of agents in manufacturing control.	2
1.2	Model of a physical holon.	4
2.1	AIP-PRIMECA cell.	13
2.2	Components, jobs and products.	14
2.3	AIP-PRIMECA cell layout.	16
2.4	Industrial Automation Laboratory (IAL).	17
2.5	IAL Manufacturing System layout.	19
3.1	Agent Management Reference Model.	21
3.2	FIPA Contract Net Interaction Protocol.	23
3.3	FIPA Propose Interaction Protocol.	23
3.4	FIPA Request Interaction Protocol.	24
3.5	Behavioural model of the Order Agent.	25
3.6	Behavioural model of the Job Agent.	26
3.7	Behavioural model of the Workstation Agent.	27
3.8	Behavioural model of the Robot Agent.	28
3.9	Behavioural model of the Conveyor Agent.	28
3.10	Interaction diagram for the Order Agents negotiation.	30
3.11	Interaction diagram for the example of operation "Axis mouting".	30
4.1	IAL Manufacturing System simulation environment.	34
4.2	IAL Manufacturing System model in GAMA.	34
4.3	Statistics display in GAMA.	35
5.1	Interface between a simulation agent and a PLC.	39
5.2	The JADE Architecture.	43
5.3	JADE Management GUI (a) and JADE Sniffer Agent GUI (b).	44
5.4	Exchanged messages between agents in different computers.	45
5.5	Exchanged messages with Job Agent "B".	46
5.6	Exchanged messages with Job Agent "B" and "E".	47
6.1	Gantt chart for Static Scenario C0 with rule base 1.	49

6.2	Gantt chart for Static Scenario C0 with rule base 2.	49
6.3	Gantt chart for Static Scenario C0 from benchmark.	51
6.4	Gantt chart for Static Scenario B0.	52
6.5	Gantt chart for Static Scenario C0 from MILP.	53
6.6	Gantt chart for Static Scenario C0 with loading order from MILP.	54
6.7	Gantt chart for Dynamic Scenario #PS5.	55
6.8	Gantt chart for Dynamic Scenario #PS7.	56
6.9	Gantt chart for Dynamic Scenario #PS7 from benchmark.	57
6.10	Gantt chart for Dynamic Scenario #PS9.	58
6.11	Gantt chart for Dynamic Scenario #PS9 from benchmark.	59
6.12	Gantt chart for Dynamic Scenario #PS10.	60
6.13	Gantt chart for Static Scenario C0 in connection to the PLC (1).	61
6.14	Gantt chart for Static Scenario C0 in connection to the PLC (2).	62
6.15	Gantt chart for the CNP example in JADE.	63

Nomenclature

ABAS Actor-Based Assembly Systems

ACC Agent Communication Channel

ACL Agent Communication Language

ACS Autonomous Cooperative System

ADACOR ADaptive holonic COntrol aRchitecture

AIP-PRIMECA Atelier Inter-Etablissements de Productique et Pôle de Ressources Informatiques pour la MECAnique

AI Artificial Intelligence

AMS Agent Management System

AP Agent Platform

BDI Belief-Desire-Intention

CFP Call For Proposals

CMU Cooperative Manufacturing Unit

CNC Computer Numerical Control

CNP Contract Net Protocol

CoBASA Coalition Based Approach for Shop Floor Agility

CPS Cyber-Physical Systems

DF Directory Facilitator

EU European Union

FIPA The Foundation for Intelligent Physical Agents

FJSP Flexible Job-Shop Scheduling Problem

FMS Flexible Manufacturing System

GAML GAMA Agent Modelling Language

GRACE InteGration of pRocess and quAlity Control using multiagEnt technology

GUI Graphical User Interface

HMS Holonic Manufacturing System

IAL Industrial Automation Laboratory

IDIT Instituto de Desenvolvimento e Inovação Tecnológica

IEC International Electrotechnical Commission

IEEE Institute of Electrical and Electronics Engineers

ILP Integer Linear Programming

IoT Internet of Things

IPC Industrial Personal Computer

IP Interaction Protocol

JADE JAVA Agent DEvelopment Framework

JVM Java Virtual Machine

MASCADA Manufacturing Control Systems Capable of Managing Production Change and Disturbances

MAS Multi-Agent System

MILP Mixed Integer Linear Programming

MTS Message Transport Service

OPC-UA Open Platform Communications Unified Architecture

ORCA dynamic Architecture for an Optimized and Reactive Control

PABADIS Plant Automation BAsed on DIstributed Systems

PERFoRM Production harmonizEd Reconfiguration of Flexible Robots and Machinery

PID Proportional-Integral-Derivative

PLC Programmable Logic Controller

PROSA Product-Resource-Order-Staff Architecture

UML Unified Modelling Language

UNINOVA Instituto de Desenvolvimento de Novas Tecnologias

WIP Work-In-Progress

Chapter 1

Introduction

Over the times, the industry has been evolving to meet the progressive improvement of life quality that the human society desires. With the aim of providing products with increasingly high-quality and customization, industry has already faced three revolutionary stages known as industrial revolutions. With the advent of emerging technologies like Internet of Things (IoT), wireless sensor networks, Big Data, cloud computing, embedded systems, mobile Internet and Artificial Intelligence (AI), and their introduction into the manufacturing environment, a new paradigm arrived to the industry world: the fourth industrial revolution [1].

As a consequence, several initiatives have been put in motion by the main industrial countries all around the world to implement this new revolutionary technologies and pursue the factory of the future. The German government started the research program called "Industrie 4.0", a similar program called "Made in Sweden 2013" was started in Sweden, the USA created the Industrial Internet consortium, China announced "Made in China 2025", and many other countries are supporting research initiatives in the field, like Spain, Canada or Korea [2].

This new industrial stage, widely known as Industry 4.0, identifies with the application of Cyber-Physical Systems (CPS) in the industrial domain, as a network of interacting software and hardware devices and systems through the combination of mechatronics and communication and information technologies. These systems will be able to analyse data and adapt to changes in a way that processes can become faster, more flexible and efficient at lower costs [3].

With the aim of achieving a smart, flexible and reconfigurable factory, capable of producing customized and small-lot products efficiently and profitably, the use of traditional manufacturing control systems is not enough. These control systems do not exhibit sufficient capabilities of responsiveness, flexibility and reconfigurability, since they are designed based on centralized and hierarchical control structures that, despite presenting good production optimization, present weak response to change due to the rigidity and centralization. Thus, some advanced manufacturing control systems have been proposed, taking advantage of the new technologies to design the smart factory of the future [4].

A very representative case of the advanced manufacturing control systems is the agent-based manufacturing control, which is the scope of this thesis. The agent-based manufacturing control systems

are based on Multi-Agent Systems (MAS) technology, and consist in an ecosystem of manufacturing resources defined as intelligent, autonomous and cooperative computational entities, known as agents, that can negotiate with each other to implement dynamical reconfiguration and decision-making, in order to achieve their individual goals. In an agent-based manufacturing control system, all the agents are in the same hierarchy level, being organized in a autonomous, distributed and decentralized architecture [4].

The distributed character of the behaviour of agents, with each one of them being responsible for one or very few industrial entities and having its own goals, yields a high flexibility for the system. Moreover, each agent software is simpler than any global approach and is independent, which allows an easy addition and removal of industrial entities without changing the network software. This ability to change fast and easily provides the reconfigurability capacity that motivates the use of agent-based systems in the manufacturing world.

Figure 1.1 exhibits a simplified example of the operation of a manufacturing system controlled by agents. Here, the part to be manufactured is represented by a part agent running in an industrial personal computer (IPC), three computer numerical control (CNC) machines are represented by machine agents running in the numerical controller of each machine and an automated guided vehicle is represented by a transport agent running on a programmable logic controller (PLC). In order to complete the drilling that the part needs, the part agent queries the machine agents to decide which one has the skills and availability to do the operation and chooses according to the received answers. After the negotiation, the part agent allocates the job to machine agent #3 and then negotiates with the transport agent to transport the part to the corresponding machine.

In this chapter, a literature review on the use of multi-agent systems technology in manufacturing is presented in the first section. In the following section, the objectives and contributions of the thesis will be stated. At the end of the chapter, the outline of the thesis is given.

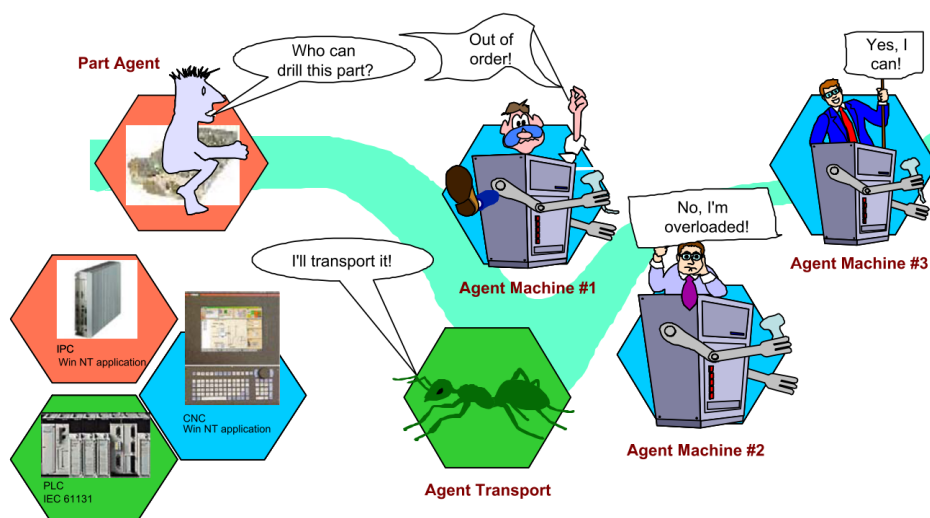


Figure 1.1: Application of agents in manufacturing control [5].

1.1 Multi-Agent Systems in Manufacturing

Throughout the last decades, several different approaches, architectures and platforms regarding MAS have been introduced and a considerable amount of industrial applications were already implemented and described in the literature. The main fields of application have been smart production, smart electric grids, smart logistics and smart healthcare [2], although some authors also have a prospect of other fields that might benefit from the application of agent technologies, namely traffic control, buildings and home automation, military and network security [6]. In this work, the focus will be on the manufacturing industry.

MAS are computational systems composed by interacting intelligent agents which can assume several forms and architectures. Agents can represent physical resources or logical objects, originating two different ways of being set up: the function mapping method, in which different agents are used to represent different functional modules in manufacturing systems, and the physical mapping method, where different agents are used to represent different real physical entities [7].

Considering that the agents need to be connected to the manufacturing entities that they are representing, the interface approach can be classified into three types [8]:

- *Coupled approach*: Software agents are running remotely and accessing the physical automation device, which is running a native logic control layer.
- *Weak embedded approach*: Software agents are directly embed in automation devices, but are accessing the native logic control layer.
- *Strong embedded approach*: Software agents are directly embed in automation devices and are running the logic control of the device, accessing input and output ports.

Furthermore, regarding their functionality and decision-making mechanism, intelligent agents are classified into the following types [9],[4]:

1. *Purely reactive*: Agents make decisions using only the present information and discarding historical data. Direct mapping from situation to action.
2. *Logic-based*: Agents make decisions through logical deduction.
3. *Belief-desire-intention (BDI)*: Agents are built using symbolic representations of their intentions, beliefs and desires and the decision-making depends on the manipulation of those.
4. *Layered architectures*: Agents combined into several software layers where agents from each layer deal with different abstract levels of the environment.

In the context of the manufacturing industry, it is important to introduce another advanced manufacturing control paradigm that has been emerging, called Holonic manufacturing control. The reason for this is the close relation to agent-based manufacturing control and the fact that it also uses agent technology in its implementation. The Holonic Manufacturing Systems (HMS) can be explained as a system of holons, organized in a hierarchical structure and cooperating to achieve the system goals by

combining their individual skills and knowledge. In the holonic paradigm, the holon is a concept that can simultaneously represent a part of a system and a whole system and is composed of two distinct parts (Figure 1.2): an information processing part, which is made of physical interfaces, control functions and inter-holon communications; and a logical or physical processing part, depending if the holon is representing a physical device or a logical activity. The high level layer of the information processing part can be implemented as an agent, using MAS technology for the inter-holon communications.

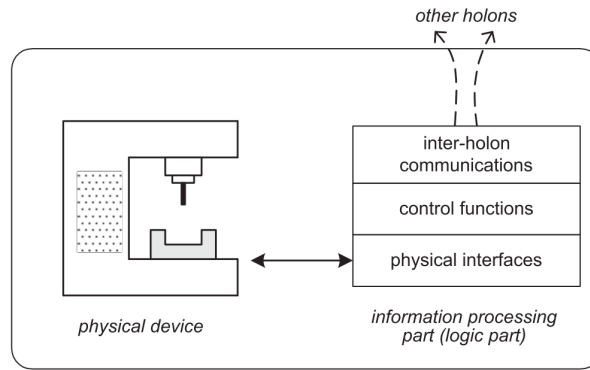


Figure 1.2: Model of a physical holon [4].

Due to the fact that these two manufacturing paradigms share the same principles of autonomy and cooperation, the similarities between agents and holons are evident. However, it is important to notice that they are different concepts with the following main distinctions: (1) agents normally represent software components while the holon concept focuses on the integration of software and physical devices; (2) the holon can be hierarchically composed by several lower-level holons, in contrast with an agent; (3) the holon is a concept while the agent is simultaneously a concept and a technology, being that the reason why the holon can be implemented with agent technology [4].

1.1.1 Protocols and Standards

With the development of agent systems, some protocols and standards were created to support and propel their adoption.

The Foundation for Intelligent Physical Agents (FIPA) organization produced a set of standards covering agent management, communication and message transport, and also established the Agent Unified Modelling Language (AUMML) to leverage UML for modelling of large-scale agent-based applications [10]. Within the agent communication, the Agent Communication Language (ACL) was defined, standardizing the messages exchanged between agents. Within the AUMML, Protocol Diagrams were introduced to specify agent interaction protocols (internal behaviour) and agent class diagrams were introduced to specify its external behaviour [11].

The Contract Net Protocol (CNP) is a high-level task sharing protocol for communication among the nodes (agents) in a distributed problem solver (MAS). The task distribution is seen as a kind of contract negotiation where nodes receive tasks, announce them to the contract net, receive bids from other nodes which are potential contractors and decide who to award the contract to [12].

Regarding the agent implementation, the IEC 61131-3 standard specifies five PLC programming languages [13]: the textual programming languages instruction list and structured text, the graphical programming languages function block diagram and ladder diagram, and the higher level programming language sequential function chart, and the IEC 61499 specifies the distributed way of designing the real-time control applications using the dedicated service interface function blocks to exchange messages between the low level (PLC) and the high level (agents) control modules [14].

1.1.2 Platforms, Frameworks and Simulators

In regard to the matter of the tools used to develop MAS, a large number of platforms, frameworks and simulators have been developed since the late 90s, targeting the gap between concept design and implementation.

An extensive survey and comparison of agent platforms can be found in [15]. In [16], a comparative analysis of 24 multi-agent platforms, frameworks and simulators is presented, evaluating the given tools in 28 universal criteria. More recently, in [17], a detailed characterization of almost the entire spectrum of agent-based modelling and simulation tools was presented, covering eighty five toolkits that go from the typical softwares used to design and develop agent-based systems to several other tools also known to be used in different fields of the engineering world. In [9], a table with 8 up-to-date and open source selected platforms, frameworks and simulators was compiled, focusing the comparison in the programming language and the operation system.

Table 1.1 presents a comparison of eight Platforms, Frameworks and Simulators, selected from the previously mentioned surveys accordingly to its current significance and possible value to this work.

In terms of frameworks, reports of applications using the JAVA Agent DEvelopment Framework (JADE) [18] have been a recurrent case in the literature, being one of the most widely used platforms for research purpose [9].

Table 1.1: Comparison of selected Platforms, Frameworks and Simulators.

Software	Type	Developer	Programming Language	Operation System(s)	Communication	Standard Compatibility
JADE	Framework	Telecom Italia	Java	All that supports JVM	ACL (Asynchronous), MTPs, RMI, IIOP, HTTP, WAP	FIPA, CORBA
Agent Factory	Framework	University College Dublin	Agent Factory Agent Programming Language (AFAPL)	All that supports JVM	HTTP	Partially FIPA (agent lifecycle)
Jadex	Framework	Hamburg University	Java (plus use of XML)	Any with JVM	HTTP	FIPA, SOA, WSDL
JACK	Platform	AOS	Java, JACK Agent Language (JAL)	Windows, Mac OS , Unix	DCI network, TCP/IP, ACL	FIPA
MASON	Simulator	George Mason University	Java	Windows	Message exchange (event-driven, platform defined syntax)	None known
GAMA	Simulator	IRD/UPMC International Research Unit UMMISCO	GAML	Windows, Linux, Mac OS	ACL (GAML messages that represent FIPA ACL messages)	FIPA, (GIS, 3D capabilities)
NetLogo	Simulator	Northwestern University	NetLogo	Any with JVM	Extensions for message exchange(extensions defined syntax)	None known
Repast	Simulator	University of Chicago	Java, C++, Prolog, Python	Any with JVM	Peer-to-Peer	None known

1.1.3 Approaches and implementations

With the focus on smart manufacturing solutions, different authors already extensively surveyed the major approaches and implementations presented so far. It is the case of Monostori, Vancza and Kumara [19], Mařík and Lažanský [20], Leitão [4], Leitão and Vrba [21], Leitão, Mařík and Vrba [6], and Leitão et al.[2]. Based on the work of the authors previously introduced, it is important to sum up the agent-based approaches of higher relevance that were presented in the last years.

In 1995, Rockwell Automation developed an agent-based solution to increase the utilization of steel milling process at the steel rod bar mill of the BHP Billiton in Melbourne, Australia, by assigning jobs to available rolling stands and cooling boxes instead of using a predefined set of equipment for a particular recipe as before. It utilizes the Autonomous Cooperative System (ACS) platform that enables to run the

C++ based agents directly on the PLC of the Logix family together with IEC 61131 control programs. Furthermore, they introduced the universal runtime interface to allow C++ control agents to access the sensor and actuator values in the data table of the ControlLogix controller. [22].

The Product-Resource-Order-Staff Architecture (PROSA) is a holonic reference architecture for manufacturing systems based on three types of basic holons: product, order and resource, plus additional staff holons [23]. The European Union (EU) MASCADA (Manufacturing Control Systems Capable of Managing Production Change and Disturbances) project had as its main test the development of a PROSA-based MAS for a section of the painting center in the Daimler-Benz plant in Sindelfingen, Germany. In this project, apart from the 3 basic PROSA agents, 4 other agents were derived from the PROSA architecture: location, switch, buffer, machine and car agent [24].

Schneider Automation developed, in 2001, a MAS application for the production of engine cylinder heads in the factory plant of Daimler Chrysler. The application uses agents to represent CNC machines, shifting tables and work-pieces, implementing a dynamic resource allocation similar to the CNP with the objective of continuous optimization of the throughput. The agent-based control system allows individual work-pieces to be directed dynamically around the production area, by auction off the processing steps that are due first [25].

In the laboratory packing cell of the University of Cambridge, an industrial agent-based control testbed for assembling Gillette packages into customer-tailored gift boxes was deployed. The approach considers resource holons that represent the physical components of the system and order holons for each gift box that must be manufactured. The dynamic resource allocation is ensured by BDI agents implemented in JACK platform [26].

Another PROSA-based MAS, entitled FABMAS, was installed for the production control of a semiconductor wafer fabrication facility. The approach considered a hierarchical organization of the agents, making the distinction between decision-making agents and staff agents as described by the PROSA architecture. The agents were designed using the FIPA Standard and the FIPA Abstract Architecture, and the MAS was implemented on C# using the Microsoft .NET Framework [27].

The PABADIS (Plant Automation BAseD on DIstributed Systems) approach, in the context of the FP5 PABADIS project, targeted a distributed, agent-based manufacturing control. This architecture, developed and demonstrated at Magdeburg University, consisted of a system of Cooperative Manufacturing Units (CMUs) and an agency that communicate to achieve decentralized control. In this system, two types of CMUs were distinguished: Manufacturing CMUs and Logical CMUs; and three types of agents were defined: Residential Agents, Product Agents and Plant Management Agents [28].

At the Tampere University of Technology in Finland, a framework called ABAS (Actor-Based Assembly Systems) was developed, aiming the 3D simulation and visualization of assembly processes carried out by autonomous mechatronic devices. These autonomous devices were represented by intelligent physical agents called actors. Its architecture is composed by the Actors, the Register, the Recruiters and the Actor Clusters. The Recruiters seek the accomplishment of assembly tasks by recruiting Actors which were admitted in the ABAS society by the Register, recognizing the ones capable of solving the required assembly needs and grouping them into clusters [29].

An example of a laboratorial application of a multi-agent control system for shop-floor assembly was implemented in the Novaflex manufacturing system at UNINOVA (Instituto de Desenvolvimento de Novas Tecnologias), Portugal [30]. The system was made of Manufacturing Resource Agents (MRA), Agent-Machine Interfaces (AMI), Broker Agents (BA) and Coalition Leader Agents (CLA), following the guidelines of the CoBASA (Coalition Based Approach for Shop Floor Agility) architecture. This architecture consists of MRAs that are joined into clusters, participate in coalition/consortia regulated by contracts and interact in order to generate aggregated functionalities that, in some cases, are more complex than the simple addition of their individual capabilities [31].

Another laboratorial application of a multi-agent manufacturing control system can be found at the IDIT (Instituto de Desenvolvimento e Inovação Tecnológica), Portugal where a real flexible manufacturing system was implemented using the ADACOR holonic architecture. The term ADACOR stands for ADaptive holonic COntrol aRchitecture and this approach considers four holon classes: product (PH), task (TH), operational (OH) and supervisor holons (SH). As far as the internal architecture of a generic ADACOR holon is concerned, it is constituted by a logic control device, organized in the three components: communication, decision and physical interface, and by a physical resource capable of performing the manufacturing tasks [32]. In the real system, the logic part of the holons (agents) was implemented as a Java class in the JADE framework, the messages were encoded using the FIPA ACL and the rule-based system in the decision component of the holons implemented using the Java Expert System Shell tool [33].

In the context of EU FP7 GRACE (InteGration of pRocess and quAlity Control using multiagEnt technology) project, a MAS for integrating process and quality control in the washing machine production line of Whirlpool in Naples, Italy, was deployed. The system is composed of 4 types of agents: product type agents (PTAs), product agents (PAs), resource agents (RAs) and independent meta agents (IMAs), which had their behaviour modelled using Petri nets and their interactions modelled using AUML. Self-adaptation procedures like the dynamic adaptation of the functional testing plan and the customization of the on-board controller parameters were designed. The deployment in the factory plant brought several improvements such as the increase in the production efficiency, the reduction of the production down-times or the reduction of nonconformities and increase in product quality [34].

An evolution of the ADACOR architecture, called ADACOR², was developed with the aim of providing a dynamic control system reconfiguration, through the introduction of a two-dimensional self-organization model which takes into consideration mechanisms at micro and macro level. At micro level, behavioural self-organization is acting towards a smooth evolution, whereas at macro level, structural self-organization makes the system evolve drastically. This system makes use of nervousness stabilizers analogous to the PID controller to stabilize the self-organization behaviour. An experimental simulation scenario was deployed in the AIP-PRIMECA (Atelier Inter-Etablissements de Productique et Pôle de Ressources Informatiques pour la MECAnique) Flexible Manufacturing System (FMS) located at the University of Valenciennes, validating the improvement in performance [35].

In the context of the PERFoRM (Production harmonizEd Reconfiguration of Flexible Robots and Machinery) project, a MAS was developed to support the seamless reconfiguration of micro-flow pro-

duction cell. In the system, two types of agents were defined: Robot agents and Process agents, which intercommunicate with the robots and the machinery via an OPC-UA (Open Platform Communications - Unified Architecture) server with the goal of managing the plugging in and out of processes in the cell and achieve on-the-fly reconfiguration. The system was deployed in a use case prototype with seven process agents and one robot agent, which validated the proposed approach [36].

Despite the afore mentioned approaches and implementations, a long path still needs to be travelled in order to have these distributed agent-based approaches fully adopted by industry. The approaches that reached full industrial implementation are still few and the implemented functionalities are normally limited. The reasons for this might be related to the fact that real industrial applications require the integration of a large number of physical automation devices with the software control system and methodologies to support an easy, transparent and re-usable hardware integration are still missing. Also the absence of industrial controllers with multi-agent systems capabilities to run the agents directly on controllers in parallel with the low-level control, rather than on a separate computer, might be part of the reason. Moreover, these systems are usually referred as performing well in dynamic environments but more evidence of that needs to be reported in the literature. The evaluation and comparison of manufacturing control systems performance requires the existence of benchmark scenarios and datasets, together with normalized performance indicators. More teaching and research in this area is needed so that these necessary tools might be developed. Real proofs about the applicability of these smart and distributed approaches in industrial automation sites are still necessary to accelerate its industrial adoption.

1.2 Contributions

The aim of this thesis is to study the applicability of agent-based manufacturing control as a production control system designed to achieve the smart, flexible and reconfigurable factory. The main focus is the successful design and simulation of a Multi-Agent System used to demonstrate a flexible manufacturing system to be implemented in the Industrial Automation Laboratory at Instituto Superior Técnico.

The study starts with the planning of the flexible production system, with operation analogous to a known benchmark for distributed control systems research, followed by the design of a multi-agent system composed of five types of FIPA compliant agents. A combination of three FIPA Interaction Protocols (FIPA-Request, FIPA-Propose and FIPA-Contract Net) was used to design the agent communication and two simple rule based systems were tested to ensure the proper decision-making in the assignment of operations in the system. A simulation study was carried to test the autonomous agent-based system through static and dynamic scenarios, aiming to validate its performance by comparison with results from the benchmark, which include an *a priori* optimization using Mixed Integer Linear Programming (MILP). Furthermore, the first steps into the implementation were taken. In terms of hardware integration, a real-time connection between a simulated agent and a real PLC was established. Regarding software, a demonstrative implementation of agents executing the Contract Net Protocol in the JADE Platform is presented.

1.3 Thesis Outline

Chapter 1 presents the scope of this thesis and a literature review on the application of MAS in the manufacturing industry.

Chapter 2 introduces the flexible manufacturing system benchmark as well as the proposed analogous production system to be the object of study in this work.

Chapter 3 addresses the modelling of the agents and the proposed communication protocols and algorithms.

Chapter 4 is dedicated to the simulation study, encompassing the explanation of the layout and the proper description of the static and dynamic scenarios.

Chapter 5 contains the experimental tests, namely the demonstrative implementation of agents and the connection to a PLC.

Chapter 6 concerns the discussion of the obtained results in the static and dynamic experiences and a comparison with results from the benchmark, whenever possible.

Chapter 7 presents the conclusions of the work carried out and the proposals for future work based on experiences started in this thesis.

Chapter 2

Manufacturing System

With a view to implementing an agent-based manufacturing control system, the manufacturing system to be controlled must be properly defined.

Considering that the aim is to design a system which is smart, flexible and reconfigurable, and is capable of producing customized and small-lot products, it is necessary to understand which manufacturing system typologies might be adequate for this purpose. From [37], it can be inferred that to produce customized and small-lot products, the more advantageous manufacturing system typology is likely to be job shop based.

To help define the manufacturing system that will be the object of this thesis, a survey through the possible benchmarks found in the literature regarding production systems for agent-related control research was performed and will be presented in the first section.

Afterwards, a benchmark system combining all the necessary characteristics and with a record of experimental agent-related implementations will be explained, followed by the definition of the proposed manufacturing system in the last section.

2.1 Benchmarking

Trentesaux et al. in [38] define benchmarking as "comparing the output of different systems for a given set of input data in order to improve the system's performance". It is of major importance to have a comparative element to validate and compare the performance of a system when designing one. This is the reason why the starting point for the design of the manufacturing system to be the object of this thesis is the search for an appropriate benchmark. From the literature review presented in the section 1.1.3, the production systems built for research purposes were selected as possible benchmarks:

- Cambridge packing cell, University of Cambridge, England.
- AIP-PRIMECA FMS, University of Valenciennes, France.
- ADACOR FMS, IDIT, Portugal.
- NovaFlex, UNINOVA, Portugal.

In Table 2.1, a description of the scenario, the layout and a list of papers and PhD thesis related to each possible benchmark are provided, so that their operation and research significance may be assessed.

Table 2.1: Scenario, description and related thesis/papers of each possible benchmark.

Name	Scenario	Layout	Papers/Thesis
Cambridge Packing Cell	Customer selects any three of four types of Gillette personal grooming items, packs them into one of two box styles and changes how their order is packed on the fly.	Three conveyor loops, two docking stations, one storage area and one robot.	[26] [39] [40] [41] [42]
AIP-PRIMECA FMS	The cell produces three types of products (words), using combinations of seven different jobs (letters), which are made of raw components. There are five types of components plus the plate to produce the specified letters.	Closed-loop conveyor System, one loading/unloading unit, one inspection area, three robots, one optional manual recovery unit and one optional robot.	[43] [44] [38] [45] [46] [35] [47] [48] [49] [50] [51]
ADACOR FMS	In the factory plant, two final products are produced by assembly of four sub-products that are manufactured in one of the manufacturing cells.	One material storage and one transportation cell, one inspection cell, one assembly cell and three flexible manufacturing cells composed of two CNC's and one robot each.	[52] [32] [33] [53]
NovaFlex	A manufacturing cell that assembles a metallic toy watch.	Two assembly robots, one automatic warehouse and one transport system that connects all the modules.	[31] [30] [54]

If the goal is to control a flexible and reconfigurable production system, there are some required characteristics concerning its physical configuration that need to be met. In the first place, the conveyor system needs to be flexible, providing more than one path to travel between the same two points. Thus, the system will provide material-handling flexibility and machine-sequence flexibility. Next, redundancy is a key point in this kind of system, being necessary to provide machine flexibility and reconfiguration of the products machine sequence in case of machine breakdown.

By analysis of the given production systems, it is clear that the AIP-PRIMECA FMS has been the most used for research purposes in the area of distributed agent-related control systems. Its conveyor system configuration allows a really flexible routing of jobs inside the production cell, and the existence of three robots, which provide some operations in common, creates the necessary redundancy for the production. Furthermore, a benchmark was defined from this production cell, aiming to support benchmarking on a physical and real-world system and stimulate benchmarking activities internationally [38].

For the previously stated reasons, the AIP-PRIMECA FMS was defined as a benchmark for this thesis, directly influencing the design and operation of the proposed manufacturing system.

In the next section, a detailed description of the AIP-PRIMECA FMS is presented, comprising the types of products and jobs, required operations for each type of job, operations processing times, transport times and operation parameters.

2.2 AIP-PRIMECA Flexible Manufacturing System

The AIP-PRIMECA FMS is a flexible assembly cell located in the AIP-PRIMECA Center at the University of Valenciennes. This assembly cell, shown in Figure 2.1, is composed of industrial components such as conveyors, industrial robots, sensors, actuators and programmable controllers, being an appropriate platform for experimental tests regarding FMS control approaches. Moreover, this cell has already been modeled as a Flexible Job Shop [38], providing the starting point for the conception of linear and quadratic programming models and already having inspired the creation of some of those models that can be used for future comparisons.



Figure 2.1: AIP-PRIMECA cell [49].

In order to explain the operation of this cell, the next four subsections describe the products, the machines, the conveyor system and the global operation of the system, according to the description provided by [38].

2.2.1 Products

The smallest elements present in the production cell are the five available components "Axis_comp", "l_comp", "L_comp", "r_comp" and "screw_comp", plus the "Plate" where they are placed.

By combination of these components, it is possible to assemble 7 different letters: "B", "E", "L", "T", "A", "I" and "P". Each letter is a sub-assembly and is denominated job.

The final products proposed to the client are words formed with these jobs and they are three: "BELT", "AIP" and "LATE". The client can submit an order with any quantity of each product. A product

is considered complete when all the jobs that compose it are finished, considering that those jobs can be manufactured in any order.

The components, jobs and products are shown in Figure 2.2.

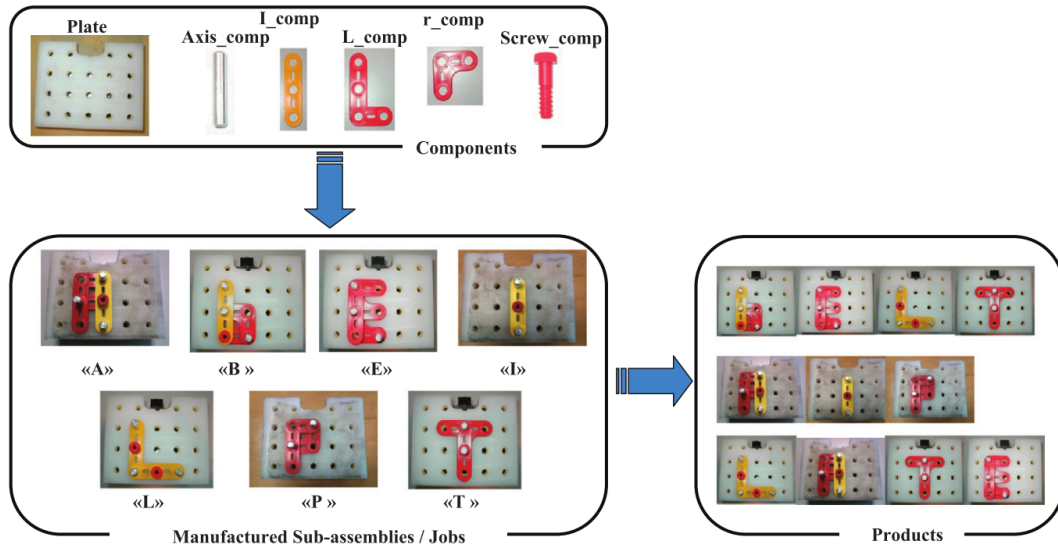


Figure 2.2: Components, jobs and products [38].

Each job has its own production sequence, i.e., an ordered list of elementary manufacturing operations. In this assembly cell, there are eight manufacturing operations: "Plate loading", "Axis mounting", "r_comp mounting", "l_comp mounting", "L_comp mounting", "Screw_comp mounting", "Inspection" and "Plate unloading". Table 2.2 shows the production sequence for each type of job.

Table 2.2: Production sequence for each type of job.

	"B"	"E"	"L"	"T"	"A"	"I"	"P"
#1	Plate loading	Plate loading	Plate loading	Plate loading	Plate loading	Plate loading	Plate loading
#2	Axis mounting	Axis mounting	Axis mounting	Axis mounting	Axis mounting	Axis mounting	Axis mounting
#3	Axis mounting	Axis mounting	Axis mounting	Axis mounting	Axis mounting	Axis mounting	Axis mounting
#4	Axis mounting	Axis mounting	Axis mounting	r_comp mounting	Axis mounting	l_comp mounting	r_comp mounting
#5	r_comp mounting	r_comp mounting	l_comp mounting	L_comp mounting	r_comp mounting	Screw_comp mounting	L_comp mounting
#6	r_comp mounting	r_comp mounting	l_comp mounting	Inspection	L_comp mounting	Inspection	Inspection
#7	l_comp mounting	L_comp mounting	Screw_comp mounting	Plate unloading	l_comp mounting	Plate unloading	Plate unloading
#8	Screw_comp mounting	Inspection	Screw_comp mounting		Screw_comp mounting		
#9	Inspection	Plate unloading	Inspection		Inspection		
#10	Plate unloading		Plate unloading		Plate unloading		

2.2.2 Machines

The cell is composed of seven machines, two of which being optional and not used in this work. The machines are represented in Figure 2.3 with the symbols M_1 to M_7 , being:

- M_1 : loading/unloading unit.
- M_2 , M_3 and M_4 : assembly workstations.
- M_5 : automatic inspection unit.

- M_6 : manual recovery unit (not used).
- M_7 : extra assembly workstation (not used).

The assembly workstations are composed of KUKA Robots responsible for the mounting operations. Table 2.3 shows the different operations executed by each machine, together with the corresponding manufacturing processing time of each operation.

Table 2.3: Manufacturing operations processing times.

	M_1	M_2	M_3	M_4	M_5
Plate loading	10				
Plate unloading	10				
Axis		20	20		
r_comp		20	20		
l_comp				20	
L_comp		20		20	
Screw_comp			20	20	
Inspection					5

2.2.3 Conveyor System

The conveyor system is composed of a main loop, four transversal sections composing multiple inner loops, several derivations to reach the machines and positioning units in front of machines. The transversal sections are responsible for the material-handling flexibility.

As it can be noticed in Figure 2.3, in the conveyor system there are also transfer gates responsible for the changes of direction of the jobs, represented in turquoise, and divergent routing nodes in which routing decisions must be made, represented with the symbols n_1 to n_{11} .

The empty plates where jobs are assembled are transported by self-propelled shuttles in the conveyor system. Those shuttles are stored in the shuttle storage area whose capacity is assumed unlimited.

The theoretical transportation times between adjacent nodes can be found in [38].

2.2.4 Operation

The jobs are loaded one by one in the loading unit M_1 . Then, the decision to which machine the job will go in order to complete its production sequence will be made in the routing nodes, and the job will be transported to the operating area in front of that machine.

Before each machine stands a job input storage area with capacity limited to one shuttle for all machines. This area is destined for a job to wait while other job is receiving mounting operations in the machine. If the operating and the job input storage area of the machines that can execute the next operation in the production sequence are occupied, the job will circulate around the outer loop of the

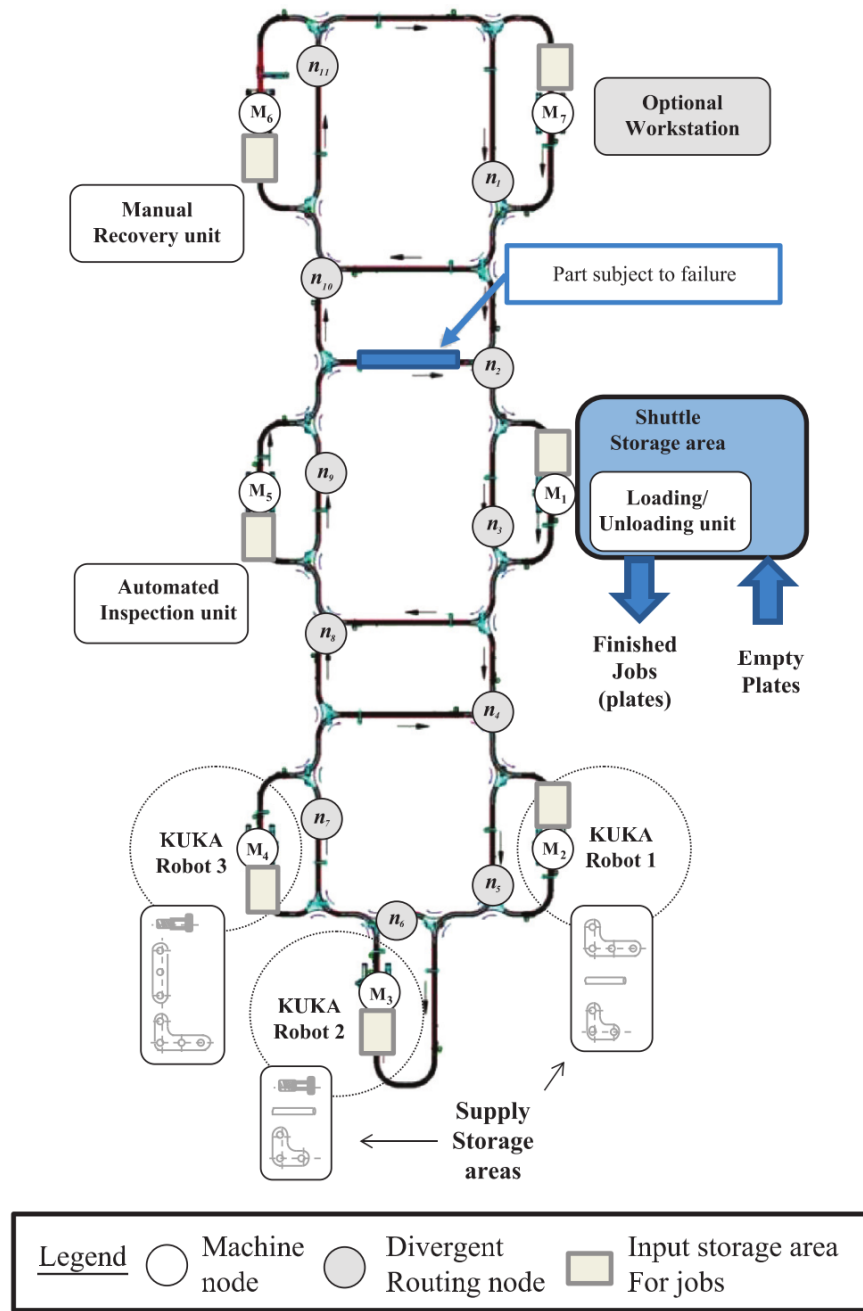


Figure 2.3: AIP-PRIMECA cell layout. (Adapted from [38].)

conveyor system while waiting for a place to get free. The transfer time the shuttle needs to move from the job input storage area to the operation area in front of the machine is neglected for all machines.

Each assembly workstation has its own supply storage area with all the necessary components for the mounting operations it executes. This supply is assumed infinite.

After completing all the mounting and inspection operations in the production sequence, the jobs return to workstation M_1 to be unloaded. At each moment, a maximum of ten jobs can be inside the cell at the same time.

2.3 Proposed IAL Manufacturing System

The proposed flexible production system was designed with the goal of exploiting some existing resources of the IAL at Instituto Superior Técnico.

In this laboratory, eight stands can be found, called Industrial Automation Stands, each composed of industrial automation elements like a conveyor belt with image recognition, an elevator, three pneumatic cylinders, several lights, different types of buttons and even an emergence sensor and an alarm.



Figure 2.4: Industrial Automation Laboratory (IAL).

Considering that those eight stands are placed in groups of four, as it can be seen in Figure 2.4, the goal was to propose a flexible system with a group of four stands as workstations working simultaneously and executing some redundant operations between them. In order to make the system feasible, a manipulator robot would be added to the centre of the four stands, to move the jobs between them, and a conveyor belt would be placed between the two stands on the right, to transport the jobs from the outside to the robot and vice-versa. Having the conveyor belt, the manipulator robot, and the four workstations, some of the industrial automation elements contained in each stand would be used to emulate the remaining functionalities of the production system.

Since the benchmark presented in the previous section already comprises the required flexibility and complexity for the design of an agent-based manufacturing control system, the IAL Manufacturing System was designed analogously to the AIP-PRIMECA FMS, so that a direct comparison of performance can be carried out. Therefore, the products, components, machines, operations, processing times and transport times of the AIP-PRIMECA FMS will be the same in the IAL Manufacturing System, but with a different physical configuration.

From the industrial automation elements present on each stand, the conveyor belt, the elevator, the three pneumatic cylinders and the green, red and yellow lights were selected to integrate the proposed manufacturing system, providing a sufficient quantity of automation elements to design the required

global operation of the system. The analogy between the two manufacturing systems is explained in Table 2.4.

Table 2.4: Analogy between the AIP-PRIMECA FMS and the IAL Manufacturing System.

AIP-PRIMECA FMS	IAL Manufacturing System
M_1 - Loading/Unloading Unit	Conveyor belt
M_2 - Assembly robot 1	Workstation B2
M_3 - Assembly robot 2	Workstation B3
M_4 - Assembly robot 3	Workstation B4
M_5 - Automated inspection unit	Workstation B5
Three mounting operations of each robot	Three pneumatic cylinders of each workstation
Inspection	Three cylinders of workstation B5 at the same time
Free workstation	Green light
Operation being executed	Red light
Job input storage area occupied	Yellow light
Conveyor System	Four conveyor belts + four elevators of the workstations

The flexible conveyor system was modelled using the conveyor belts and the elevators of the workstations as follows:

1. By analysis of the conveyor system layout in Figure 2.3, it can be stated that there are 4 different paths for a job to go from one machine to another, knowing that while the job is waiting for a free spot it will go around the outer loop.
2. These paths are defined in the following way for the example of the trip between M_2 and M_3 : if the job can go directly to machine M_3 , it follows the shortest way just passing near node n_5 ; if it only receives the information that there is a free place in the machine after passing it, it will turn around in the nearest transversal section, which can happen after node n_7 , defining the second path, after node n_9 , defining the third path, or after any of the following nodes, going for a complete turn around the outer loop and in this way defining the fourth path.
3. These four paths can be defined for all the necessary trips between machines in the same way, with the exception of the trip between machine M_3 and M_2 , where the shortest way involves the use of the first transversal section.
4. Having defined these possible paths, the time spent going around the outer loop while waiting for a free place is modelled in the IAL Manufacturing System as a trip in a closed-loop conveyor system composed of the four conveyor belts of the workstations.
5. When the job already knows to which machine is going, the path it uses is modelled in the IAL Manufacturing System by the robot placing the job in one of floors of the elevator of the destination workstation, being each one of the four possible paths represented by a different elevator floor.

Table 2.5 presents the transportation times for each one of the necessary trips that happen inside the manufacturing system. These times were computed by consulting the table of theoretical transportation times between adjacent nodes presented in [38].

Table 2.5: Transportation times between machines (in seconds).

Path	0	1	2	3
M1-M2	11	27	43	59
M1-M3	21	37	53	69
M2-M4	10	26	42	58
M3-M4	7	23	39	55
M2-M3	13	29	45	61
M3-M2	—	16	32	48
M2-M5	18	36	52	66
M3-M5	15	33	49	63
M4-M5	11	29	45	59
M5-M1	12	28	46	60

The final IAL Manufacturing System layout is depicted in Figure 2.5. Here, the robot stands in the centre of the four workstations B2, B3, B4 and B5 in which the industrial elements composing each one of them are clearly identifiable. In each workstation, the conveyor belt and the elevator are positioned closer to the robot, since they are used to model the operation of the conveyor system from the benchmark rather than the operation of the workstations itself. Apart from these two automation elements, the workstations have three pneumatic cylinders each, with the indication of the operation they perform, and three lights to exhibit their internal state. The conveyor belt placed between workstations B2 and B3 also contains the same three lights so that its state can be exhibited similarly to the workstations.

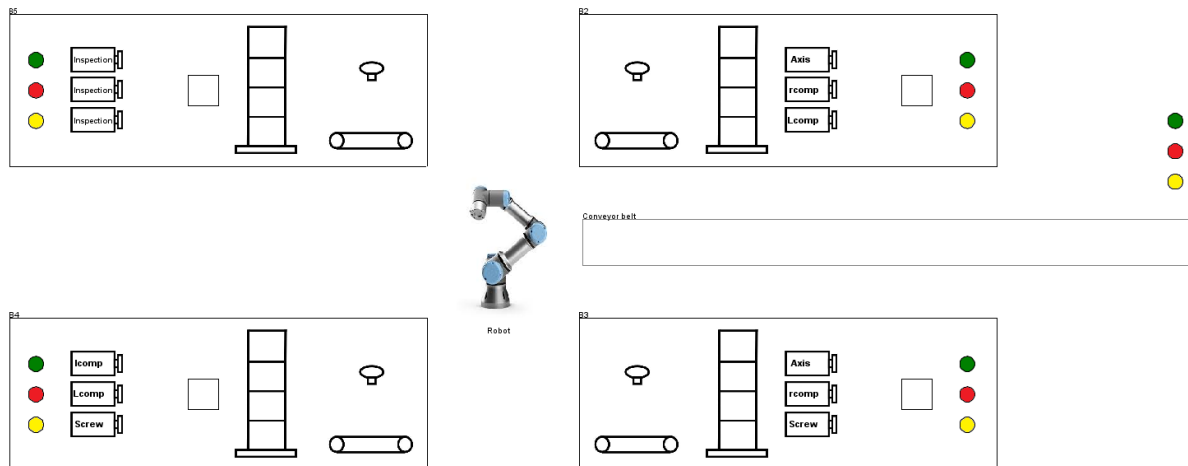


Figure 2.5: IAL Manufacturing System layout.

Chapter 3

Multi-Agent System Architecture

For the control level of the manufacturing system introduced in the previous chapter, a MAS was designed. The application of this system involves a set of distributed, autonomous and cooperative agents representing each one of the elements of the manufacturing system.

As already mentioned in subsection 1.1.1, the FIPA organization has been promoting agent-based technology by openly developing specifications supporting interoperability among agents and agent-based applications. Therefore, the design of the MAS was performed according to the FIPA Specifications, in order to take advantage of the standardized communication, message transport and agent management.

In this chapter of the thesis, a set of FIPA compliant agents and agent interactions will be presented in detail, composing the control architecture of the manufacturing system. As a starting point, a description of the main FIPA Specifications, with emphasis on agent communication, is provided in the first section.

3.1 FIPA Specifications

The FIPA organization is an IEEE Computer Society standards organization since 2005 that promotes agent-based technology by developing specifications.

FIPA Specifications represent a collection of standards which are intended to promote the interoperation of heterogeneous agents and the services that they can represent. These specifications cover different categories, namely agent management, agent communication, agent transport, abstract architecture and applications, among which the core category at the heart of the FIPA MAS model is agent communication [10].

The first specification worth mentioning is the FIPA Agent Management Specification [55], where the Agent Management Reference Model is introduced. This reference model can be seen in Figure 3.1 and is important to understand the normative framework within which FIPA agents exist and operate. Here, the logical components Agent, Agent Platform (AP), Agent Management System (AMS), Directory Facilitator (DF), Message Transport Service (MTS) and Software establish the reference model for the creation, registration, location, communication, migration and retirement of agents.

Still within the scope of the Agent Management Specification, the Agent Life Cycle is defined, containing the following states: Initiated, Active, Waiting, Suspended and Transit.

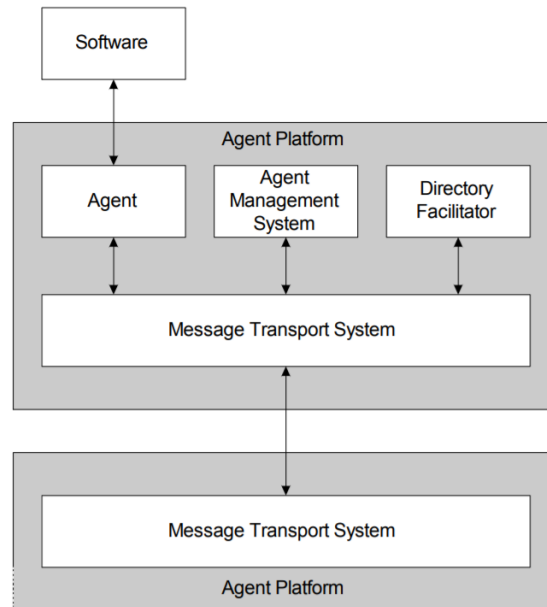


Figure 3.1: Agent Management Reference Model [55].

Regarding agent communication, in the first place it is important to define the structure of a FIPA ACL message, which is composed of the Envelope, the Payload, the Message and the Content of the Message.

In what concerns the Envelope of a FIPA message, it must have the mandatory parameters "to", "from", "acl-representation" and "date". Furthermore, it can also include the optional parameters "payload-length", "payload-encoding", "received" and "security-object".

With a higher importance, the elements that constitute the Message part of a FIPA ACL message are presented in Table 3.1 [56].

One of the key elements of the Message is the performative, where the purpose of the message is stated. The performative is chosen from the FIPA Communicative Act Library [57], which is resumed in Table 3.2.

As to the content of the message, it is encoded in a specific content language. Any language can be used as a content language, with languages like KIF, Prolog, SQL, FIPA-SL, FIPA-CCL, FIPA-RDF, FIPA-KIF being more used than others. Also related to the content, in cases of complex conversations between agents, an ontology containing definitions and relationships between those definitions describing a particular subject domain might be defined, so that all the agents in the conversation can share that ontology and understand the content of the messages.

Combining messages with different Communicative Acts in a common pattern originates Interaction Protocols (IP). FIPA already has a set of pre-defined standard IPs that can be useful for a good amount of agent applications. The FIPA defined IPs are: FIPA-Request, FIPA-Query, FIPA-Request-When, FIPA-Contract-Net, FIPA-Iterated-Contract-Net, FIPAAuction-English, FIPA-Auction-Dutch, FIPA-Brokering, FIPA-Recruiting, FIPA-Subscribe and FIPA-Propose.

Table 3.1: FIPA ACL Message Elements.

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

Table 3.2: FIPA ACL Communicative Acts.

Communicative Acts			
accept-proposal	agree	cancel	cfp
confirm	disconfirm	failure	inform
not-understood	query-if	request	propose
query-ref	refuse	reject-proposal	request-when
request-whenever	subscribe	propagate	proxy

From the above mentioned protocols, the FIPA-Contract-Net is specially preponderant for its wide applicability and known applications. Along with the FIPA-Propose and the FIPA-Request Protocols, these three IPs will have a relevant role in the agent communication throughout this thesis.

The FIPA CNP (Figure 3.2) is an IP used by an Initiator that wants to explore the best proposal to make a contract. It starts with a Call For Proposals (CFP) message sent by the Initiator to all the possibly interested participants, which reply with a Propose or a Refuse message before the stipulated deadline. Among the proposals, the Initiator chooses the one that fits its goals better and answers that proposal with an Accept-Proposal message, answering all the others with a Reject-Proposal. The protocol ends with the chosen participant sending an Inform-done, Inform-result or Failure message to the Initiator.

The Propose Protocol (Figure 3.3) is a very simple IP that only includes a Propose message being sent from one Initiator to one participant. Having received the proposal, the participant answers positively with an Accept-Proposal message or negatively with a Reject-Proposal.

The Request Protocol (Figure 3.4) is an IP used by an Initiator to get the participant to perform a specific action. A Request message specifying the action is sent, being answered with an Agree or a Refuse message. Then, in case the participant agrees to perform the task, it sends an Inform-done or an Inform-result message after executing the required action, or a Failure message otherwise.

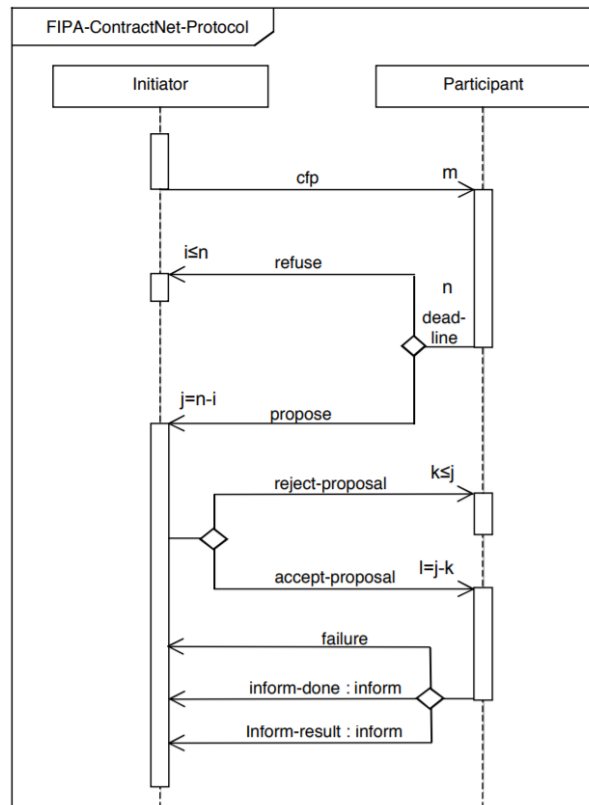


Figure 3.2: FIPA Contract Net Interaction Protocol [58].

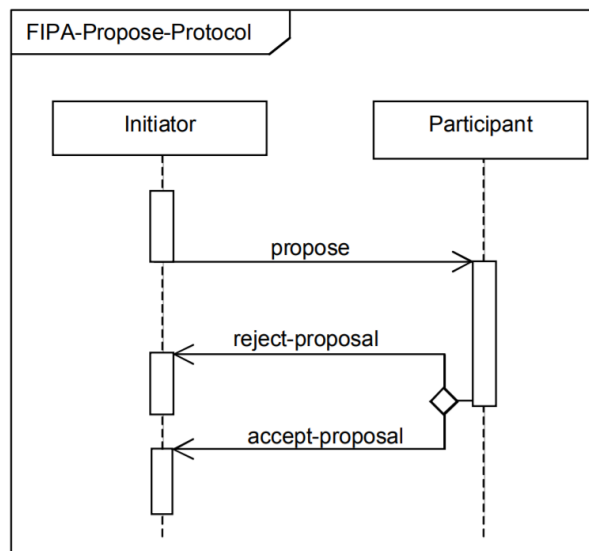


Figure 3.3: FIPA Propose Interaction Protocol [59].

Having described some important specifications useful to understand certain aspects of the design of the MAS, it is time to proceed and introduce the control architecture of the system.

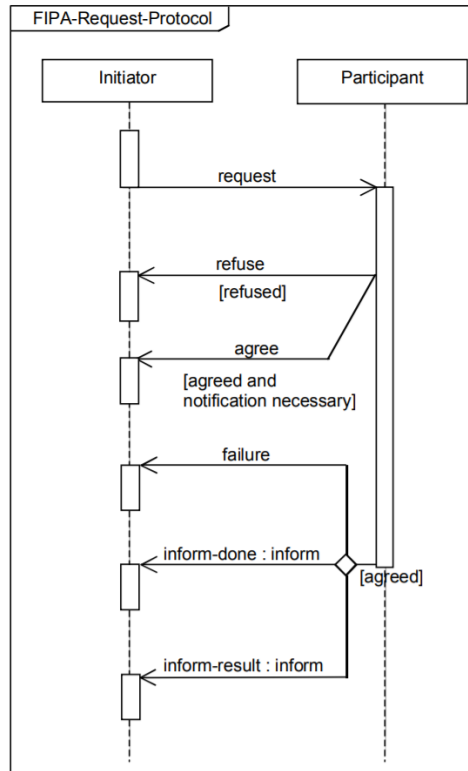


Figure 3.4: FIPA Request Interaction Protocol [60].

3.2 Agents

The agent-based model presented in this thesis was built according to the physical mapping method, by which different agents are used to represent different real physical entities. Taking this into consideration, the designed MAS is composed of five different types of reactive agents: (1) Order Agent; (2) Job Agent; (3) Workstation Agent; (4) Robot Agent; (5) Conveyor Agent. The Order Agents represent the orders submitted by the clients. The Job Agents represent the jobs that are loaded into the production system, necessary to complete the orders. The Workstation Agents represent the four workstations B2, B3, B4 and B5. Lastly, the Robot Agent and the Conveyor Agent represent the central robot and the conveyor belt, respectively.

The creation of five types of agents increases the modularity and flexibility of the system, mapping each different element to an agent with distinct functions and creating a better representation of the system at the logical level. The behaviour of each type of agent was modelled by using the Petri nets formalism [61], which is a tool fit to model and to analyse the behaviour of complex event-driven systems. The Petri net behavioural models for the five types of agents are presented in Figures 3.5-3.9.

An Order Agent is created for each existing order in the beginning of the production and deals with the creation and management of the Job Agents that compose it. The first task it executes is to start the Propose Protocol with the other existing Order Agents, so that, through a simple negotiation algorithm, they can organize themselves and internally define the order in which they will be produced. After concluding the Propose stage, the Order Agent sends a group of Request messages to the Conveyor

Agent, one message for each job it needs to load in the manufacturing system. Whenever the Order Agent receives an Agree message from the Conveyor Agent, it creates the Job Agent corresponding to the job that is being loaded. When a job finishes its production, the corresponding Job Agent sends an Inform-done message to the Order Agent that created it. Once an Order Agent has received all the Inform-done messages from the Job Agents it created, that order is complete and the Order Agent terminates.

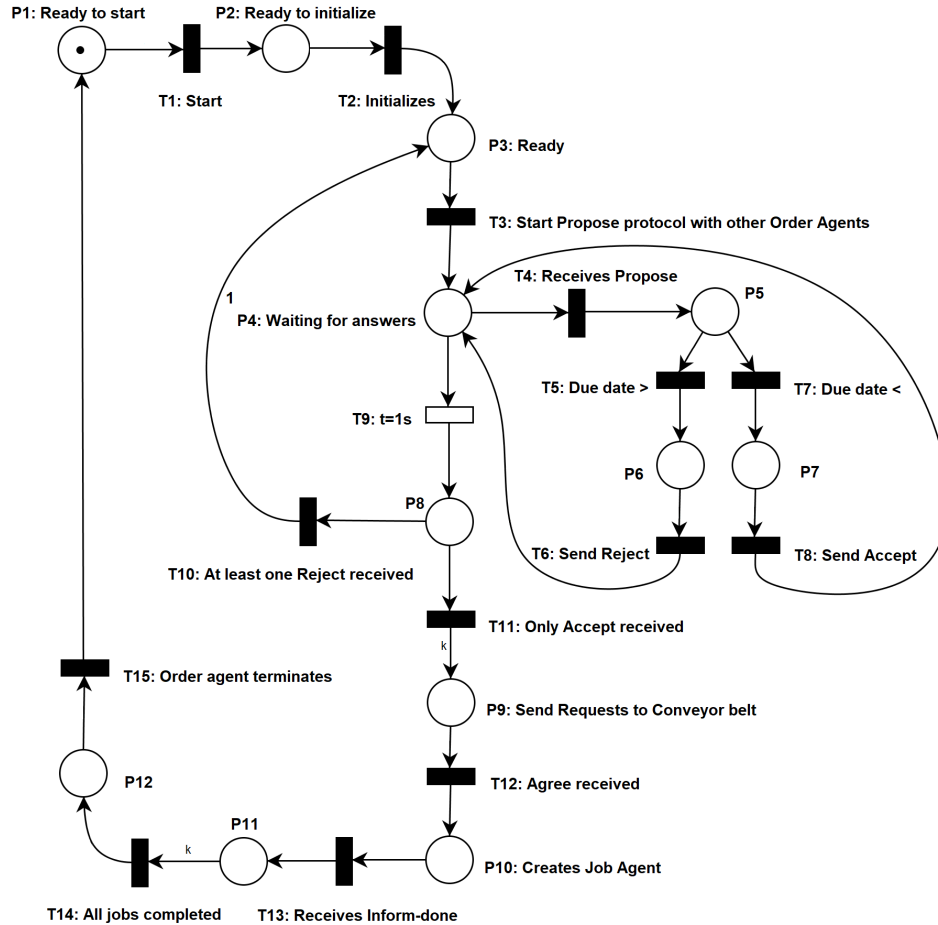


Figure 3.5: Behavioural model of the Order Agent.

The Job Agent manages the route that its job follows inside the production system, by communicating with the workstations, the robot and, in the end, with the conveyor belt. When the Job Agent is created, its job is being loaded in the conveyor belt. When it finishes the loading, the agent starts the CNP with the workstations and analyses the proposals, in order to find the workstation that will execute the next operation in its production sequence. If no proposal is received, the job will travel in the closed-loop conveyor system and the Job Agent will periodically initiate the CNP until a Propose message is received. The Propose messages contain the state of the workstation, which can be "Free" or "CanWait", and the Job Agent prioritizes the proposals of "Free" workstations. As soon as the Job Agent finishes the negotiation with the workstations, it starts the Request protocol with the Robot Agent, requesting to be moved from its current location to the right floor of the elevator of the destination workstation. When it arrives at the workstation, The Job Agent initiates the Request Protocol with the workstation to start

the operation and repeats it as soon as it ends, to start the next one. In the case of positive answers, the job continues in that workstation completing the following production steps, in case of a negative answer, the Job Agent goes back to starting the CNP with the other workstations. After completing all the operations except the unloading, the Job Agent requests the robot to move the job from the current workstation to the conveyor belt. When it arrives, asks the Conveyor Agent to exit the production system, and, when it finishes the unloading, sends the Inform-done message to its corresponding Order Agent and terminates.



Figure 3.6: Behavioural model of the Job Agent.

The Workstation Agents are created in the beginning of the production, one for each of the four workstations, and manage the global operation of the workstation, knowing which operations their workstation can provide. Three states were defined for a workstation and consequently for the Workstation Agent: "Free", when the workstation is empty, "CanWait", when the workstation is executing one operation but the job input storage area is empty, and "Occupied", when both the operating area and the job input storage area are occupied. Regarding its operation, after initializing and registering in the DF, the Workstation Agent executes two main actions simultaneously: responds to CFP messages from agents that initiated the CNP and responds to Request messages from agents that are in the workstation area. When it receives a CFP message, if its state is not "Occupied" and it can provide that operation, the

Workstation Agent answers with a Propose message containing its state as content. In case of a positive answer, it changes its state accordingly. When it receives a Request message, in case it can provide the required operation, the agent sends an Agree message and executes the operation, in case it does not provide the operation, the agent sends Refuse and changes its state accordingly.

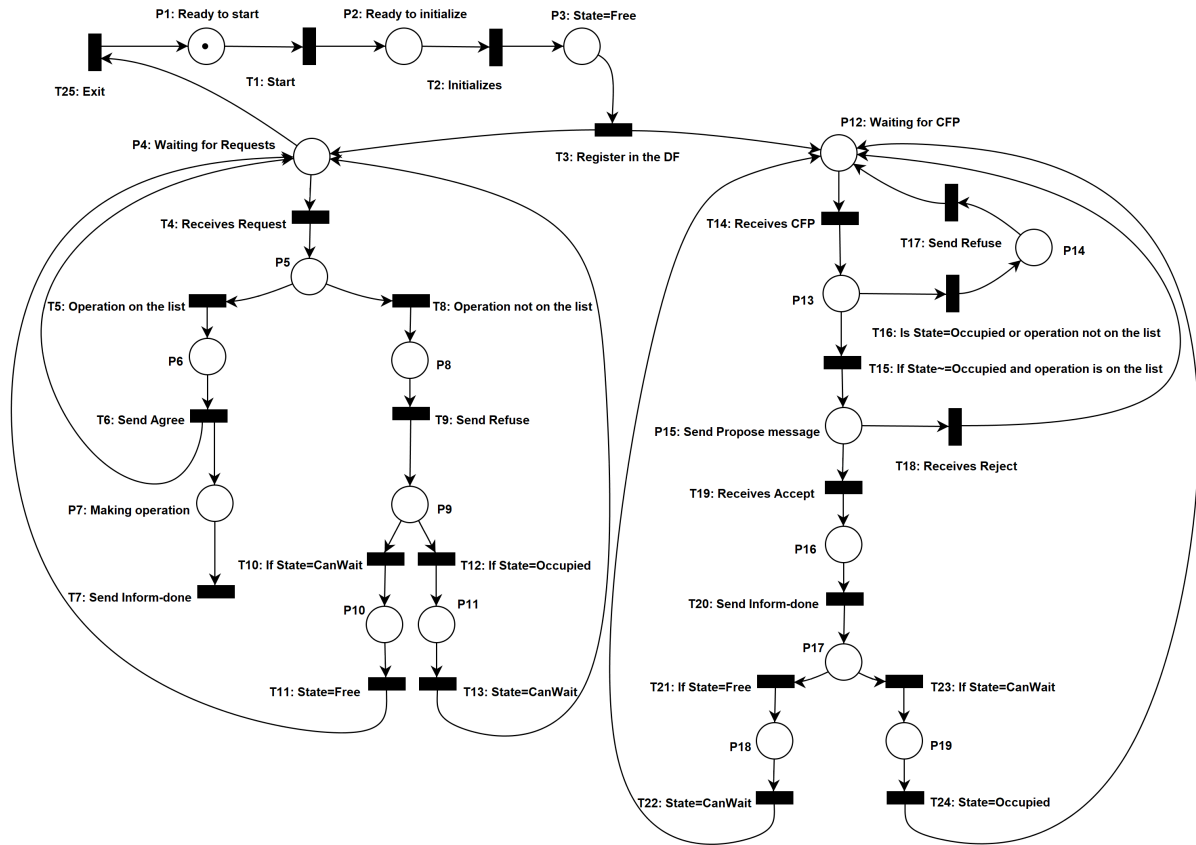


Figure 3.7: Behavioural model of the Workstation Agent.

The Robot Agent is initialized in the beginning of the production and controls the operation of the central robot. This agent receives Request messages from the Job Agents, selects one according to its priorities, in case it has more than one Request in the mailbox, and sends the Agree message to the selected job. Then, it executes the transport of the job between the places specified in the content of the Request message and sends the Inform-done message in the end of the task.

The Conveyor Agent is also initialized in the beginning of the production and has a similar function to the Robot Agent, but for the operation of the conveyor belt. In regard to the operation, it receives Requests from the Order Agents and from the jobs that finished production, selects one request giving priority to the jobs that are exiting the production system and moves the conveyor belt in the proper direction.

The similarities between the behavioural models of the Conveyor Agent and the Robot Agent might lead to think that they could be the same type of agent. The reasons why two different types of agents were designed are the fact that in practical terms the agents control very different industrial components, that require different commands, and the fact that the Conveyor Agent needs to process messages from

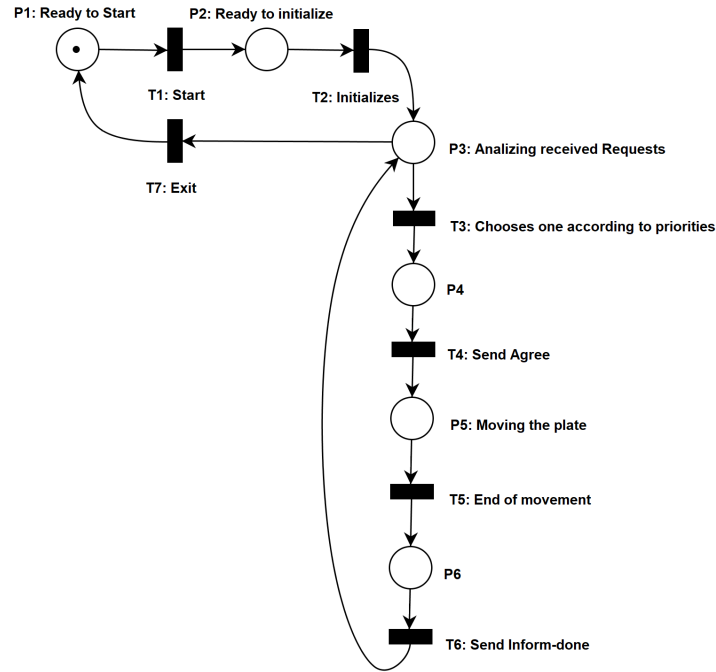


Figure 3.8: Behavioural model of the Robot Agent.

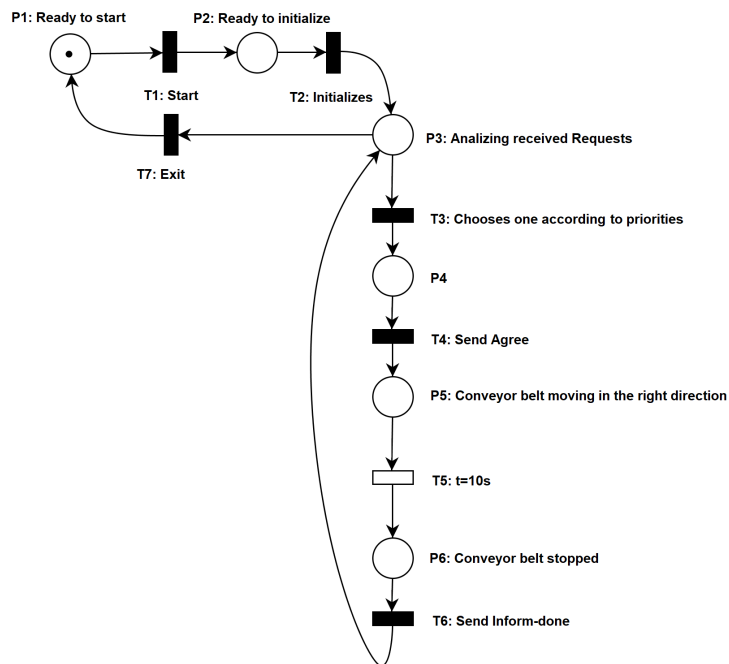


Figure 3.9: Behavioural model of the Conveyor Agent.

different types of agents and even be prepared to process rush orders, while the Robot Agent only needs to process messages from one type of agent but requires a different analysis of the contents of those messages. In the case of a practical implementation of the Robot Agent, an analysis and knowledge of the contents of the messages are required, commanding the robot to do different pre-defined motions.

3.3 Agent Communication

Along the descriptions of the agents behaviour, the use of some FIPA IPs for the agent communication was mentioned. The use of such protocols provides substantial help in the design and implementation of MAS, sparing extra modelling efforts. Furthermore, some agent-related platforms and simulators have been developed in accordance with the FIPA specifications, exhibiting tools to implement FIPA protocols in a faster and more efficient way. Table 3.3 summarizes the communications between the different types of agents.

Table 3.3: Communication protocols.

Agents	Communication
Order Agent - Order Agent	Propose Protocol
Order Agent - Conveyor Agent	Request Protocol
Job Agent - Workstation Agent	Contract Net Protocol, Request Protocol
Job Agent - Robot Agent	Request Protocol
Job Agent - Conveyor Agent	Request Protocol
Job Agent - Order Agent	Inform (message only)

Recalling the behavioural model of the Order Agent in Figure 3.5, the Propose Protocol is used as a means of negotiation between Order Agents, so that they can internally define the order in which they will be produced. The negotiation pseudo-algorithm is in Algorithm 1.

Algorithm 1 Order Agents negotiation pseudo-algorithm.

```

1: Send Propose message to other Order Agents with own Due Date as content
2: while  $t < 1$  second do
3:   Wait for messages
4:   if Receives Propose message then
5:     if Due Date  $>$  Own Due Date then
6:       Send Reject-Proposal
7:     else
8:       Send Accept-Proposal
9:     end if
10:  end if
11: end while
12: if At least one Reject-Proposal received then
13:   Return to step 1
14: else
15:   Send Requests to Conveyor Agent
16: end if
```

The AUML interaction diagram exemplifying this negotiation for one production scenario where there are three orders to be produced is represented in Figure 3.10.

The use of the CNP by the Job Agents has a central role in the agent communications. This is due to the fact that this protocol allows the workstation allocation for each operation the jobs have in their production sequence. A typical example of the interactions happening in the case where job "B" needs to start its production by finding a workstation that can execute operation "Axis mounting" is shown in Figure 3.11.

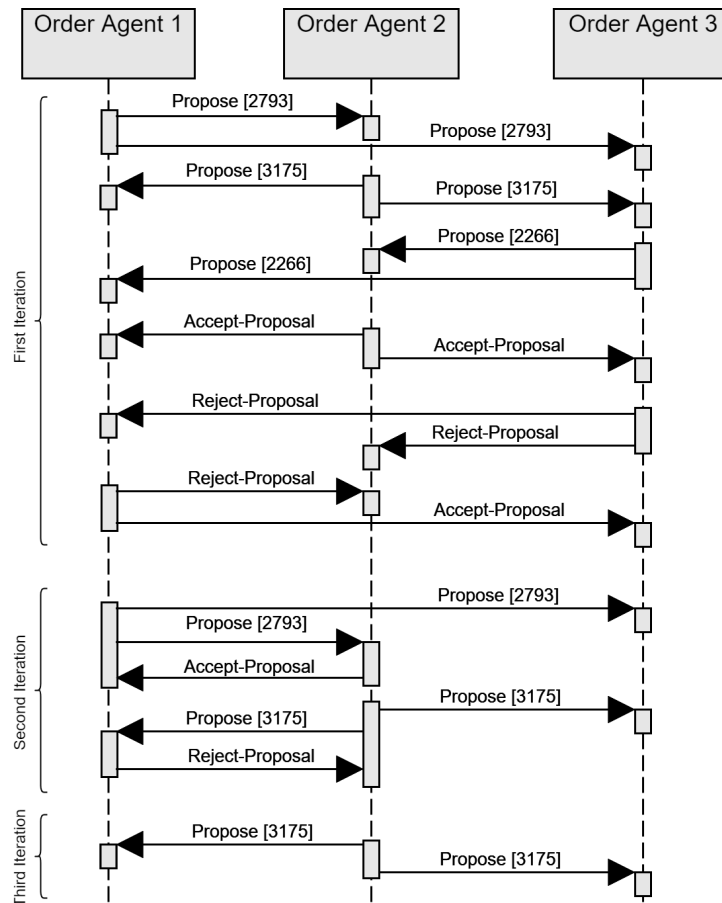


Figure 3.10: Interaction diagram for the Order Agents negotiation.

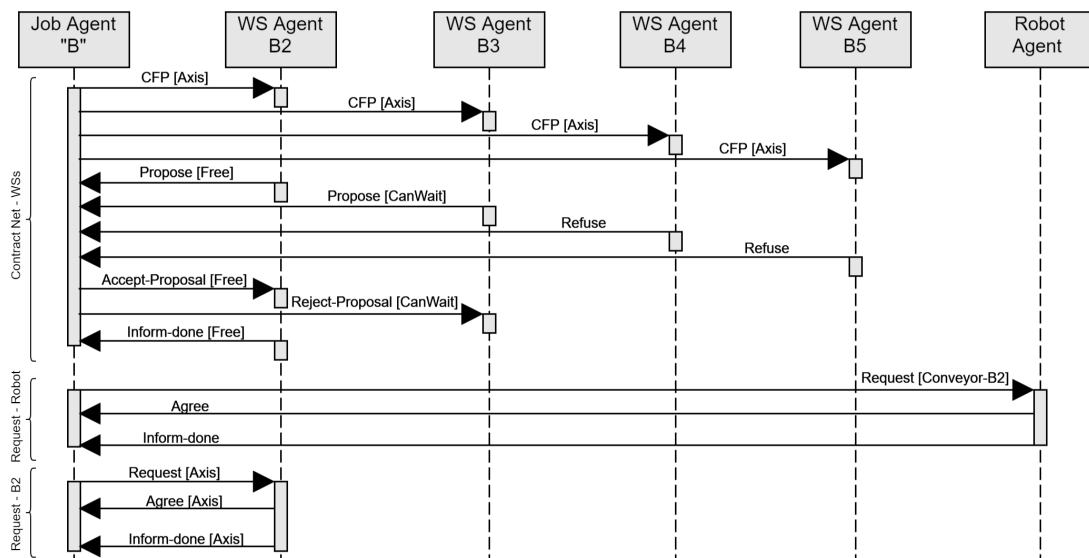


Figure 3.11: Interaction diagram for the example of operation "Axis mouting".

In the diagram, the use of the different protocols is clearly identified. This succession of messages is repeated by all Job Agents whenever they need to find a new workstation to execute the following operation in the production sequence of their job.

3.4 Decision Rules

A key aspect not mentioned up until now is the decision rules that manage the operation of the agents, and consequently the global operation of the system. In regard to this matter, two decision rule bases were defined in this work.

The first one has been presented along the last sections, both in the Petri nets of section 3.2 and in the interaction diagrams of section 3.3. It relates with the fact that the orders load their jobs by Due Date order and the jobs choose the proposal from the first workstation that proposes in the Contract Net, which coincides with the nearest. For instance, workstation B2 is chosen over B3 and B4, and workstation B3 is chosen over B4. The core lines of this rule base are:

- Jobs are loaded by increasing Due Date order of the orders to which they belong.
- Whenever is possible to load jobs in the manufacturing system, they are immediately loaded.
- When analysing the Proposals from the workstations, the jobs accept the first proposal they receive.
- Jobs request all the operations that one workstation can provide them before moving to another workstation.

In addition to this set of decision rules, another rule base was defined and considered to be tested in the control system. These rules were defined so that a different loading order of jobs could be tested and with a view to balance the utilization of the workstations. This attempt to balance workstation utilization is made by having the jobs choosing for the least used workstations whenever they have a choice, which translates to choosing workstation B4 over workstation B2 or B3 whenever they present the same conditions to execute an operation. The core lines of the second decision rule base are:

- Jobs are loaded by the order in which their orders are submitted.
- Whenever is possible to load jobs in the manufacturing system, they are immediately loaded.
- When analysing the Proposals from the workstations, the jobs opt for the least used workstations, specifically B4 over B2 or B3.
- Jobs request all the operations that one workstation can provide them before moving to another workstation.

Chapter 4

Simulation

Simulation plays a central role in the design phase of any system, either for evaluation purposes or to gain an understanding of the system operation.

In order to validate the applicability of the agent-based model presented in the previous section, the first crucial step is to conduct a rigorous simulation. With this in mind, the first step is to select an appropriate agent-based simulation software.

The selection of a simulation software should be a thorough process, since the selected software must present all the required features to allow a proper representation of the system. In the case of an agent-based simulation, the choice should be even more careful, as the software should provide adequate conditions to model agents and their interactions, as well as the manufacturing system itself.

In Table 1.1 from subsection 1.1.1, four up-to-date and qualified agent simulators are presented. In order to conduct the comparison of these softwares, the decisive aspects are the communication and compatibility, characteristics of utmost importance in the implementation of agents. Considering that the agent communication in the designed MAS employs FIPA IPs with standardized messages, the selected simulation software is the GAMA Platform, due to its FIPA compatibility and ability to represent FIPA ACL messages in GAML, the agent-based language used in GAMA.

GAMA is a modelling and simulation development environment for building spatially explicit agent-based simulations [62]. In the context of the GAMA environment, a simulation of the IAL Manufacturing system was created, where all the simulation tests presented in this work took place.

In this chapter, the description of the layout and operation of the developed simulation is presented. Moreover, a detailed description of the Static and Dynamic Scenarios to be simulated is provided.

4.1 Simulation Layout



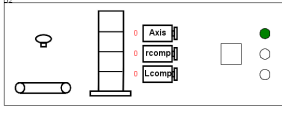


A simulation was developed in the agent-based simulation software GAMA, using the comprehensive agent-oriented language GAML. In GAML, each agent type is defined as a species, and different experiments can be defined for the same group of species within a simulation model. In general terms, a model written in GAML is structured in three parts [62]:

1. Model Header: declaration of the name of the model, optional import statements and definition of the global species (the model species itself).
2. Species declarations: definition of the set of attributes, actions, behaviours and properties of the population of each different species of agents that populate the model.
3. Experiment declarations: definition of the simulation parameters and the required outputs of each experiment to be performed.

Thus, the five types of agents considered in this work were defined in GAML as five different species, specifying the actions and behaviours of each one in accordance with the Petri nets behavioural models presented in section 3.2 of the previous chapter.

It is important to mention that while in an implementation the agents would be just computational scripts controlling real industrial elements of the production system, in the case of the simulation in GAMA the agents have a designated aspect, and so they will appear on the screen as the industrial elements themselves, in order to compose the simulation model of the manufacturing system. Thereby, the aspect of each species of agents was defined and can be seen in Table 4.1. These appearances express the way in which each agent is represented in the simulation of the manufacturing system, composing the simulation layout.

Table 4.1: Agents representation in the simulation.

Order Agent	Job Agent	Workstation Agent	Robot Agent	Conveyor Agent
				

In the experiment declaration, two displays were drafted as outputs: one with the manufacturing system simulation model (Figure 4.2) and the other containing some statistical graphics about the production (Figure 4.3). These displays appear as tabs in the simulation environment of the GAMA platform. Moreover, a simulation parameter was set to alternate between the different Static Scenarios that can be simulated.

Figure 4.1 shows a view of the simulation environment, where the tab with the simulation model is selected and the different areas are marked in red. These areas are the Parameter View on the upper left corner, which contains the selection of the Static Scenario, the Output Console on the lower left corner, where the informations about the operation of the system and the messages sent by the agents are written, and the simulation area in the centre. Inside the simulation area, indication on the location of the Order Agents, the operation area, the job input storage area and the area where the agents that are travelling in the closed-loop conveyor system, while waiting for an available workstation, are placed is provided.

Regarding the operation of the IAL Manufacturing System model, shown in Figure 4.2, the jobs appear and are loaded in the conveyor belt, from right to left. Then, they enter in the closed-loop conveyor

Whenever a simulation is run, the graphics in the Statistics display (Figure 4.3) are built in real time and assume its final form in the end of the simulation. The graphics exhibit some data related to the performance of the manufacturing system during production, namely:

- Time to due date of each order.
- Workstation idle times.
- Number of executions of each operation on each workstation.
- Job throughput times.
- Job waiting times.
- Total Makespan.

By analysis of this data, it is possible to have a better understanding of the performance of each individual workstation as well as to identify the causes of the waiting times. Furthermore, the value of Makespan can be used as criteria to measure the quantitative performance of the system and to allow comparisons to the benchmarks.

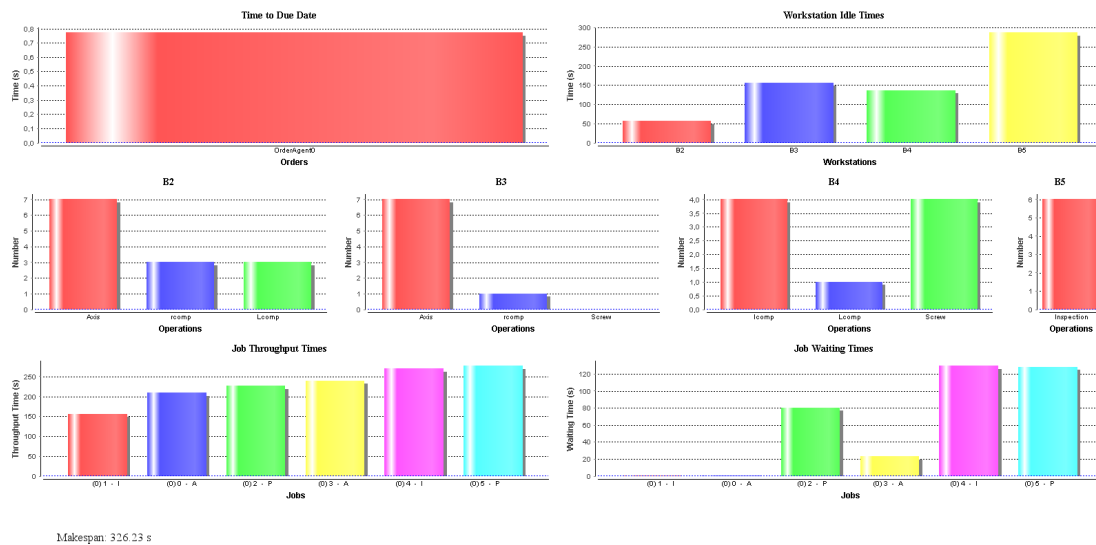


Figure 4.3: Statistics display in GAMA.

In terms of simulation step (sample time), a commitment between trying to have the agent communications happening as instantly as they would happen in an agent platform and not decreasing the sample time so much that the performance of the simulation could be affected needs to be made. This is due to the fact that the simulation is refreshed in every step and so the messages are sent with a step time between them. In this way, considering that the system has its transport and processing times in seconds and the communications cannot induce any noticeable delay, the defined simulation step was 0.01 seconds.

4.2 Static Scenarios

In the simulation, the manufacturing control system is tested under different conditions defined in static and dynamic scenarios.

A static scenario is a production scenario where all the data is known initially, this meaning that there are no perturbations to be considered. This type of scenario is usually defined as the reference performance of the system, being used for comparisons with other possible production scenarios that may involve perturbations.

In this thesis, the static scenarios to be tested were selected considering the amount of available data for comparisons. Hence, the table of possible static data sets introduced in [38] was considered and the scenarios C0 and B0 were selected. The data related to each static scenario is presented in Table 4.2. In this table, the number of shuttles allowed in the production system and the specification of the orders corresponding to each scenario are provided. This specification comprises the quantity of each product required and the due date for each order.

The scenario C0 will be the main reference for the operation of the manufacturing system. This happens once the dynamic scenarios will be simulated under C0 conditions and the differences of performance will be measured in comparison to C0.

Table 4.2: Static Scenarios data.

Scenario	No. of shuttles	Order #	BELT	AIP	LATE	Due date
C0	4	#1	1	-	-	382
		#2	-	1	-	238
B0	10	#1	-	2	-	327

4.3 Dynamic Scenarios

Real life production system events are dynamic, facing perturbations that cannot be predicted in the beginning of production. A dynamic scenario is characterized by having certain data that is unknown in the initial time, generally perturbations.

In order to test the designed control system under the existence of some perturbations, a set of dynamic scenarios was selected. Once more, these scenarios were mainly selected with a view to comparing performances with existing data of the benchmark production system. Nevertheless, attention was paid to choosing a varied selection of dynamic situations that would test different capabilities of the system. Taking this into consideration, the dynamic scenario list introduced in [38] was used and the scenarios #PS5, #PS7, #PS9 and #PS10 were selected. Table 4.3 presents the description of those scenarios in the context of the AIP-PRIMECA FMS.

The dynamic scenario #PS5 evaluates the capacity of the system to manage the introduction of a rush order. This situation is very common in the real world, since the client desires are unpredictable

Table 4.3: Dynamic Scenarios data.

Scenario	Description	Parameters
#PS5	At a given time, a rush order appears.	Type of order: AIP. Arrival Time: just after the end of the production of the fourth job in the cell. Due date: ASAP.
#PS7	At a given time, a part of the conveyor system is due for maintenance in a given time window.	Start time: just after the fourth job is unloaded. The conveyor must no longer accept shuttles, and as soon as it is empty, the maintenance starts. Duration (seconds): $25 \times \text{Total number of jobs}$.
#PS9	At a given time, one of the redundant machines will go down in a given time window.	Machine: M2. Start time: just after the departure of the first shuttle from M2. Duration (seconds): $25 \times \text{Total number of jobs}$.
#PS10	At a given time, one of the critical machines will go down in a given time window.	Machine: M4. Start time: just after the departure of the second shuttle from M4. Duration (seconds): $25 \times \text{Total number of jobs}$.

and may evolve over time. As a response to the submitted rush order, the system should prioritize it and have its jobs skipping the queue to be loaded in the production system.

Scenario #PS7 simulates a maintenance operation in one part of the conveyor system, which makes that part unavailable to carry jobs. The capacity of the control system to manage a routing change is tested, with jobs having to opt for an alternative route (in the IAL Manufacturing System this situation is visible in the different transportation times).

In the scenarios #PS9 and #PS10, two different breakdown situations are evaluated. In the first, one of the machines that only executes redundant operations will go down. The specified machine is M2 which translates to workstation B2 in the IAL Manufacturing System. This situation causes the production system to loose production capacity but the production does not need to stop at any moment. In the case of #PS10, it is a critical machine that goes down in a given time window, leaving the system with one or more operations that cannot be executed during that time. This situation can cause production to stop as there will be some jobs that will not find a workstation for certain operations. In the context of these scenarios, the specified machine is machine M4, equivalent to workstation B4.

The results of the simulation tests under the conditions of the described scenarios will be presented in chapter 6 of this thesis.

Chapter 5

Experimental Implementation

The real world presents a variety of new challenges compared to the simulated world. In order to perform the implementation of a simulated system in the real world, hardware and connectivity issues need to have a significant focus, since they represent the root of a generous amount of implementation problems. Some of these problems have its cause in the different proprietary protocols provided by the hardware automation devices, which make the process of integrating hardware devices complex and time consuming. However, due to standards like OPC or OPC-UA, which intend to promote hardware integration at all levels of an industrial facility, this current landscape is changing [63].

With a view to implementing the agent-based system previously introduced and simulated, some experiences were conducted so that the conditions under which the system would be implemented could be understood.

The first challenge addressed is the integration of agents with hardware, where the hardware component is a PLC from the IAL. A simulation with hardware-in-the-loop was conducted, aiming to connect one of the simulated agents with the PLC during the simulation time.

Then, the focus will be on the deployment of agents in a real experimental facility, which needs to be performed by means of an appropriate agent platform.

Lastly, an illustrative example on how the agents would be implemented is presented, approaching the allocation of workstations through the use of the CNP.

5.1 Hardware integration

The implementation of an agent-based system requires the integration of agents with physical devices. Recalling section 1.1 of the first chapter, there are mainly two ways of performing this integration: embedding the agent within the physical control device or connecting the agent with the existing control device in a coupled approach. The reasons to use one of the approaches are related to the ability of control platforms to host the agents and the required level of control, which can be focused on real-time control capabilities or on more visualization and adaptation [63].

When dealing with low to medium range industrial PLCs, the chances are that an embedded ap-

proach will not be really suitable and easy to implement, since the controllers will not provide the necessary programming resources to implement the agents. Thus, a coupled approach might be preferable, allowing the use of all the existent automation equipments as long as they enable communication between devices.

In what concerns the technologies to interface automation devices, Table 5.1 summarises some industrial communication technologies which use has been described in the literature, going from the traditional serial communication to the modern OPC-UA that considers cloud integration [63][8].

To assess the possibility of implementing the proposed control strategy, a simulation with hardware-in-the-loop was conducted. The integration of agents and hardware addressed in this simulation was the connection between one of the simulated agents and one of the existent PLCs in the IAL. The goal was to have one of the simulated Workstation Agents communicating in real time with a PLC from one of the automation stands. In this way, the simulation would run normally but one of the Workstation Agents would instruct the PLC to execute the given operations and wait for it to complete the operations before proceeding.

In order to perform this experience, a group of PLC variables designated "Flags" was created to define the communication channels between the agent and the PLC. Furthermore, the PLC was programmed in a way that the activation of each variable would originate the execution of the required task autonomously. The Instruction List source code is presented in Appendix A.

Table 5.2 exhibits the PLC variables to which the agent can access to command the PLC or to consult the state of the workstation. The first three variables ("Free", "JobWaiting" and "Operating") relate to the state of the workstation and the other nine represent commands that the agent can give to the PLC. After instructing the PLC to perform one of the operations "Axis", "rcomp" or "Lcomp", the agent uses the variable "Operating" to perceive when the operation is concluded and proceeds from that point.

In what concerns the communication technologies, the given PLC is a Saia Burgess Controls® PCD3.M3330 that allows Ethernet Communication. As to the GAMA simulation software, it was not originally conceived for network communications, since it is a simulation environment built to contain all the simulation components. For this reason, an intermediate software was used and an alternative communication method was defined between this software and the simulation.

The selected intermediate software was a MATLAB Script. In what concerns the communication between the simulation and MATLAB, the alternative communication method consists in having both softwares reading and writing from the same data file, which is a .txt file containing as content the state of the twelve PLC variables. The scheme of this interface is displayed in Figure 5.1.

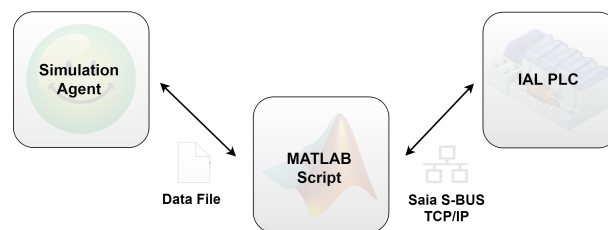


Figure 5.1: Interface between a simulation agent and a PLC.

Table 5.1: Interface Technologies [63][8].

Technology	Overview
Serial link RS232	RS232 is a communication channel where the data is transmitted in a serialized manner according to a proprietary protocol. The communication configuration should be somehow hard coded according to the device communication protocol, making this way of communication really vendor dependent.
MODBUS	MODBUS is a serial communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its PLCs. Following a simple communication protocol, the access to input and output registers of PLCs is possible by using serial or Ethernet communication.
PROFIBUS	PROFIBUS is the fieldbus-based automation standard of PROFIBUS & PROFINET International. Via a single bus cable, PROFIBUS links controllers or control systems with decentralized field devices on the field level and also enables consistent data exchange with higher ranking communication systems.
PROFINET	PROFINET is an industry technical standard for data communication over Industrial Ethernet, designed to control and collect data from equipment in industrial systems, with a particular strength in delivering data under tight time constraints.
EtherCAT	EtherCAT (Ethernet for Control Automation Technology) is an Ethernet-based fieldbus system, invented by Beckhoff Automation. The protocol is standardized in IEC 61158 and is suitable for both hard and soft real-time computing requirements in automation technology.
OPC-UA	OPC-UA (Open Platform Communications - Unified Architecture) is an open, vendor- and platform-independent standard developed by the OPC Foundation for horizontal communication from machine to machine (M2M) and for vertical communication from machine all the way to the cloud.
TCP/IP	TCP/IP (Transmission Control Protocol/Internet Protocol) is the conceptual model and set of communications protocols used on the Internet and similar computer networks. The Internet protocol suite provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed, and received.
CIP	The Common Industrial Protocol (CIP) aims a complete integration of control with information over Internet technologies. It allows users to integrate these manufacturing applications with enterprise-level Ethernet networks and the Internet.
CORBA	CORBA is based essentially in the Object Request Broker (ORB) concept, which allows a local client to invoke methods on a remote platform as if they were local. It is a standard defined by the Object Management Group (OMG) designed to facilitate the communication of systems that are deployed on diverse platforms.
REST	Representational State Transfer (REST) is an architectural style that defines a set of constraints and properties based on HTTP. Web Services that conform to the REST architectural style, or RESTful web services, provide interoperability between computer systems on the Internet.
USB	USB, short for Universal Serial Bus, is an industry standard that was developed to define cables, connectors and protocols for connection, communication, and power supply between personal computers and their peripheral devices.

As to the MATLAB function, it is running a script that periodically updates the values of the variables from the data file and the PLC to be in accordance with each other. The MATLAB Script pseudo-algorithm is presented in Algorithm 2 and the complete source code is displayed in Appendix A.

Table 5.2: PLC variables.

PLC variable	Description	Address
Free	Active if the workstation is Free.	F 1087
JobWaiting	Active if a job arrived to the job input storage area.	F 1087
Operating	Active when the workstation is executing one operation.	F 1089
Axis	Command to perform operation Axis.	F 1091
rcomp	Command to perform operation rcomp.	F 1092
Lcomp	Command to perform operation Lcomp.	F 1093
ElevadorP0	Command to position the elevator on the floor zero.	F 1112
ElevadorP1	Command to position the elevator on the floor one.	F 1113
ElevadorP2	Command to position the elevator on the floor two.	F 1114
ElevadorP3	Command to position the elevator on the floor three.	F 1115
TapeteOnOff	Command to switch on the conveyor belt.	F 1094 (O 19 to write)
TapeteDirec	Command to define the conveyor belt direction.	F 1095 (O 20 to write)

The script reads the values of the variables from the PLC, then reads the values of the variables in the data file and, in case they are different, writes those values in the PLC. Afterwards, reads the variables from the PLC again and, in case any of them is already changed, writes the states of the PLC variables in the FILE to update it.

In regard to the interface performance, the simulation has a defined step of 0.01 seconds and the same sample time was set for the operation of the MATLAB script, aiming at a real time communication between the simulated agent and the PLC.

Algorithm 2 Interface MATLAB Script pseudo-algorithm.

```
1: Define 2 vectors with the addresses to read and the addresses to write in the PLC
2: while stop  $\neq$  0 do
3:   for  $i = 1 : n$  do
4:      $ValuesPLC(i) \leftarrow$  read value from reading address  $i$ 
5:   end for
6:   Open data file to read
7:   while  $i \leq n$  do
8:      $ValuesFILE(i) \leftarrow$  read value from line  $i$ 
9:     if  $ValuesFILE(i) \neq ValuesPLC(i)$  then
10:      Write  $ValuesFILE(i)$  in the writing address  $i$ 
11:    end if
12:     $i = i + 1$ 
13:  end while
14:  Close data file
15:  for  $i = 1 : n$  do
16:     $ValuesPLC(i) \leftarrow$  read value from reading address  $i$ 
17:    if  $ValuesPLC(i) \neq ValuesFILE(i)$  then
18:      Increase different counter
19:    end if
20:  end for
21:  if different  $\neq$  0 then
22:    Open data file to write
23:    Write  $ValuesPLC$  in the file
24:    Close file
25:  end if
26: end while
```

5.2 Agent Development Platform

The development and deployment of MAS requires the implementation of features which are not usually supported by regular programming languages, such as message transport, yellow and white pages services, agent life-cycle management services and support to ontologies. In order to simplify the development of agent-based applications and save some programming effort, it is important to use an adequate agent platform which is prepared to implement the afore mentioned features.

In Table 1.1, four up-to-date agent platforms were introduced (Agent Factory, Jadex, JADE and JACK), which provide different services and agent models and come from different developers and domains. When it came to choose the most suitable agent platform, some criteria was defined:

- To be an open source platform.
- Existence of good documentation and available support.
- Compatibility with the FIPA Standards.
- Agent management features available.

Analysing the four presented agent platforms, JADE is the one that better fits the criteria, justifying why it is one of the most widely used platforms for agent-related research purposes.

JADE is an open source platform for peer-to-peer agent based applications that aims to simplify the development of MAS by providing a set of system services and agents in compliance with the FIPA

specifications. JADE uses the FIPA-ACL as the communication language to represent messages and provides the mandatory components defined by FIPA to manage an agent platform, which are the Agent Communication Channel (ACC), the AMS that provides white pages and agent life-cycle management services and the DF that provides yellow pages services and the capability of federation within other DFs on other existing APs. Furthermore, a JADE-based system can be distributed across several machines which do not need to share the same Operating System, as long as they support the Java Virtual Machine (JVM) [64].

Figure 5.2 represents the main JADE architectural elements. Agents live on top of a Platform which is composed by a Main Container (where the AMS and the DF are included) and other optional regular containers. The AP provides basic services to the agents, such as message delivery, and the containers can be executed on different hosts, composing a distributed AP. In addition, JADE is prepared to execute all the necessary communications via TCP/IP, ensuring message transport.

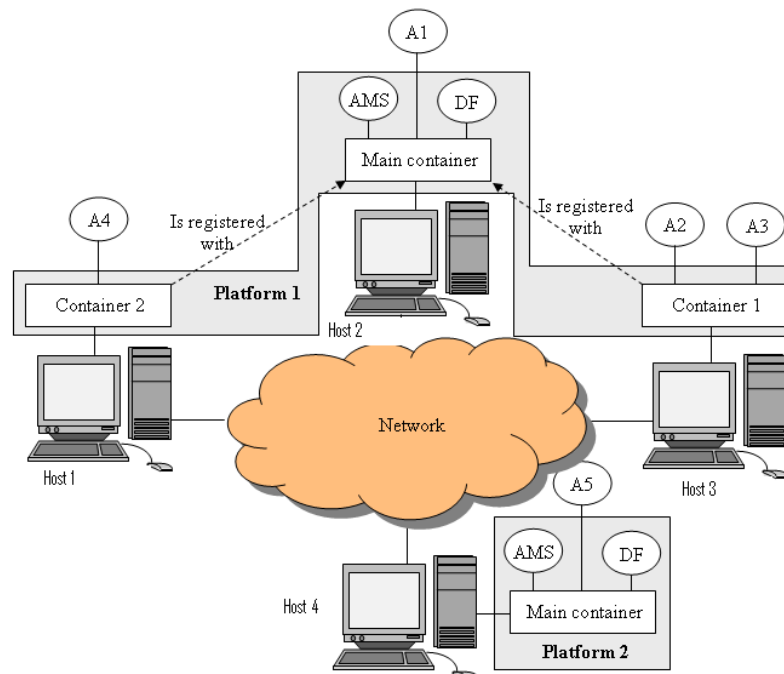


Figure 5.2: The JADE Architecture [64].

The remote management of agents within an AP is performed through the use of a Graphical User Interface (GUI), displayed in Figure 5.3 (a), where the agents can be monitored and controlled. Moreover, JADE provides a set of other graphical tools to support the debugging phase, which can be a complex process in distributed systems. These tools are the *Dummy Agent*, which is a monitoring and debugging tool that allows editing, composing sending and receiving ACL messages from agents, the *Introspector Agent*, which allows to monitor and control the life-cycle of a running agent, together with its exchanged ACL messages and behaviours in execution, and the *Sniffer Agent* (Figure 5.3 (b)), that allows to track messages exchanged within a JADE AP using a similar notation to the AUML interaction diagrams.

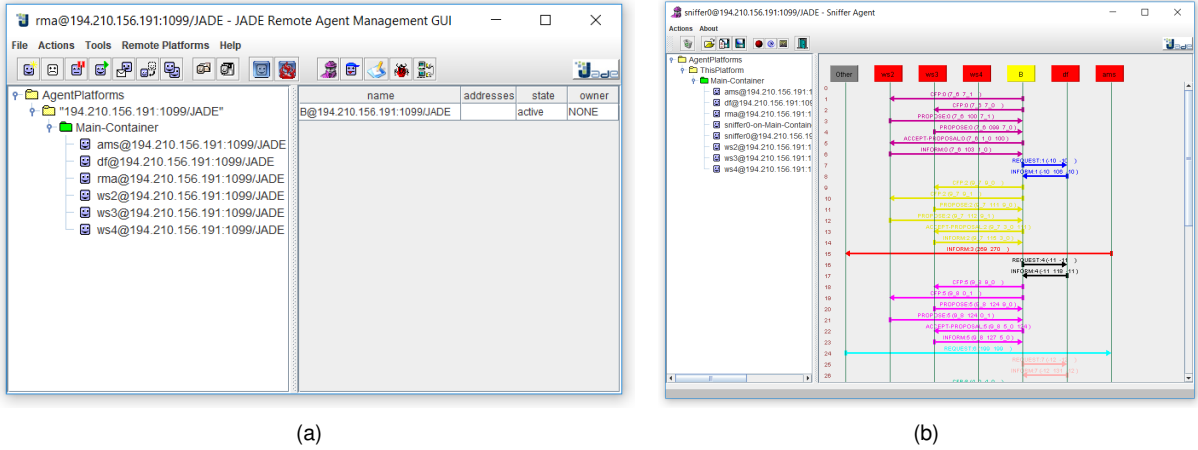


Figure 5.3: JADE Management GUI (a) and JADE Sniffer Agent GUI (b).

5.3 Contract Net Protocol Example

The deployment of the five types of agents that compose the designed MAS in the real experimental facility through the use of JADE would be the next step in the implementation of the given agent-based system. However, due to the limited duration of this research work, it was not possible to accomplish the complete agent deployment.

An illustrative experience on how the agents would be implemented in the experimental facility was performed, using JADE and two networked computers. The experience portrays the application of the CNP in the allocation of workstations for each operation contained in the production sequence of a job.

In a regular JADE application, each type of agent is defined as a JAVA class, and programmed by taking advantage of JADE built-in features like the concept of behaviour to model concurrent tasks and a library of FIPA Interaction Protocols ready to be used. Regarding the fact that both the simulation software GAMA and the platform JADE are compatible with the FIPA ACL for the communications among agents, the transition from GAMA to JADE in terms of agent programming is facilitated, since all protocols used by the agents in the simulation are promptly available in JADE and thus the structure of the JAVA classes in JADE would be similar to the structure of each species in GAMA.

Illustrative parts of both the Job Agent and the Workstation Agent were implemented in JADE. The source code of the corresponding JAVA classes can be consulted in Appendix A.

The programmed operation of the Job Agent goes as follows:

1. The agent gets the operations in the production sequence as start-up arguments.
2. Creates a sequential behaviour that will repeat itself for all the operations in the sequence.
3. The first part of the behaviour comprises requesting to the DF the workstations which provide the next operation in the production sequence and filling the CFP message to send them.
4. The second part of the behaviour is the execution of the CNP by sending the CFP messages and by answering the first Propose message that it receives with an Accept-Proposal.

5. Lastly, after concluding the CNP successfully, the agent waits the corresponding operation processing time before proceeding.

6. After completing all the operations read as arguments, the agent terminates.

As to the Workstation Agent, it was programmed as follows:

1. The agent gets the operations which its workstation provides as start-up arguments.
2. The agent registers the operations in the yellow pages (DF).
3. Creates two cyclic behaviours that will repeat themselves until the end of the production.
4. In the first behaviour the agent answers the CFP message with a Propose message in case the content of that message contains one of the operation its workstation provides.
5. In the second behaviour the agent answers the Accept-Proposal message with an Inform-done message and becomes unavailable in a waiting mode during the processing time of the operation.
6. Before the Workstation Agent terminates, it de-registers from the DF.

To conduct the experience, JADE was launched in one of the computers, creating the AP, the Main Container and starting the Remote Agent Management GUI, and three Workstation Agents ("WS2", "WS3" and "WS4") were created in the "Main-Container". Then, using the other computer, a second container called "Container-1" was remotely launched in a way it would belong to the same AP. After the creation of the remote container, it became visible in the Remote Agent Management GUI of the first computer and a Job Agent "B" was created.

Figure 5.4 displays a screen-shot of the operation of JADE, where the Remote Agent Management GUI can be seen in the upper left corner, the command prompt where JADE was launched can be seen in the lower left corner and the exchanged messages between the Job Agent "B" and the Workstation Agents "WS2", "WS3" and "WS4" can be observed in the *Sniffer Agent* GUI on the right.

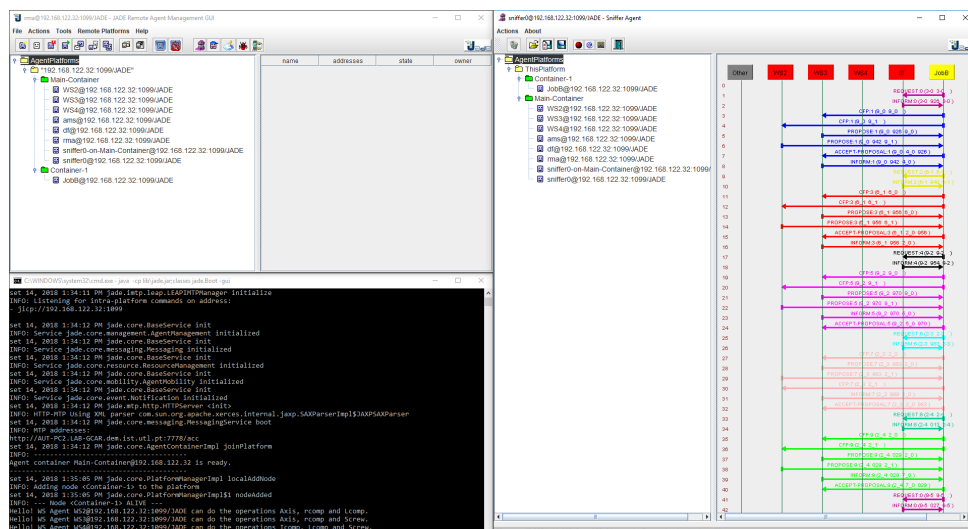


Figure 5.4: Exchanged messages between agents in different computers.

Looking closely to one section of the exchanged messages, depicted in Figure 5.5, it is possible to identify the pattern: before initiating the CNP with the workstations, marked by the CFP messages being sent, the Job Agent sends a Request message to the DF to query the list of workstations that provide the next operation in the sequence. Then, it can be noticed that the Job Agent answers to the first Propose message that it receives and the CNP gets concluded with the Inform-done message.

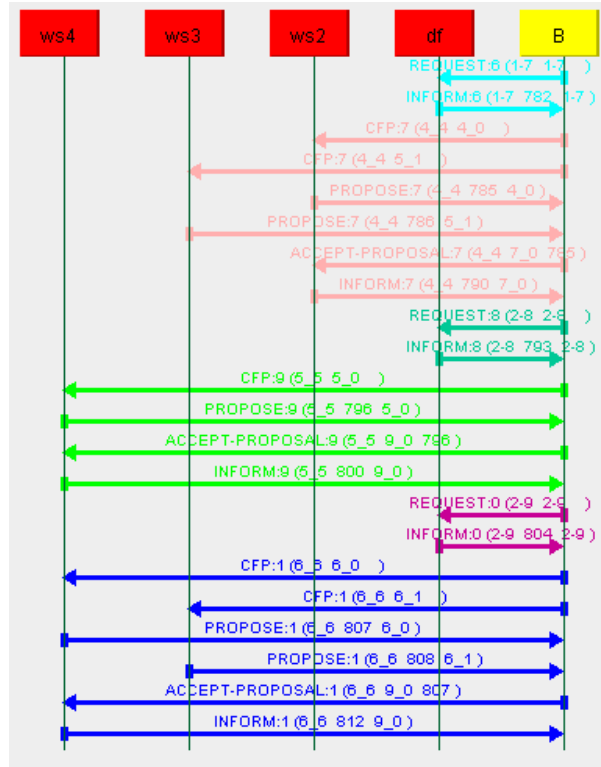


Figure 5.5: Exchanged messages with Job Agent "B".

Recalling the MAS model presented in Chapter 3, the agent communications displayed in JADE are slightly different from the ones designed for the system and presented in the simulation. That relates to the fact that this illustrative example was not developed with the same depth as the simulation, where the agents do not get unavailable and always respond to all the messages.

Another section of exchanged messages is displayed in Figure 5.6 for the case of two Job Agents communicating with the Workstation Agents. In this figure, jobs "B" and "E" alternately query the DF to find which workstations can provide the next operation in their production sequence and start the CNP afterwards. It can be noticed that in some cases the Propose message from one workstation only arrives after the conclusion of the CNP with the other. This is due to fact that while a workstation is executing an operation it does not answer CFP messages, only sending the response after finishing the operation it is providing.

Further detail on the result of this experiment is provided in the next chapter of this thesis.

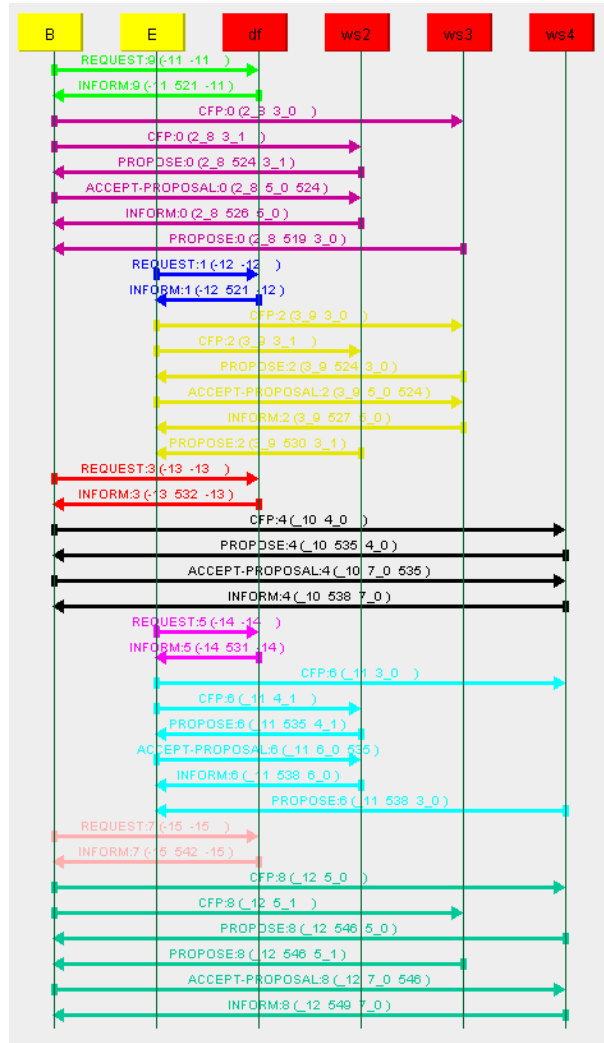


Figure 5.6: Exchanged messages with Job Agent "B" and "E".

Chapter 6

Results and Discussion

This chapter contains the results from the afore mentioned simulations and experiences performed along the course of this thesis. To begin with, results from the static simulations are presented and compared to other control approaches. Then, the results from the dynamic scenarios are presented, where, in two of them, the behaviour of the system can be compared to the one presented by other control approach in the benchmark. Finally, the results of the experimental tests performed are presented and commented.

Considering that the results are the operation of a FMS, and that it is of utmost importance to have a clear view of the workstation allocation along the manufacturing process, Gantt charts will be used to display the results of the experiences.

As a quantitative performance indicator used for comparisons, the global makespan (C_{max}) of the manufacturing process is considered.

6.1 Static Simulation

6.1.1 Static Scenario C0

The static scenario C0 was the first scenario to be tested and was considered as the reference performance of the system. Taking into account that it is composed by two orders (1 "BELT" and 1 "AIP"), this scenario consists of 7 jobs, with between 7 and 10 operations each, and 5 workstations. The Gantt chart for the performance of the system in this scenario and using rule base 1 is depicted in Figure 6.1

By analysis of the chart, the workstation B2 has a considerably higher utilization rate than workstations B3 and B4, with 19 operations performed versus 10 of the other workstations. This occurrence is part of the reasons for the creation and use of a second rule base. Furthermore, a clear sign of an allocation far from optimal is visible when Job "E" (in green) and Job "T" (in cyan) are loaded and, faced with both workstation B2 and B3 in the same state of operation, choose workstation B2, increasing the makespan of the global system.

Regarding second rule base, the Gantt chart of the operation of the system for scenario C0 with rule base 2 is presented in Figure 6.2. Here, a more balanced workstation utilization is evident, with the workstation B2 having performed 15 operations for 12 of the other two workstations. The overall

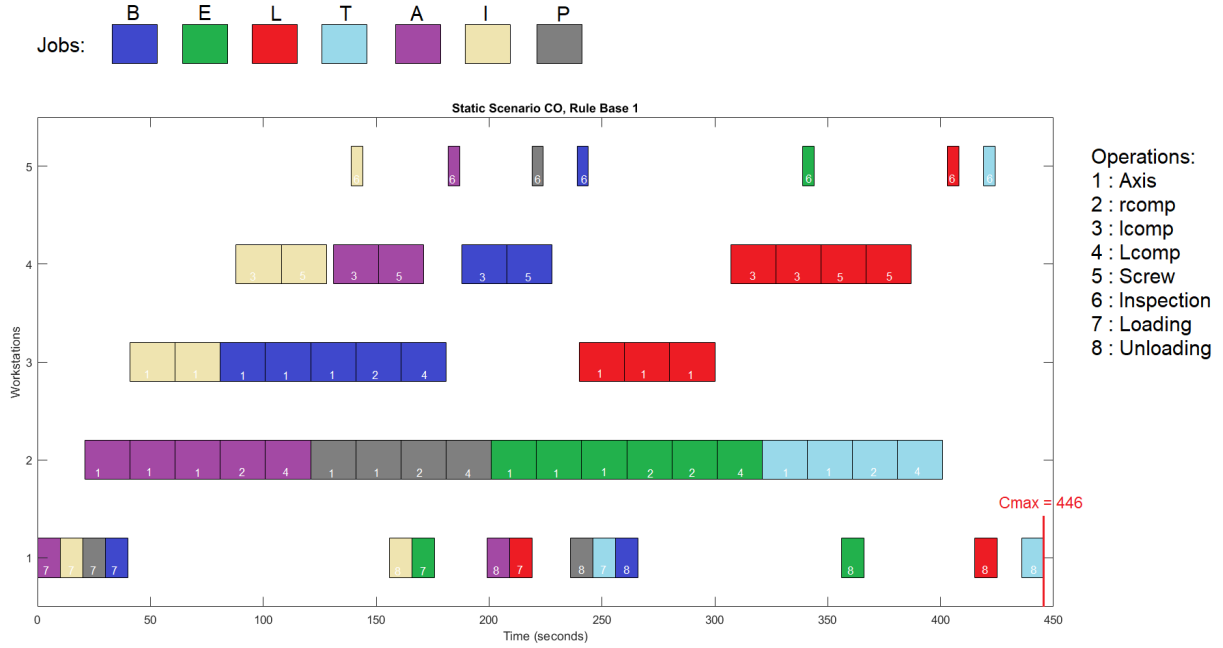


Figure 6.1: Gantt chart for Static Scenario C0 with rule base 1. ($C_{max} = 446s$)

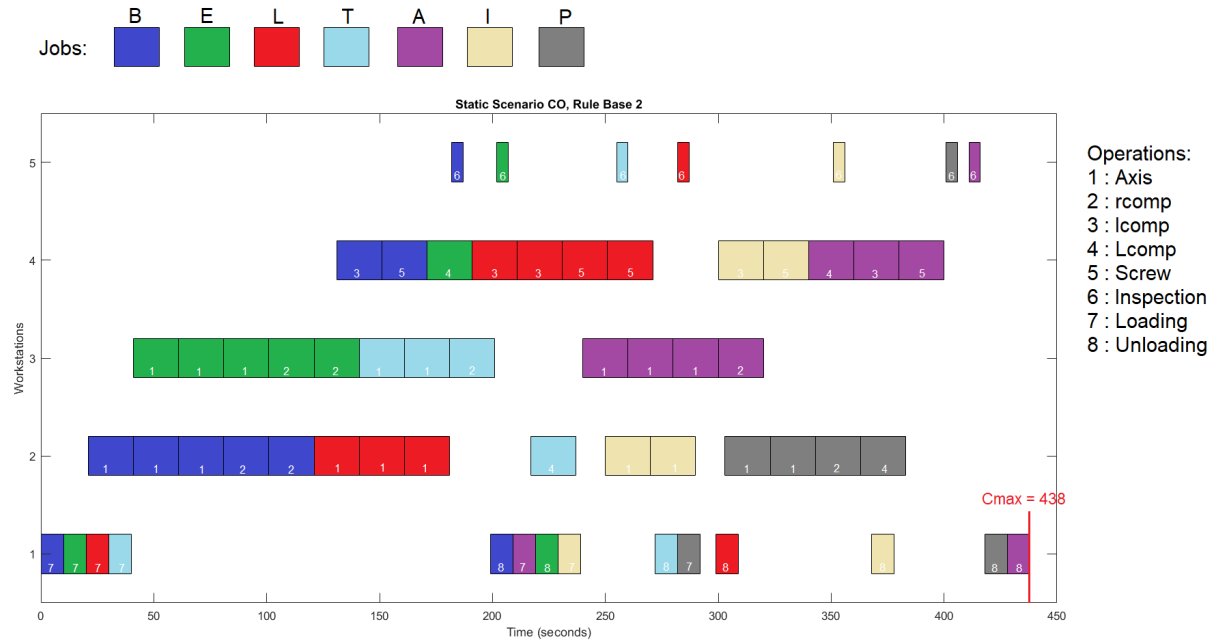


Figure 6.2: Gantt chart for Static Scenario C0 with rule base 2. ($C_{max} = 438s$)

performance of rule base 2 is better than rule base 1 ($C_{max} = 438s < C_{max} = 446s$), and no obvious sign of bad allocation in this configuration is visible.

Considering the differences of performance between the rule bases, the reference performance of the system was defined as scenario C0 with rule base 2. The operations performed on each workstation are displayed in Table 6.1.

In the table, a relatively good distribution of "Axis" and "rcomp" operations between B2 and B3 is

Table 6.1: Workstation utilization.

Workstation	Axis	rcomp	lcomp	Lcomp	Screw
B2	10	3	—	2	—
B3	8	4	—	—	0
B4	—	—	5	2	5

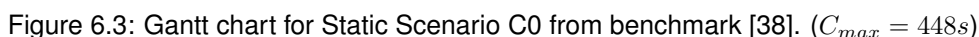
visible, and an even better division of the "Lcomp" operations happens between B2 and B4. In what concerns the operation "lcomp", which is an operation provided by B4 only, it was executed 5 times by B4 as expected. Lastly, operation "Screw" is the worse balanced operation, since it is executed 5 times by workstation B4 over 0 of B3. This can be easily explained with the fact that operation "Screw" always follows "lcomp" on the production sequence of the jobs, and considering that "lcomp" is always executed in B4 and the Job Agents were designed to request all the operations a workstation can provide before exiting it, the jobs that need operation "Screw" always have it done in B4 after "lcomp".

To assess the performance of this rule based scheduling system, some results introduced in [38], regarding simulation and real experiments using the potential fields approach in the AIP-PRIMECA FMS, were used for comparison. The potential fields approach [44], like the CNP, is also a reactive approach used in heterarchical control architectures. The first of these results comprises the performance of the potential fields approach for simulation of static scenario C0 and is presented in the Gantt chart of Figure 6.3.

Analysing the chart and comparing with scenario C0 with rule base 2, it is observable that both systems executed the exact same workstation allocation. Furthermore, the makespan of the benchmark is higher ($C_{max} = 448s$), which would mean an improvement percentage of approximately 2% of the proposed control system over the potential fields approach.

However, this difference in the makespan is caused by the not documented small time intervals between the unloading of a job and the loading of a new one, visible after the unloading of job "B" (in blue), job "E" (in purple) and job "T" (in cyan). In addition to not having their cause documented, these small time intervals are not present in other results of the same author, namely the Gantt chart with the solution of the MILP presented in Figure 6.5 or the Gantt chart with the result of the dynamic scenario #PS7 in the real AIP-PRIMECA presented in Figure 6.9.

For this reason, the performance of the proposed control system is considered validated, once it achieves a workstation allocation similar to the one of a demonstrated distributed control system, but the improvement percentage over the given system is not to be considered.



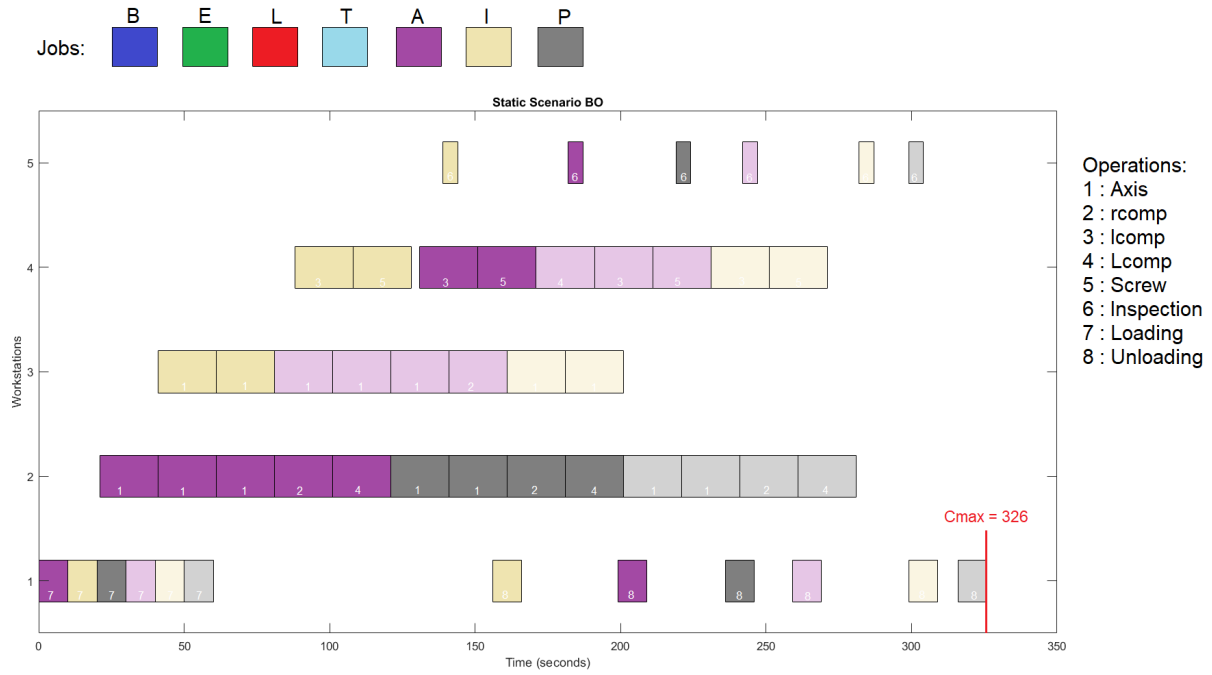


Figure 6.4: Gantt chart for Static Scenario B0. ($C_{max} = 326s$)

drawback that can be observed is the excess of utilization of workstation B2 compared to B3. If the last two jobs "I" and "P" switched order, that utilization would be more balanced and the global makespan would be inferior.

Table 6.2 establishes the comparison of the proposed control system with the above mentioned control approaches, using results introduced in [45]. The first system is once more the potential fields approach, this time applied to the scenario B0. Then, two results for the ORCA-FMS are presented, since this hybrid control architecture has two hybridization levels: (1) with the ILP (Integer Linear Programming) solver, only providing the job order (ORCA-FMS 1) and (2) providing also the machine sequence to be followed (ORCA-FMS 2).

Table 6.2: Makespan for several control approaches under B0 conditions.

Production	C_{max} (seconds)			
Scenario	Potential Fields	ORCA-FMS 1	ORCA-FMS 2	Proposed MAS
B0 (2×AIP)	345	323	314	326

By analysing the given results, the proposed MAS control system achieved a better performance in this scenario than the potential fields approach ($C_{max} = 326s < C_{max} = 345s$), with a reduction of makespan in the order of 5.5%. However, as expected, both hybridization levels of the ORCA architecture demonstrated better performances, due to the presence of the ILP optimization. The first level ORCA-FMS 1 achieved an almost similar performance, once the ILP only provided the job loading order and from there the system worked purely reactive. As to the second level ORCA-FMS 2, it achieved the best performance of all systems, with a reduction of 3.7% relatively to the proposed MAS control system. This can be easily understood, since with this hybrid control approach, the jobs not only were loaded in the

optimal order, but also respected the optimal machine sequence, just differing from the optimal solution in the arrival times.

6.1.3 Static Scenario with loading order from MILP

To conclude the static simulation section of this thesis, one last experiment was performed. Having as inspiration the operation of the hybrid manufacturing control approach mentioned in the last section and taking advantage of the MILP formulation and solution of the Flexible Job-Shop Scheduling Problem (FJSP) instantiated for the AIP-PRIMECA FMS introduced in [38], an hybrid control approach was simulated. This hybrid approach consisted in using the optimal job loading order provided by MILP as a starting point for the operation of the proposed reactive MAS control system.

Regarding the MILP, the parameters, variables, constraints and objective function are presented in Appendix B. As to the solution, it is depicted in the Gantt chart of Figure 6.5, and coincides with the optimal performance for static scenario C0. (In the chart, the operation "Lcomp" of job "P" is missing in machine M4 between the operation "Screw" of job "I" and the operation "Lcomp" of job "L".)

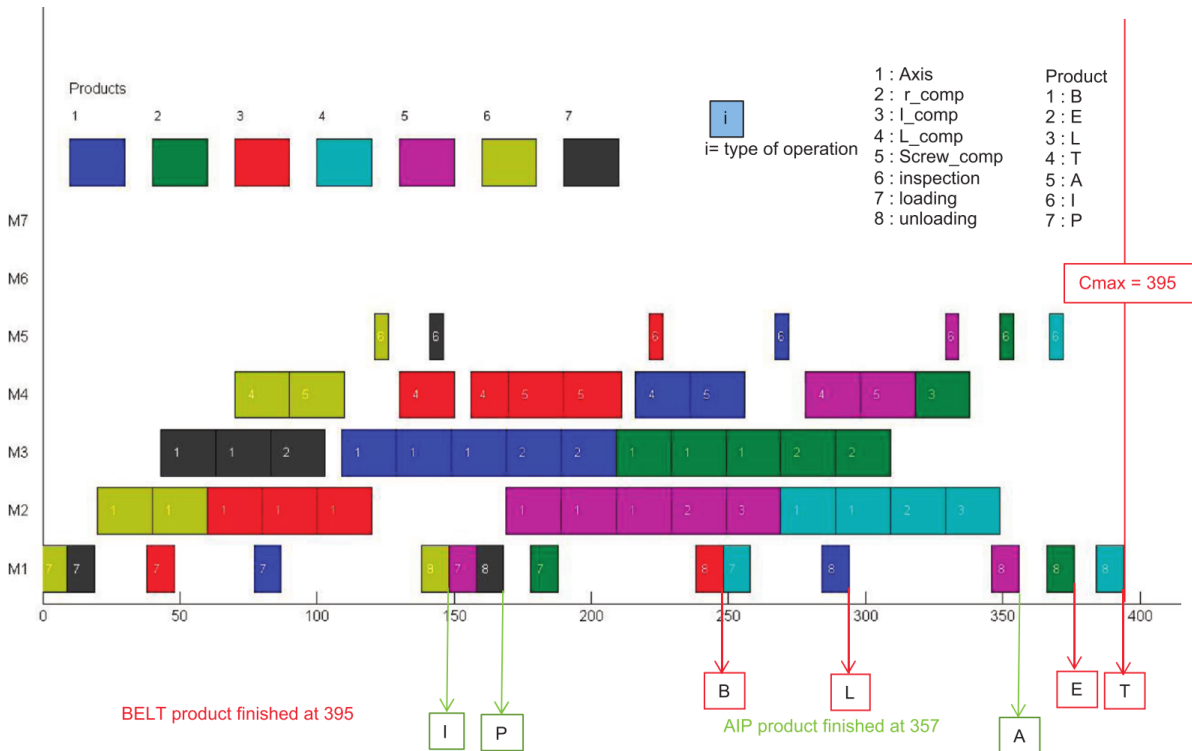


Figure 6.5: Gantt chart for Static Scenario C0 from MILP [38]. ($C_{max} = 395s$)

Thus, the optimal loading order "I", "P", "L", "B", "A", "E", "T" was inserted in the simulation and the proposed system acted according to its rule base but upon that initial data. The result of this hybrid experiment is presented in the Gantt chart of Figure 6.6.

To begin with, a difference can be noticed in the loading of the jobs: while according to the proposed rule bases the jobs should be loaded whenever the system can receive them, which results in the first 4 jobs being loaded continuously, in the optimal plan the first two jobs are loaded immediately but then

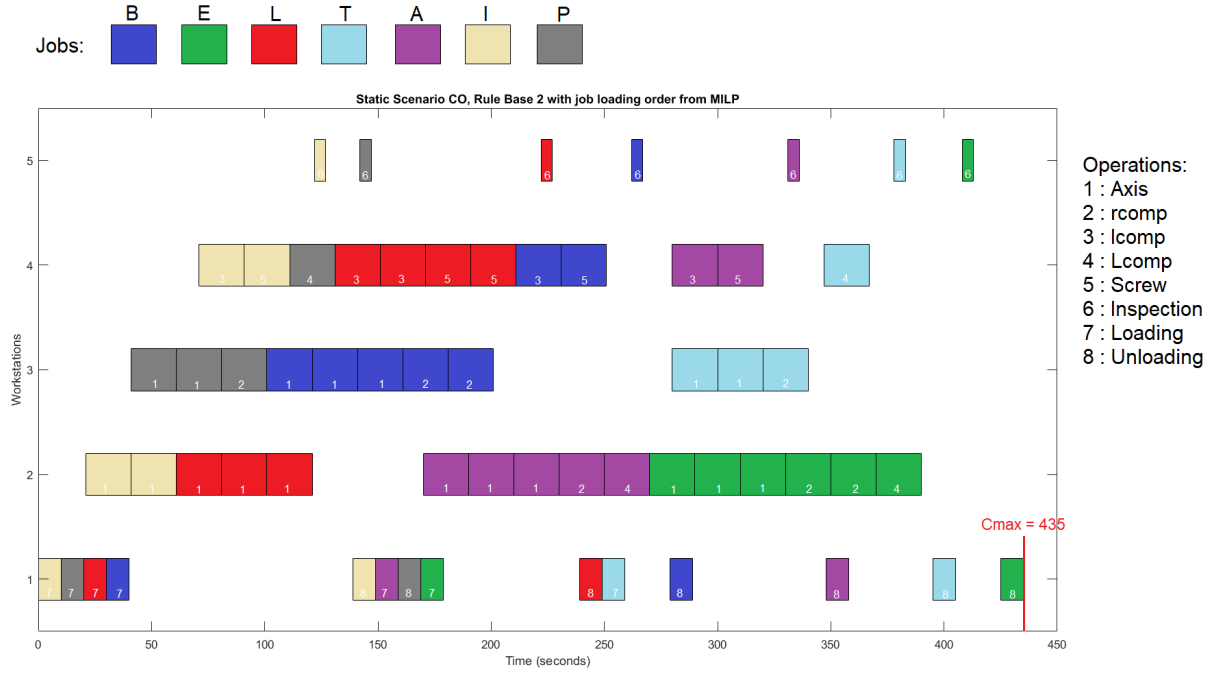


Figure 6.6: Gantt chart for Static Scenario C0 with loading order from MILP. ($C_{max} = 435s$)

the third and fourth jobs are only loaded with the exact time to enter the system and start receiving operations, which makes sense with a view to minimize the Work-In-Progress (WIP). Then, the major difference between the performance of the system and the optimal operation can be observed when job "E" is loaded: after entering the system and receiving proposals from workstation B2 and B3, which were both currently operating and in the same state, the job rules make him choose B2, while in an optimal solution that job would start its manufacturing in B3.

This difference of operation results in an increase of approximately 10% in the global makespan of the hybrid system compared to the optimal solution ($C_{max} = 435s > C_{max} = 395s$).

This comparison is a clear example of the difference between an *a priori* optimization, where data from the entire system operation is known initially for the scheduling, and a reactive agent-based system, where the local decision making of the agents generate schedule solutions differing from the optimal.

6.2 Dynamic Simulation

The dynamic study of this work consisted in testing the proposed control system under the conditions of 4 selected dynamic scenarios explained in Section 4.3.

In order to execute these experiments in the simulation, the perturbations were artificially set to happen during the course of the simulation. However, none of those was initially known by any of the agents, which continued acting reactively and making their decisions on-line.

All dynamic scenarios were tested under the same conditions of reference scenario C0 with the rule base 2, so that the effects of perturbations can be quantified relatively to the reference performance of the system.

Furthermore, in some of the experiences a comparison to the performance of the potential fields approach tested under those dynamic conditions in the AIP-PRIMECA FMS was conducted.

6.2.1 Dynamic Scenario #PS5

The dynamic scenario tests the ability of the system in handling the introduction of a rush order.

Differently from how it was defined in [38] (Table 4.3), a parameter was changed in the course of this simulation to better exhibit the response of the control system. The change was in the arrival time of the rush order, which was defined in the table for after the end of the production of the fourth job in the cell. The reason was the fact that scenario C0 has 7 jobs to be produced, which means that after the end of production of the fourth job there is no more jobs waiting to be loaded in the manufacturing system. Thus, the rush order would be just normally loaded in the system and the response of the control system would not be clear. To avoid this, and to show that the control system has the ability of prioritizing rush orders against normal orders, the arrival time of the rush order was set to the end of production of the second job in the system.

Figure 6.7 shows the response of the system to the introduction of the rush order. That order is submitted immediately after the end of the production of the job "E" (in green) at $t = 229s$ and it can be observed that in spite of loading the rest of the jobs that were already waiting to be loaded in the cell (i.e. job "I" and job "P" of order #2), the next jobs loaded were the "A", "I" and "P" of the rush order (in faded purple, beige and gray).

The global makespan of the production increased from 438 to 611 with the production of three more jobs, which translates to a raise of 39.5% in the makespan.

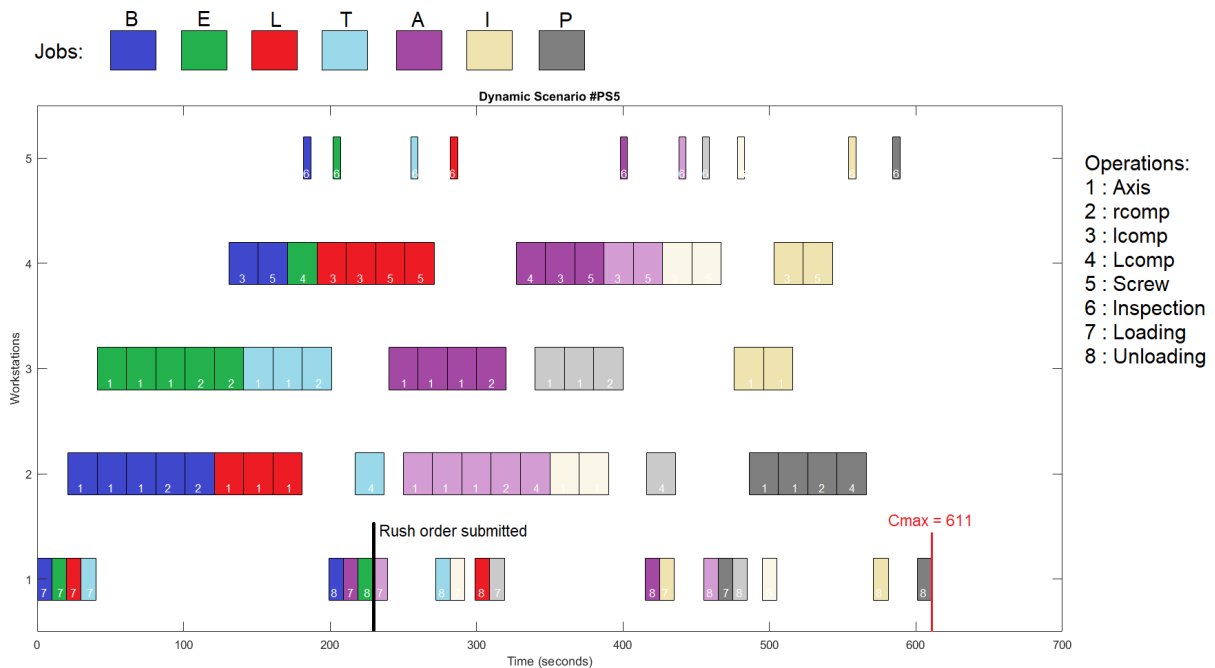


Figure 6.7: Gantt chart for Dynamic Scenario #PS5. ($C_{max} = 611s$)

6.2.2 Dynamic Scenario #PS7

Scenario #PS7 is a simple dynamic scenario that simulates a maintenance operation in one part of the conveyor system, indicated in Figure 2.3.

In the context of the IAL Manufacturing System and the proposed control system, this routing change is only visible in the difference of transportations times. Even so, it is important to test if that routing change is possible in the case of the simulated IAL Manufacturing System and even in the remote scenario of an implementation in the real benchmark cell, without modifying the control system.

The maintenance was simulated as a temporary change in the transportation times table, changing to zero the duration of the shortest path between workstation B5 and the conveyor belt (machines M5 and M1 in the AIP-PRIMECA), which resulted in the agents deciding for the second path using an alternative and longer route.

The Gantt chart for the dynamic scenario #PS7 is depicted in Figure 6.8, where the maintenance in the conveyor system started after the production of job "L" (in red) at $t = 309s$. The visible effects of that route change are the delays in the travels between workstation B5 and the conveyor belt. Without perturbations that path would take 12 seconds, while going for the second alternative path takes 28 seconds. The reference arrival times at the conveyor belt are represented by the dashed lines in the Gantt chart and the 16 seconds difference is visible in the chart and in the resulting makespan of $C_{max} = 454s$.

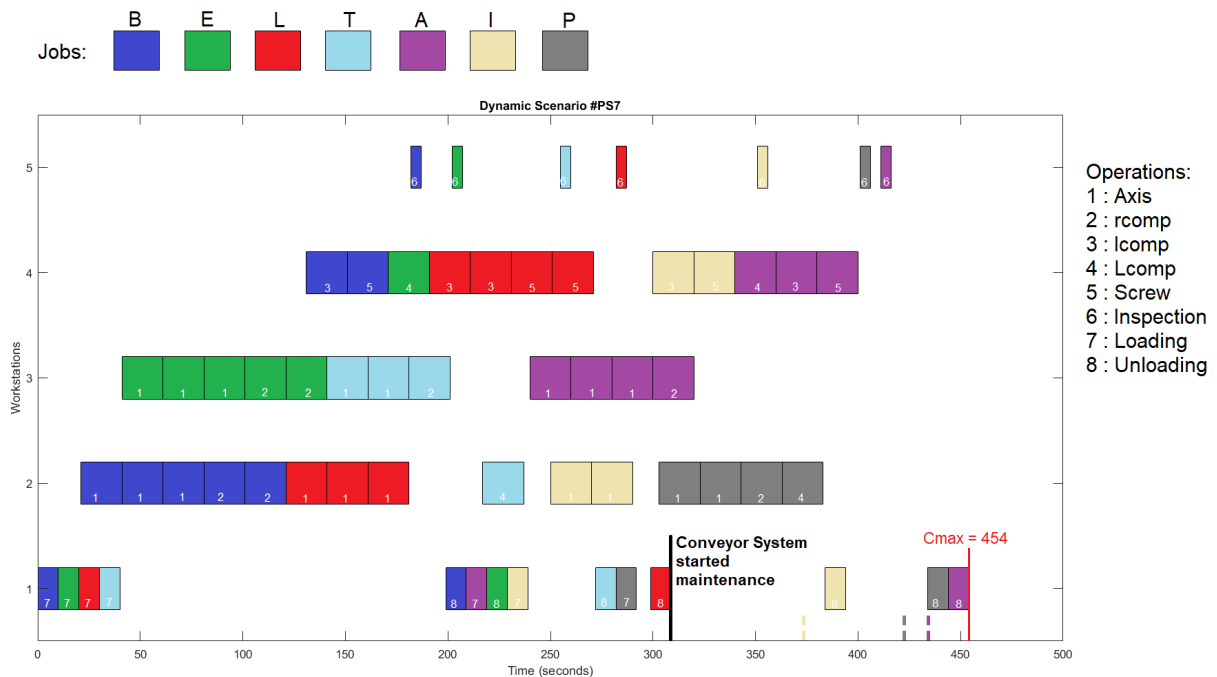


Figure 6.8: Gantt chart for Dynamic Scenario #PS7. ($C_{max} = 454s$)

The behaviour of the system can be compared to the behaviour of the potential fields approach in a real experiment in the AIP-PRIMECA FMS. Figure 6.9 shows the Gantt chart for the dynamic scenario #PS7 tested in the physical cell. Unfortunately, the fact that this result comes from the real cell makes it unsuitable for comparison of performances, since it considers certain waiting times that are neglected in

the simulations (examples of these waiting times are the time a shuttle needs to move from the job input storage area to the operation area and the time a shuttle needs to reach the main loop of the conveyor system before the machine start operating on another job).

However, a parallelism can be established between both performances, since the delays caused by the maintenance operation are of the same order. In the real cell without perturbations, the time a job would need to go from machine M5 to machine M1 is near 18 seconds, while in this situation the jobs need almost 35 seconds for the same path.

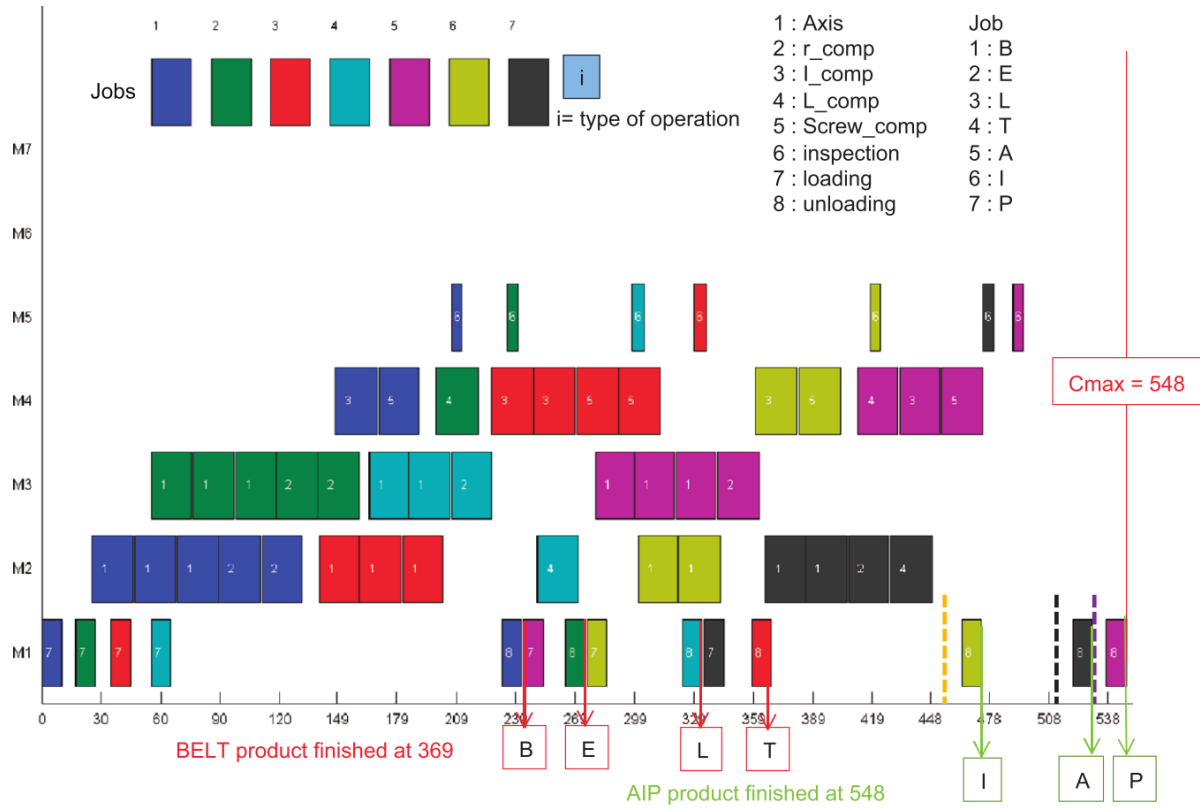


Figure 6.9: Gantt chart for Dynamic Scenario #PS7 from benchmark [38]. ($C_{max} = 548s$)

6.2.3 Dynamic Scenario #PS9

Dynamic scenario #PS9 tests a very usual situation of a breakdown in one workstation, which in this specific case corresponds to a redundant one.

In terms of simulation, workstation B2 was programmed to go down after the departure of the first job and stay down for $25 \times 7 = 175s$. With the workstation in breakdown, the Workstation Agent refuses all CFP and Requests messages. Due to this situation, the jobs must wait in the closed-loop conveyor system for a place in the other available workstations. Given that workstation B2 only executes redundant operations, no job needs to stop its manufacture, only a delay in the production is expected.

The performance of the control system under this breakdown situation is presented in Figure 6.10. The breakdown of B2 happened after the execution of 5 operations on job "B", at $t = 121s$. Due to that unavailability, the global makespan increased to 486s, about 11% higher than the reference scenario.

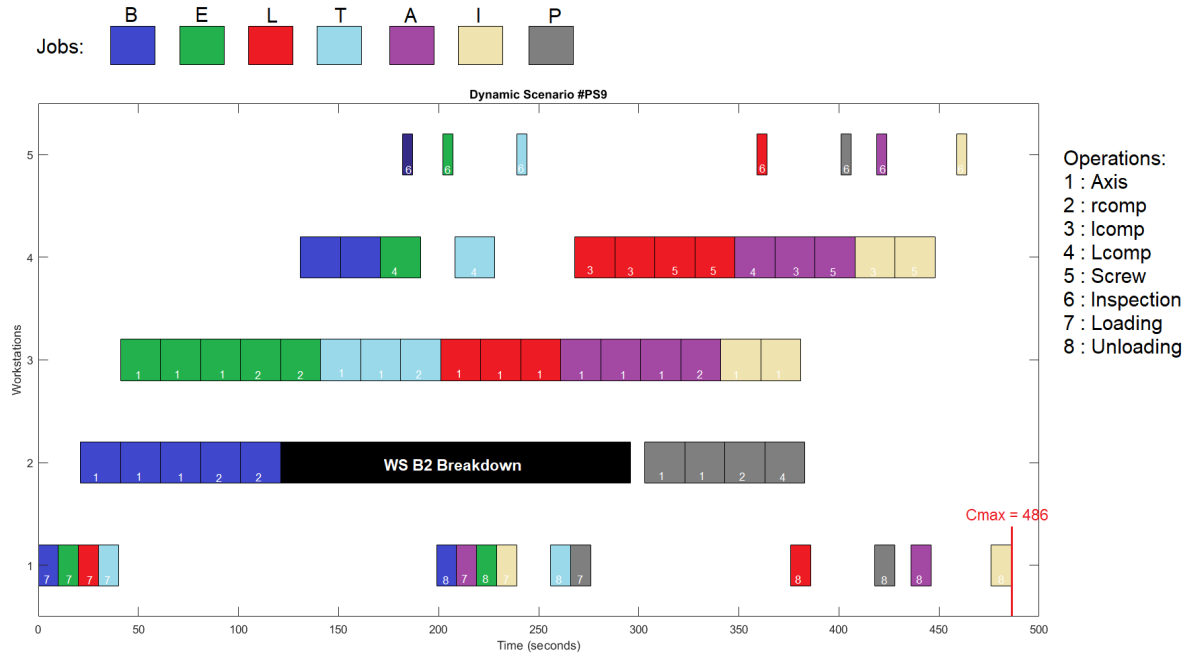


Figure 6.10: Gantt chart for Dynamic Scenario #PS9. ($C_{max} = 486s$)

In the chart, it can be observed that job "L" (in red) was in the job input storage area of workstation B2 and due to the breakdown had to exit the workstation and wait for a place to complete the same operations in workstation B3. Furthermore, both jobs "A" (in purple) and "I" (in beige) also went to workstation B3 during the downtime of B2. When the workstation turned available again, job "P" (in gray) was currently waiting in the closed-loop conveyor system for an available workstation and immediately began its travel towards workstation B2, arriving there at $t = 303s$.

Similarly to what was introduced for the simulation of reference scenario C0, a Gantt chart with the simulation performance of the potential fields approach under the conditions of scenario #PS9 in the AIP-PRIMECA is also available in [38].

The performance is portrayed in Figure 6.11, from where it can be concluded that the proposed control system had a similar behaviour to the potential fields control approach. This fact is important to, once more, validate the operation of the designed agent-based manufacturing control system.

In terms of makespan, the proposed system achieved a global makespan 5 seconds inferior ($C_{max} = 486 < C_{max} = 491$). Nevertheless, like it was said for the comparison of scenarios C0, the performances are considered similar since the difference is again justified with the undocumented time intervals between the unloading of a job and the immediate loading of the next one.

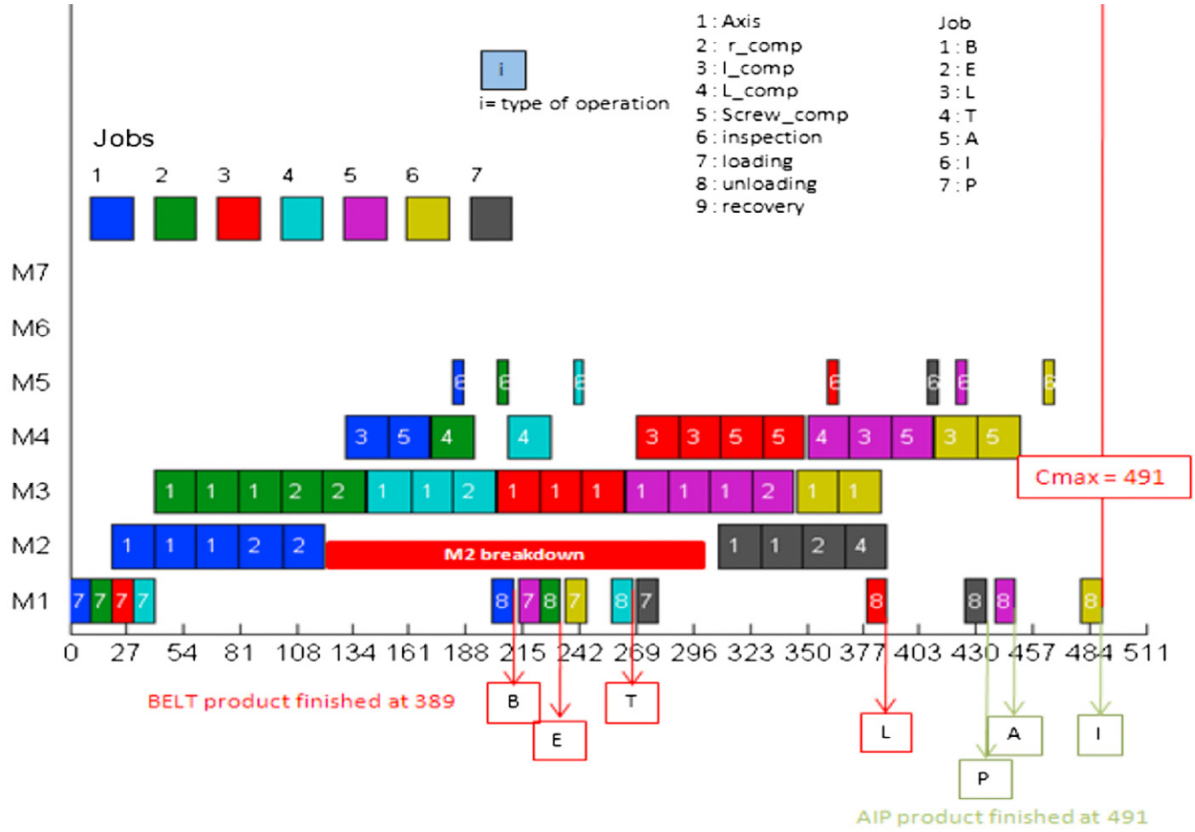


Figure 6.11: Gantt chart for Dynamic Scenario #PS9 from benchmark [38]. ($C_{max} = 491s$)

6.2.4 Dynamic Scenario #PS10

The last dynamic experience conducted is an extension of scenario #PS9. Dynamic scenario #PS10 also comprises a breakdown situation, but, in this case, of one critical workstation. The temporary unavailability of a critical workstation, which will cause leaving jobs with no other option for the execution of a certain operation, is expected to have a considerable higher impact on the system performance than the previous scenario.

For simulation purposes, workstation B4 was set to break down after the departure of the second job at $t = 191s$ and stay down for 175s like in scenario #PS9. The Gantt chart for scenario #PS10 is depicted in Figure 6.12.

The first two jobs, job "B" (in blue) and job "E" (in green), managed to complete the operations in workstation B4 and thus completing their production sequence. As to job "L" (in red), it already found workstation B4 unavailable and had to wait in the closed-loop conveyor system for it to become available and propose again. Due to the breakdown, job "L", job "A" and job "I" had their production delayed and all waited for the return to operation of B4 to complete their production steps.

The resulting global makespan was $C_{max} = 578s$, which represents an increase of 32% in respect to the reference scenario C0. As expected, the impact of a breakdown in one of the critical workstations caused a considerably higher impact in the makespan of the system than the breakdown in one redundant one, leading to an increase of 32% in the global makespan against just 11% of the redundant workstation breakdown.

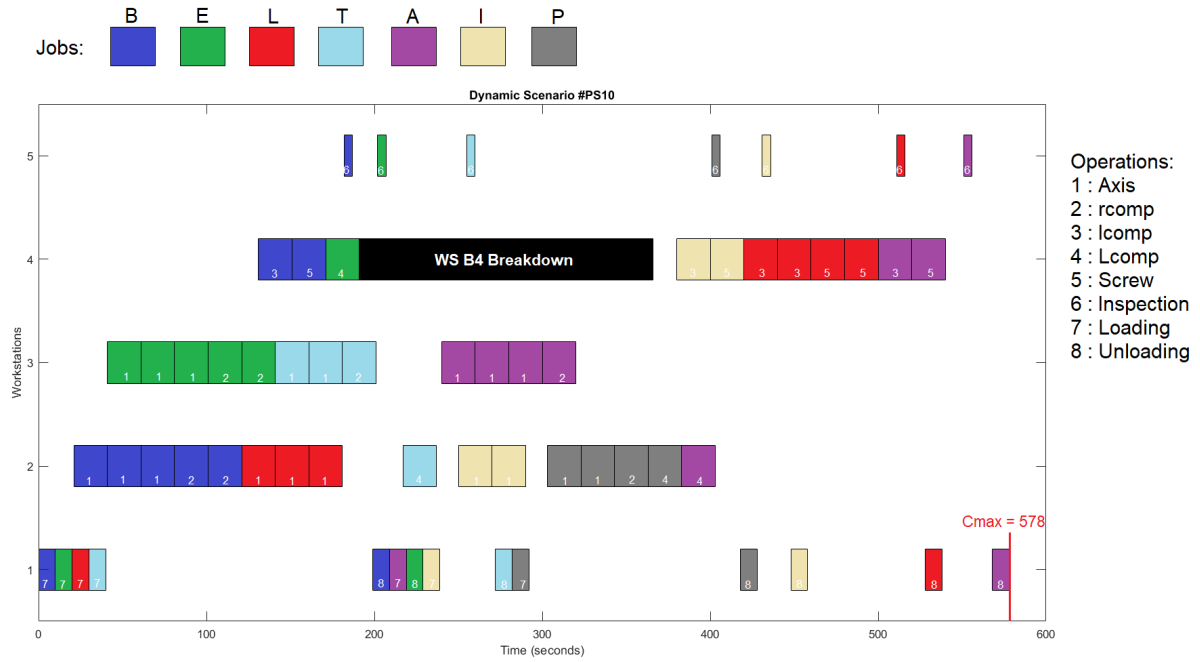


Figure 6.12: Gantt chart for Dynamic Scenario #PS10. ($C_{max} = 578s$)

6.3 Experimental implementations

In this section, results from the experimental tests described in 6.3 are presented and discussed. Both experiments were performed in computers from the IAL at Instituto Superior Técnico, which are Intel Core i7-4790 CPU @ 3.60GHz with 8,00 GB of RAM and were running Windows 10.

6.3.1 Connection to PLC

The Hardware integration experience consisted in running the simulation with Workstation Agent B2 reading and writing from a data file, so that by action of an intermediary MATLAB script also reading and writing from the same file and then updating the PLC variables, a real time connection between a simulated agent and a PLC could be established. The realization of this experience aimed at showing that is possible, and relatively easy, to integrate hardware with the agent environment, even if the agent platform is not the best suited for real time operation.

The performance of the simulation was expected to be similar to the reference performance without hardware integration, which in this case was the static scenario C0 with rule base 1. However, during the experience, it could be noticed that the simulation time was running slower than the internal clock of the PLC. The cause for this discrepancy is the lack of enough processing power to run the simulation in real world time, and so, simulation time was running slower than PLC time. The result was a simulation with makespan inferior to the reference performance, since in simulation time the operations executed by the PLC in workstation B2 were faster than the operations executed on the other simulated workstations. The Gantt chart exhibiting this discrepancy is presented in Figure 6.13.

In practical terms, the simulation model in the GAMA software was programmed with a simulation

step of 0.01s and in this experience was set to run in real world time, which means with a step duration of 0.01s. Due to the lack of processing power, the step duration in world time was in average 0.016s instead of 0.01s. This means that the 20s which the PLC would take to perform an operation in real world time, in simulation time were closely around 12.5s.

In the Gantt chart of Figure 6.13, the duration of the operations in workstation B2 is inferior to the 20s taken by workstations B3 and B4 to execute the mounting tasks, which results in some differences in terms of arrival times to the workstations in the comparison with the reference simulation. The main differences are the exchanges in the "Inspection" and "Unloading" order between jobs "A" (in purple) and "P" (in gray) and later between jobs "L" (in red) and "T" (in cyan). The global makespan is $C_{max} = 402s$.

In order to evaluate if, in case the simulation was running at the same real world time of the PLC, the performance of the system with the PLC integration would be the same as the reference simulation, a second experience was realized. The timers in the PLC program were adjusted with a factor of $f = \frac{0.016-0.010}{0.010} \times 100 = 60\%$, going from 20 to 32, and the simulation was run again with a view to having the PLC working in simulation time. The result of this experiment is depicted in the Gantt chart of Figure 6.14.

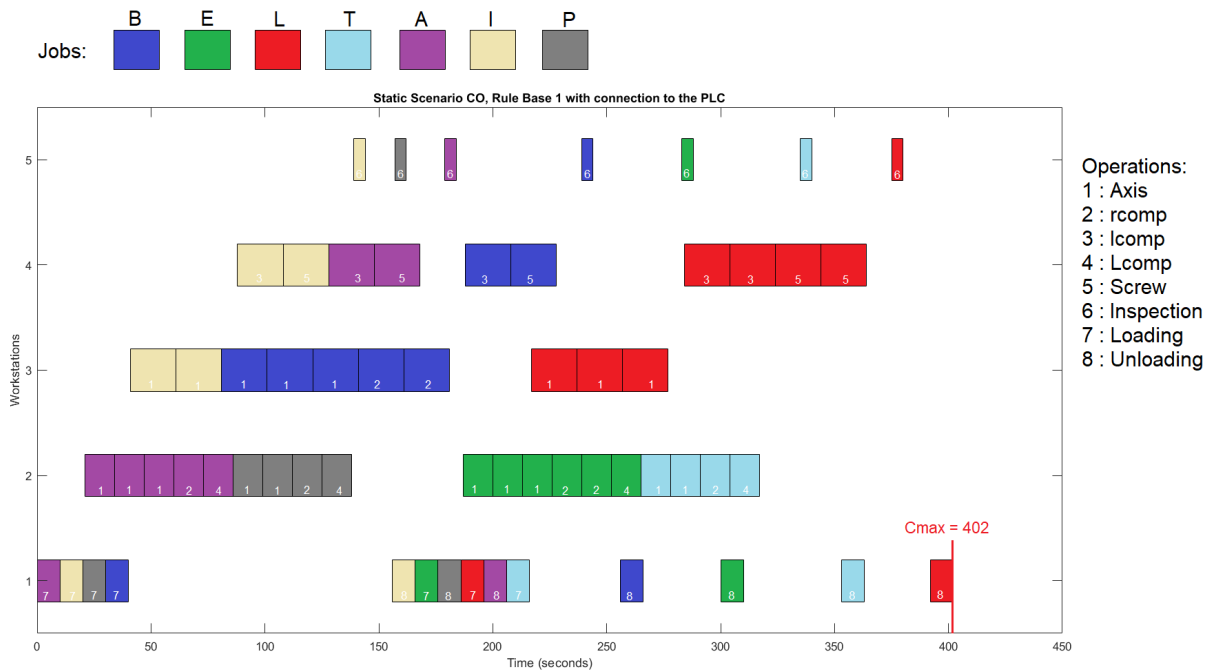


Figure 6.13: Gantt chart for Static Scenario C0 in connection to the PLC (1). ($C_{max} = 402s$)

As it can be seen, once the PLC operations were adjusted to the simulation time, the scheduling was similar to the one presented in the simulation of scenario C0 with rule base 1 (Figure 6.1). This shows that is possible to integrate hardware and achieve the same workstation allocation as in the pure simulation. Nevertheless, no conclusions can be drawn regarding the communications induced delays, since no in-depth study to assess those delays was conducted.

Contrary to what happens in all the simulations presented so far, whenever this experiment is executed, the global makespan value has a small deviation in the order of the second compared to the previous value. This is caused by the variations in the running time of the simulation, which only high-

lights the limitations of the simulation software for real time experiments. In what concerns real time agent-based experiments, a robust and proper agent platform such as JADE is necessary.

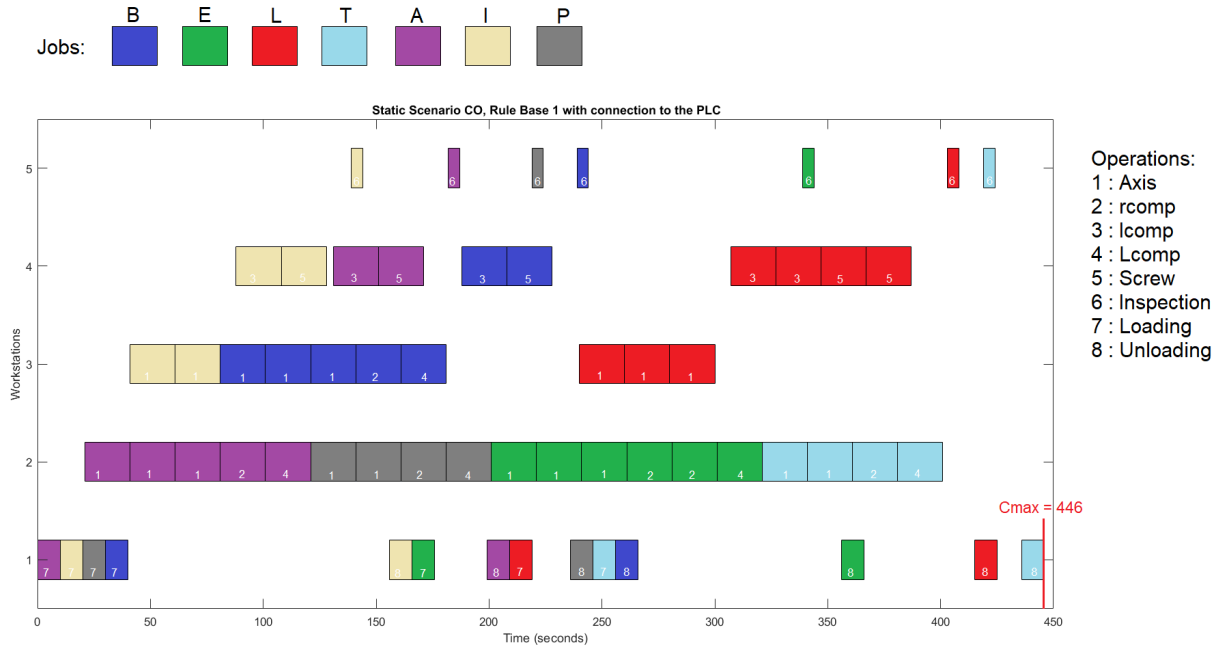


Figure 6.14: Gantt chart for Static Scenario C0 in connection to the PLC (2). ($C_{max} = 446s$)

6.3.2 JADE Example

The last experiment to be presented is the simple CNP example implemented in JADE. This illustrative case was described in section 5.3, where interaction diagrams with the exchanged messages from JADE were displayed.

The example only comprises the three workstations that execute mounting operations (WS2, WS3 and WS4) and two jobs "B" and "E". First, JADE was launched with the creation of the Workstation Agents and the Job Agent "B" only. As soon as the platform was launched, the Job Agent immediately started production. After 10 seconds, Job Agent "E" was launched to begin its production sequence too.

The message exchange in the JADE Platform is close to instantaneous and as expected, the operations were executed in real time with the absence of delays. The Gantt chart with the result of this simple example is presented in Figure 6.15.

Considering that no transportation times were considered in this example, the global makespan $C_{max} = 140s$ corresponds to the expected, since it is the sum of the duration of 7 operations of job "B". The fact that job "B" starts its production in workstation B3 is explained by the Job Agent accepting the first Propose that it receives, which in JADE happen to be workstation B3 instead of B2.

More than the result itself, it is important to highlight the ability of JADE to allow easy and fast development of agent-based applications to work efficiently in real time and distributed across different machines.

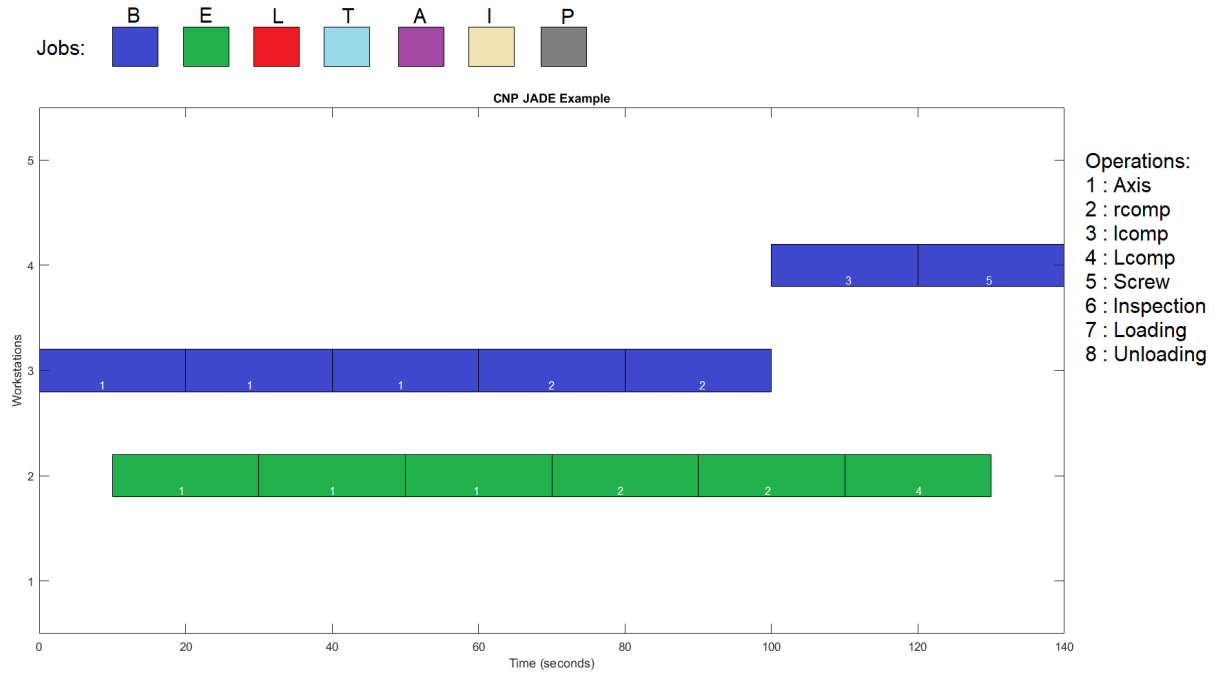


Figure 6.15: Gantt chart for the CNP example in JADE. ($C_{max} = 140s$)

Table 6.3 summarizes all the simulations performed with the corresponding quantitative performance indicator C_{max}). For the cases where comparisons with other approaches were possible, those results are also presented.

Table 6.3: Summary of all the simulations performed.

Simulation	$C_{max}(s)$	Comparison	$C_{max}(s)$
Static Scenario C0, rule base 1	446		
Static Scenario C0, rule base 2	438	Potential Fields Approach	448
Static Scenario B0, rule base 1 and 2	326	Potential Fields Approach	345
		ORCA-FMS 1	323
		ORCA-FMS 2	314
Static Scenario C0, loading order from MILP	435	MILP	395
Dynamic Scenario #PS5	611		
Dynamic Scenario #PS7	454	Potential Fields Approach (real experiment)	548
Dynamic Scenario #PS9	486	Potential Fields Approach	491
Dynamic Scenario #PS10	578		
Static Scenario C0, connection to PLC	402		
Static Scenario C0, connection to PLC (adjusted times)	446		

Chapter 7

Conclusions

The objective of this thesis was to study the applicability of agent-based manufacturing control as a production control system designed to achieve the smart, flexible and reconfigurable factory. To achieve this objective, a Multi-Agent System was designed to demonstrate a flexible production system and a simulation study was conducted to test the system through static and dynamic scenarios. Moreover, the preliminary steps into an experimental implementation were taken.

In the first place, a FMS designated IAL Manufacturing System was planned, and its operation was modelled in accordance with the operation of a known benchmark for research in the field of distributed control systems, the AIP-PRIMECA FMS. Having defined the production system to be controlled analogously to a benchmark cell, the results of some experiments conducted in the cell and reported in the literature were used for direct comparison and validation.

In the design phase, a MAS composed of 5 types of reactive agents (Order Agent, Job Agent, Workstation Agent, Robot Agent and Conveyor Agent) was introduced and the agents were modelled using Petri nets. Agent interactions to allow execution of tasks in the system and workstation allocation were defined through the combination of 3 FIPA Interaction Protocols (FIPA-Request, FIPA-Propose and FIPA-Contract Net). Furthermore, two decision rule bases were tested to govern the global operation of the system and the decision-making in the assignment of operations.

In what concerns the simulation study, two static scenarios (B0 and C0) were tested in the static part and four dynamic scenarios (#PS5, #PS7, #PS9 and #PS10) were tested in the dynamic part. In addition, a simulation where the jobs were loaded with order given by MILP was also performed as a hybrid control approach.

The first static simulation was the reference scenario C0, which was tested for both rule bases. The system achieved better performance for rule base 2 than for rule base 1, which can be explained by the better workstation utilization balance and the different loading order. In order to validate the behaviour of the designed control system, a comparison was established with the performance of another reactive distributed control architecture, the Potential Fields approach, through simulation in the benchmark FMS. Both systems presented a similar behaviour and thus, the proposed control system operation was validated.

Apart from scenario C0, static scenario B0 was also tested with the purpose of establishing comparisons with the performance of other distributed control systems under different scenarios. For scenario B0, the proposed control system achieved a better global performance than the Potential Fields approach and, as expected, had a worse performance than the hybrid manufacturing control architecture ORCA. Nevertheless, it showed very promising results, considering that ORCA uses an optimization algorithm.

The static simulation was concluded with the simulation of an hybrid control approach that allowed to compare the reactive on-line performance of the agents with the optimized schedule given from an *a priori* optimization algorithm. The job loading order given by MILP was used in the simulation of the reference scenario C0 with rule base 2 and its performance was compared to the optimal scheduling. Despite having the optimal job loading order, the performance of the hybrid approach was still far from optimal due to its on-line decision making along the production.

The system successfully responded to all dynamic scenarios. In #PS5, the rush order was prioritized over the other orders and its jobs were immediately loaded in the system. In scenarios #PS7 and #PS9, the performance of the proposed system was also like the expected and a comparison of behaviour with the Potential Fields approach through simulation or experimentation in the AIP-PRIMECA was possible, resulting in similar workstation allocation. As to scenario #PS10, similarly to #PS9 the system managed to continue the production through the breakdown situation. However, it showed a clearly higher impact in the global performance due to the fact that the breakdown happened in one critical workstation.

Regarding the experimental tests, a successful connection was established between one simulated agent and one PLC, disclosing the way in terms of hardware integration. However, no conclusion could be drawn concerning the real time communication induced delays, since the IAL computer did not present enough processing power to run the simulation in real time. As to the JADE example, the way in which the agents would be implemented and would operate in real time was successfully illustrated.

After testing the designed agent-based control system under static and dynamic scenarios, the goals of this thesis can be considered accomplished. The control system presented a reactive behaviour performance comparable with other distributed control architectures introduced in the literature. Yet, a lack of long term vision over the behaviour of the distributed MAS control architecture can be observed throughout the simulations in the local decision making of the agents, resulting in workstation allocations far from optimal. This is a drawback of this type of implementation of agent-based systems for manufacturing control: the lack of horizon of the agents driven from the distributed local decision making.

An agent-based system is a promising solution for the control of flexible and reconfigurable systems that experience dynamic environments and require a very reactive behaviour. However, the optimal solution of an *a priori* optimization can always be considered as a way of reducing the lack of horizon and improving agent decision-making. Reactive and optimized hybrid control architectures are likely to be composing the control systems for the factories of the future.

7.1 Future Work

Considering the introductory character of this work in the context of agent-based manufacturing control approaches, future developments can be done in several topics addressed along the thesis.

To begin with, the complete implementation of the simulated agents in the JADE Platform should be performed. Together with the implementation, all the possible required restructuring or completion should be executed, so that the agents are implemented in a way that the control system can be successfully used in a real system.

A direct connection between JADE and industrial devices would be the next step. Communication protocols should be explored so that hardware integration is implemented as direct and efficient as possible.

In order to provide the agents with more information about the global system and reduce their lack of horizon, an ontology can be created to allow more complex interactions between agents and more exchanged information between workstations and jobs. It would be interesting to develop decision making algorithms to take advantage of the received system information and improve agents intelligence and the quality of autonomous decisions.

Lastly, the design of an hybrid system combining a reactive MAS and an optimization algorithm like MILP would be of great interest, allying an optimal static performance with the ability to react in case of dynamic perturbations. The design of this hybrid system could be accomplished, for instance, by means of the creation a supervisor agent in the MAS that would execute MILP and advise other agents to act in accordance with its solution.

Bibliography

- [1] S. Wang, J. Wan, D. Li, and C. Zhang. Implementing Smart Factory of Industrie 4.0: An Outlook. *International Journal of Distributed Sensor Networks*, 2016. doi: 10.1155/2016/3159805.
- [2] P. Leitão, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo. Smart Agents in Industrial Cyber-Physical Systems. In *Proceedings of the IEEE*, volume 104, pages 1086–1101, 2016. doi: 10.1109/JPROC.2016.2521931.
- [3] P. Leitão, A. W. Colombo, and S. Karnouskos. Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*, 81:11–25, 2016. ISSN 01663615. doi: 10.1016/j.compind.2015.08.004.
- [4] P. Leitão. Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7):979–991, 2009. doi: 10.1016/j.engappai.2008.09.005.
- [5] A. Colombo. Industrial agents: towards collaborative production -automation, -management and -organization. *IEEE Industrial Electronics Society Newsletter*, 52((4)):17–18, 2005.
- [6] P. Leitão, V. Marik, and P. Vrba. Past, Present, and Future of Industrial Agent Applications. *IEEE Transactions on Industrial Informatics*, 9(4):2360–2372, nov 2013. doi: 10.1109/TII.2012.2222034.
- [7] J. Zhang, X. Yao, J. Zhou, J. Jiang, and X. Chen. Self-Organizing Manufacturing: Current Status and Prospect for Industry 4.0. In *Proceedings - 2017 5th International Conference on Enterprise Systems: Industrial Digitalization by Enterprise Systems, ES 2017*, pages 319–326, 2017. doi: 10.1109/ES.2017.59.
- [8] P. Leitao, S. Karnouskos, L. Ribeiro, P. Moutis, J. Barbosa, and T. I. Strasser. Common practices for integrating industrial agents and low level automation functions. In *Proceedings IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, volume 2017-Janua, pages 6665–6670, 2017. doi: 10.1109/IECON.2017.8217164.
- [9] J. Xie and C.-C. Liu. Multi-agent systems and their applications. *Journal of International Council on Electrical Engineering*, 7(1):188–197, 2017. doi: 10.1080/22348972.2017.1348890.
- [10] FIPA Web Site: <http://www.fipa.org/>, accessed on March 14, 2018.

- [11] B. Bauer, J. P. Müller, and J. Odell. Agent UML: A formalism for specifying multiagent software systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1957 LNCS(3):91–103, 2001. doi: 10.1007/3-540-44564-1_6.
- [12] R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, C-30(5):372, 1980. doi: 10.1109/TC.1981.1675799.
- [13] A. Otto and K. Hellmann. IEC 61131: A general overview and emerging trends. *IEEE Industrial Electronics Magazine*, 3(4):27–31, 2009. doi: 10.1109/MIE.2009.934793.
- [14] V. Vyatkin. IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review. *IEEE Transactions on Industrial Informatics*, 7(4):768–781, 2011. doi: 10.1109/TII.2011.2166785.
- [15] T. R.-S. M. Sánchez-Marrè. A review on Multi-Agent Platforms and Environmental Decision Support Systems Simulation Tools. 2006.
- [16] K. Kravari and N. Bassiliades. A survey of agent platforms. *JASSS*, 18(1):1–19, 2015. doi: 10.18564/jasss.2661.
- [17] S. Abar, G. K. Theodoropoulos, P. Lemarinier, and G. M. O'Hare. Agent Based Modelling and Simulation tools: A review of the state-of-art software. *Computer Science Review*, 24:13–33, 2017. ISSN 15740137. doi: 10.1016/j.cosrev.2017.03.001.
- [18] F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent with JADE Systems*. Wiley, 2007. doi: 10.1002/9780470058411.
- [19] L. Monostori, J. Vancza, and S. R. T. Kumara. Agent-based systems for manufacturing. *CIRP Annals - Manufacturing Technology*, 55(2):697–720, jan 2006. doi: 10.1016/j.cirp.2006.10.004.
- [20] V. Mařík and J. Lažanský. Industrial applications of agent technologies. *Control Engineering Practice*, 15(11):1364–1380, 2007. doi: 10.1016/j.conengprac.2006.10.001.
- [21] P. Leitão and P. Vrba. Recent Developments and Future Trends of Industrial Agents. *HoloMAS 2011*, LNAI 6867:15–28, 2011. doi: 10.1007/978-3-642-59723-7_11.
- [22] V. Mařík, P. Vrba, K. H. Hall, and F. P. Maturana. Rockwell automation agents for manufacturing. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems - AAMAS '05*, page 107, 2005. doi: 10.1145/1082473.1082812.
- [23] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry*, 37(3):255–274, 1998. doi: 10.1016/S0166-3615(98)00102-X.
- [24] S. Brückner and J. Wyns. Designing agents for manufacturing control. In *Proceedings of the 2nd AI & Manufacturing Research Planning Workshop*, pages 40–46, 1998.

- [25] S. Bussmann and K. Schild. An agent-based approach to the control of flexible production systems. In *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597)*, volume 2, pages 481–488, 2001. doi: 10.1109/ETFA.2001.997722.
- [26] D. Mcfarlane and A. Lucas. The Cambridge Packing Cell — A Holonic Enterprise Demonstrator. In *Multi-Agent Systems and Applications III, 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague, Czech Republic*, volume 2691, 2003. doi: 10.1007/3-540-45023-8.
- [27] L. Mönch, M. Stehli, and J. Zimmermann. FABMAS: An agent-based system for production control of semiconductor manufacturing processes. In *1st International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2003*, volume 2744, pages 258–267, 2003. doi: 10.1007/978-3-540-45185-3_24.
- [28] A. Lüder, J. Peschke, T. Sauter, S. Deter, and D. Diep. Distributed intelligence for plant automation based on multi-agent systems: The PABADIS approach. *Production Planning and Control*, 15(2): 201–212, 2004. doi: 10.1080/09537280410001667484.
- [29] J. L. M. Lastra, E. L. Torres, and A. W. Colombo. A 3D visualization and simulation framework for intelligent physical agents. *Holonic and Multi-agent Systems For Manufacturing, Proceedings*, 3593:AgentLink III Coordinat Act; I*PROMS EU Network Ex, 2005.
- [30] G. Cândido and J. Barata. A multiagent control system for shop floor assembly. *Holonic and Multi-Agent Systems for Manufacturing*, pages 293–302, 2007. doi: 10.1007/978-3-540-74481-8_28.
- [31] J. Barata and L. M. Camarinha-Matos. Coalitions of manufacturing components for shop floor agility - the CoBASA architecture. *International Journal of Networking and Virtual Organisations*, 2 (1):50–77, 2003. doi: 10.1504/IJNVO.2003.003518.
- [32] P. Leitão and F. Restivo. ADACOR: A holonic architecture for agile and adaptive manufacturing control. *Computers in Industry*, 57(2):121–130, 2006. doi: 10.1016/j.compind.2005.05.005.
- [33] P. Leitão and F. J. Restivo. Implementation of a holonic control system in a flexible manufacturing system. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 38(5):699–709, 2008. doi: 10.1109/TSMCC.2008.923881.
- [34] P. Leitão, N. Rodrigues, C. Turrin, and A. Pagani. Multiagent System Integrating Process and Quality Control in a Factory Producing Laundry Washing Machines. *IEEE Transactions on Industrial Informatics*, 11(4):879–886, 2015. doi: 10.1109/TII.2015.2431232.
- [35] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux. Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution. *Computers in Industry*, 66:99–111, 2015. doi: 10.1016/j.compind.2014.10.011.

- [36] J. Dias, J. Vallhagen, J. Barbosa, and P. Leitao. Agent-based reconfiguration in a micro-flow production cell. In *Proceedings - 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017*, 2017. doi: 10.1109/INDIN.2017.8104931.
- [37] R. Chase, N. J. Aquilano, and F. R. Jacobs. *Production and operations management: manufacturing and services*. Irwin/McGraw-Hill, 1998.
- [38] D. Trentesaux, C. Pach, A. Bekrar, Y. Sallez, T. Berger, T. Bonte, P. Leitão, and J. Barbosa. Benchmarking flexible job-shop scheduling and control systems. *Control Engineering Practice*, 21(9): 1204–1225, 2013. doi: 10.1016/j.conengprac.2013.05.004.
- [39] M. Fletcher, D. McFarlane, A. Thorne, D. Jarvis, and A. Lucas. Evaluating a holonic packing cell. In *Holonic and Multi-Agent Systems for Manufacturing*, volume 2744, pages 246–257. Springer, Berlin, Heidelberg, 2003. doi: 10.1007/978-3-540-45185-3_23.
- [40] M. Fletcher, D. McFarlane, A. Lucas, J. Brusey, and A. Thorne. The divergence of theoretical and actual designs in a holonic packing cell. *Intelligent Manufacturing Systems*, 36(3):Int Federat Automat Control; IFAC TC MIM; IFAC TC, apr 2003. doi: 10.1016/S1474-6670(17)37744-3.
- [41] P. Vrba and V. Marcík. Simulation in Agent-Based Control Systems: MAST Case Study. In *IFAC Proceedings Volumes*, volume 38, pages 145–152. Elsevier, jan 2005. doi: 10.3182/20050703-6-CZ-1902.01508.
- [42] P. Vrba, M. Radaković, M. Obitko, and V. Mařík. Semantic technologies: latest advances in agent-based manufacturing control systems. *International Journal of Production Research*, 49(5):1483–1496, 2011. doi: 10.1080/00207543.2010.518746.
- [43] Y. Sallez, T. Berger, and D. Trentesaux. A stigmergic approach for dynamic routing of active products in FMS. *Computers in Industry*, 60(3):204–216, apr 2009. doi: 10.1016/j.compind.2008.12.002.
- [44] N. Zbib, C. Pach, Y. Sallez, and D. Trentesaux. Heterarchical production control in manufacturing systems using the potential fields concept. *Journal of Intelligent Manufacturing*, 23(5):1649–1670, 2012. doi: 10.1007/s10845-010-0467-3.
- [45] C. Pach, T. Berger, T. Bonte, and D. Trentesaux. ORCA-FMS: A dynamic architecture for the optimized and reactive control of flexible manufacturing scheduling. *Computers in Industry*, 65(4): 706–720, 2014. doi: 10.1016/j.compind.2014.02.005.
- [46] A. Bekkar, O. Guemri, A. Bekrar, N. Aissani, B. Beldjilali, and D. Trentesaux. An Iterative Greedy Insertion Technique for Flexible Job Shop Scheduling Problem. *IFAC-PapersOnLine*, 49(12):1956–1961, jan 2016. doi: 10.1016/j.ifacol.2016.07.917.
- [47] J. F. Jimenez, A. Bekrar, G. Zambrano-Rey, D. Trentesaux, and P. Leitão. Pollux: a dynamic hybrid control architecture for flexible job shop systems. *International Journal of Production Research*, 55(15):4229–4247, 2017. doi: 10.1080/00207543.2016.1218087.

- [48] J. F. Jimenez, A. Bekrar, D. Trentesaux, and P. Leitão. A switching mechanism framework for optimal coupling of predictive scheduling and reactive control in manufacturing hybrid control architectures. *International Journal of Production Research*, 54(23):7027–7042, dec 2016. doi: 10.1080/00207543.2016.1177237.
- [49] G. Zambrano Rey, T. Bonte, V. Prabhu, and D. Trentesaux. Reducing myopic behavior in FMS control: A semi-heterarchical simulation-optimization approach. *Simulation Modelling Practice and Theory*, 46:53–75, 2014. doi: 10.1016/j.simpat.2014.01.005.
- [50] J. Barbosa. *Self-organized and evolvable holonic architecture for manufacturing control. Université de Valenciennes et du Hainaut-Cambrésis*. PhD thesis, 2015.
- [51] J.-F. Jiménez. Dynamic and hybrid architecture for the optimal reconfiguration of control systems: application to manufacturing control. Université de Valenciennes et du Hainaut-Cambresis. PhD Thesis. 2017.
- [52] P. Leitão, F. Casais, and F. Restivo. *Holonic Manufacturing Control: A Pratical Implementation*, volume 159. Kluwer Academic Publishers, Boston, 2005. doi: 10.1007/b99775.
- [53] P. J. P. Leitaó. *An Agile and Adaptive Holonic Architecture for Manufacturing Control. Faculty of Engineering of University of Porto*. PhD thesis, 2004.
- [54] J. A. B. D. Oliveira and P. L. C. D. Matos. *Coalition Based Approach for Shop Floor Agility – A Multiagent Approach. Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia*. PhD thesis, 2003.
- [55] Foundation For Intelligent Physical Agents. FIPA Agent Management Specification, 2002. URL <http://www.fipa.org/specs/fipa00023/SC00023K.pdf>.
- [56] Foundation For Intelligent Physical Agents. FIPA ACL Message Structure Specification, 2002. URL <http://www.fipa.org/specs/fipa00061/SC00061G.pdf>.
- [57] Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification, 2002. URL <http://www.fipa.org/specs/fipa00037/>.
- [58] Foundation For Intelligent Physical Agents. FIPA Contract Net Interaction Protocol Specification, 2002. URL <http://www.fipa.org/specs/fipa00029/SC00029H.html>.
- [59] Foundation For Intelligent Physical Agents. FIPA Propose Interaction Protocol Specification, 2002. URL <http://www.fipa.org/specs/fipa00036/SC00036H.html>.
- [60] Foundation For Intelligent Physical Agents. FIPA Request Interaction Protocol Specification, 2002. URL <http://www.fipa.org/specs/fipa00026/SC00026H.html>.
- [61] T. Murata. Petri Nets: Properties, Analysis and Applications. In *Proceedings of the IEEE*, volume 77, pages 541–580, 1989. doi: 10.1109/5.24143.

- [62] GAMA Platform Web Site: <http://gama-platform.org/>, accessed on May 10, 2018.
- [63] J. Dias, J. Barbosa, and P. Leitão. Deployment of industrial agents in heterogeneous automation environments. In *Proceeding - 2015 IEEE International Conference on Industrial Informatics, INDIN 2015*, pages 1330–1335, 2015. doi: 10.1109/INDIN.2015.7281928.
- [64] JADE Web Site: <http://jade.tilab.com/>, accessed on April 5, 2018.

Appendix A

Source Code

Instruction List program loaded in the PLC:

XOB	16	STH	ElevadorP3	STH	TapeteDirec
EXOB		ANL	ElevadorS3	SET	tdirec
COB	0	SET	ElevDirec	STL	TapeteDirec
	0	SET	Elev0n0ff	RES	tdirec
STH	ElevadorP0	STH	ElevadorP3	CPB	ProgramaElevador
ANL	ElevadorS0	ANH	ElevadorS3	ECOB	
RES	ElevDirec	RES	Elev0n0ff		
SET	Elev0n0ff	RES	ElevadorP3		
STH	ElevadorP0	STH	Free		
ANH	ElevadorS0	SET	LVerde		
RES	Elev0n0ff	STL	Free		
RES	ElevadorP0	RES	LVerde		
STH	ElevadorP1	STH	JobWaiting		
ANH	ElevadorS0	SET	Lamar		
SET	ElevDirec	STL	JobWaiting		
SET	Elev0n0ff	RES	Lamar		
STH	ElevadorP1	STH	Operating		
ANH	ElevadorS2	SET	LEnc		
RES	ElevDirec	STL	Operating		
SET	Elev0n0ff	RES	LEnc		
STH	ElevadorP1	STH	Axis		
ANH	ElevadorS3	SET	Operating		
RES	ElevDirec	SET	Cil_A		
SET	Elev0n0ff	LD	temp		
STH	ElevadorP1	RES	200		
ANH	ElevadorS1		Axis		
RES	Elev0n0ff	STH	rcomp		
RES	ElevadorP1	SET	Operating		
STH	ElevadorP2	SET	Cil_B		
ANH	ElevadorS0	LD	temp		
SET	ElevDirec	RES	200		
SET	Elev0n0ff		rcomp		
STH	ElevadorP2	STH	Lcomp		
ANH	ElevadorS1	SET	Operating		
SET	ElevDirec	SET	Cil_C		
SET	Elev0n0ff	LD	temp		
STH	ElevadorP2	RES	200		
ANH	ElevadorS3		Lcomp		
RES	ElevDirec	STL	temp		
SET	Elev0n0ff	RES	Operating		
STH	ElevadorP2	RES	Cil_A		
ANH	ElevadorS2	RES	Cil_B		
RES	Elev0n0ff	RES	Cil_C		
RES	ElevadorP2	RES	Tapete0n0ff		
		SET	tonoff		
		STL	Tapete0n0ff		
		RES	tonoff		

Intermediate MATLAB Script source code:

```

1      % Interface between the Simulation and the PLC
2
3      % Create the dialog box where the connection can be stopped
4      InterfaceDialog = dialog('Position',[400 600 250 150],'Name','Interface');
5      Text = uicontrol('Parent',InterfaceDialog,'Style','text','Position',[20 80 210 40],...
6          'String','Click the button to stop the connection and close the window.');
```

```

7      StopButton = uicontrol('Parent',InterfaceDialog,'Position',[75 40 100 25],'String',...
8          'Stop Connection','Callback','stop=0;delete(gcf)');
```

```

9
10     % Initialize variable stop
11     stop=1;
12
13     % Necessary addresses to write in the real lab
14     AddressW=[1087,1088,1089,1091,1092,1093,1112,1113,1114,1115,19,20];
15     % Necessary addresses to read in the real lab
16     AddressR=[1087,1088,1089,1091,1092,1093,1112,1113,1114,1115,1094,1095];
17     %(there is no function to read outputs, so we read flags with the same value)
18
19     while stop~=0
20
21         % Read values from PLC
22         for i = 1:12
23             ValuesPLC(i)=readFlag(AddressR(i));
24         end
25
26         % Open file to read
27         fid = fopen('C:\Users\alunos_ai\Desktop\gama\Workspace\Thesis\connection.txt','rt');
```

```

28         |
29         % Read values from file and write them in the PLC if they are different
30         i=1;
31         while i<=12
32             line = fgetl(fid);
33             ValuesFILE(i)=str2double(line);
34             if ValuesFILE(i)~=ValuesPLC(i) && i<11
35                 writeFlag(AddressW(i),ValuesFILE(i));
36             elseif ValuesFILE(i)~=ValuesPLC(i) && i>=11
37                 writeOutput(AddressW(i),ValuesFILE(i));
38             end
39             i=i+1;
40         end
41
42         % Close file
43         fclose(fid);
44
45         % Read values from PLC and check if they are different from the file
46         different=0;
47         for i = 1:12
48             ValuesPLC(i)=readFlag(AddressR(i));
49             if ValuesPLC(i)~=ValuesFILE(i)
50                 different=different+1;
51             end
52         end
53
54         % If they are different, open file with write permission and write the PLC values
55         if different~=0
56             fid = fopen('C:\Users\alunos_ai\Desktop\gama\Workspace\Thesis\connection.txt','wt');
```

```

57             fprintf(fid,'%d\n',ValuesPLC);
58             fclose(fid);
59         end
60
61         % Pause to set the required response time
62         pause(0.01);
63     end

```


JAVA class Job Agent source code:

```
1// Source code for the definition of public class JobAgent
2
3//Import section
4import jade.core.Agent;
5import jade.core.AID;
6import jade.core.behaviours.*;
7import jade.lang.acl.ACLMessage;
8import jade.proto.ContractNetInitiator;
9import jade.domain.DFService;
10import jade.domain.FIPAException;
11import jade.domain.FIPAAgentManagement.DFAgentDescription;
12import jade.domain.FIPAAgentManagement.ServiceDescription;
13import jade.domain.FIPANames;
14import java.util.Vector;
15
16public class JobAgent extends Agent {
17
18    // Current operation
19    private Integer OpNumber=0;
20    // The list of WS
21    private AID[] WS;
22
23    // Initializations
24    protected void setup() {
25
26        // Get the required operations as start-up arguments
27        Object[] args = getArguments();
28        if (args != null && args.length > 0) {
29            // Printout a welcoming message
30            System.out.println("Hello! Job Agent "+getAID().getName()+" is ready! Required operations are:");
31            for (int i = 0; i < args.length; ++i) {
32                System.out.println(args[i]);
33            }
34        }
35        else {
36            // Make the agent terminate
37            System.out.println("No required operations specified.");
38            doDelete();
39        }
40
41        // Create CFP message
42        ACLMessage mm = new ACLMessage(ACLMessage.CFP);
43
44        // Create Sequential Behavior production
45        SequentialBehaviour production = new SequentialBehaviour(this);
46        for (int i = 0; i<args.length; ++i) {
47
48            // Add SubBehaviour to fill the CFP message
49            production.addSubBehaviour(new OneShotBehaviour(this) {
50                public void action() {
51                    System.out.println(OpNumber+" operation done.");
52                    if (OpNumber.equals(args.length)) {
53                        // Make the agent terminate
54                        System.out.println("Production finished.");
55                        doDelete();
56                    }
57                }
58                else {
59                    // Find WS that can do the current operation
60                    DFAgentDescription template = new DFAgentDescription();
61                    ServiceDescription sd = new ServiceDescription();
62                    sd.setType((String)args[OpNumber]);
63                    template.addServices(sd);
64                    try {
65                        DFAgentDescription[] result = DFService.search(myAgent, template);
66                        System.out.println("Found the following WS for "+args[OpNumber]+":");
67                        WS = new AID[result.length];
68                        for (int i = 0; i < result.length; ++i) {
69                            WS[i] = result[i].getName();
70                            System.out.println(WS[i].getName());
71                        }
72                    }
73                    catch (FIPAException fe) {
74                        fe.printStackTrace();
75                    }
76                    // Fill the message
77                    mm.reset();
78                    for (int i = 0; i < WS.length; ++i) {
79                        mm.addReceiver(WS[i]);
80                    }
81                    mm.setPerformative((ACLMessage.CFP));
82                    mm.setProtocol(FIPANames.InteractionProtocol.FIPA_CONTRACT_NET);
83                    mm.setContent((String)args[OpNumber]);
84                    System.out.println("Agent "+getAID().getName()+" sent a CFP message for the operation "+args[OpNumber]+".");
85                }
86            });
87        }
88    }
89}
```

```

88 // Add SubBehaviour ContractNetInitiator
89 production.addSubBehaviour(new ContractNetInitiator(this, mm) {
90     protected void handlePropose(ACLMessage propose, Vector accept) {
91         System.out.println("Agent "+propose.getSender().getName()+" proposed "+propose.getContent()+".");
92         ACLMessage reply = propose.createReply();
93         reply.setContent((String)args[OpNumber]);
94         reply.setPerformative(ACLMessage.ACCEPT_PROPOSAL);
95         accept.addElement(reply);
96         skipNextResponses();
97         System.out.println("Accepting proposal from "+propose.getSender().getName()+".");
98     }
99     protected void handleRefuse(ACLMessage refuse) {
100         System.out.println("Agent "+refuse.getSender().getName()+" refused.");
101     }
102     protected void handleFailure(ACLMessage failure) {
103         if (failure.getSender().equals(myAgent.getAMS())) {
104             System.out.println("Responder does not exist");
105         }
106         else {
107             System.out.println("Agent "+failure.getSender().getName()+" failed.");
108         }
109     }
110     protected void handleInform(ACLMessage inform) {
111         System.out.println("Agent "+inform.getSender().getName()+" successfully started "+args[OpNumber]+" on " +getAID().getName()+".");
112         try {
113             Thread.sleep(20000);
114         }
115         catch (Exception e) {
116         }
117         System.out.println("Product Agent "+getAID().getName()+" completed "+args[OpNumber]+".");
118         OpNumber=OpNumber+1;
119     }
120 });
121 }
122 addBehaviour(production);
123 }
124
125 // Clean-up operations
126 protected void takeDown() {
127     // Printout a dismissal message
128     System.out.println("Product Agent "+getAID().getName()+" terminating.");
129 }
130 }
131 }

```

JAVA class Workstation Agent source code:

```

1@// Source code for the definition of public class WsAgent
2
3 //Import section
4import jade.core.Agent;
5import jade.core.behaviours.*;
6import jade.lang.acl.ACLMessage;
7import jade.lang.acl.MessageTemplate;
8import jade.domain.DFService;
9import jade.domain.FIPAException;
10import jade.domain.FIPAAgentManagement.DFAgentDescription;
11import jade.domain.FIPAAgentManagement.ServiceDescription;
12
13public class WsAgent extends Agent {
14
15    // Type of operations
16    private String operation1;
17    private String operation2;
18    private String operation3;
19
20    // Initializations
21    protected void setup() {
22
23        // Get the type of operations as a start-up arguments
24        Object[] args = getArguments();
25        if (args != null && args.length > 2) {
26
27            // Save the arguments in strings
28            operation1 = (String) args[0];
29            operation2 = (String) args[1];
30            operation3 = (String) args[2];
31
32            // Printout a welcoming message
33            System.out.println("Hello! WS Agent "+getAID().getName()+" can do the operations "+operation1+", "+operation2+" and "+operation3+ ".");
34
35            // Register the machine operations in the yellow pages
36            DFAgentDescription dfd = new DFAgentDescription();
37            dfd.setName(getAID());
38            ServiceDescription s1 = new ServiceDescription();
39            s1.setType(operation1); s1.setName("Machine");
40            ServiceDescription s2 = new ServiceDescription();
41            s2.setType(operation2); s2.setName("Machine");
42            ServiceDescription s3 = new ServiceDescription();
43            s3.setType(operation3); s3.setName("Machine");

```

```

44         dfd.addServices(s1);
45         dfd.addServices(s2);
46         dfd.addServices(s3);
47         try {
48             DFService.register(this, dfd);
49         }
50         catch (FIPAException fe) {
51             fe.printStackTrace();
52         }
53
54         // Add the behavior answer proposals from product agents
55         addBehaviour(new AnswerCFP());
56
57         // Add the behavior answer acceptances from product agents
58         addBehaviour(new AnswerAcceptances());
59     }
60
61     else {
62         // Make the agent terminate
63         System.out.println("No operations specified.");
64         doDelete();
65     }
66 }
67
68 // Clean-up operations
69 protected void takeDown() {
70
71     // Deregister from the yellow pages
72     try {
73         DFService.deregister(this);
74     }
75     catch (FIPAException fe) {
76         fe.printStackTrace();
77     }
78
79     // Printout a dismissal message
80     System.out.println("Machine Agent "+getAID().getName()+" terminating.");
81 }
82
83

```

```

84 private class AnswerCFP extends CyclicBehaviour {
85     public void action() {
86         MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.CFP);
87         ACLMessage msg = myAgent.receive(mt);
88         if (msg != null) {
89             String content = (String) msg.getContent();
90             if (content.equals(operation1) || content.equals(operation2) || content.equals(operation3)) {
91                 ACLMessage reply = msg.createReply();
92                 reply.setPerformative(ACLMessage.PROPOSE);
93                 reply.setContent(msg.getContent());
94                 myAgent.send(reply);
95             }
96             else {
97                 ACLMessage reply = msg.createReply();
98                 reply.setPerformative(ACLMessage.REFUSE);
99                 reply.setContent(msg.getContent());
100                 myAgent.send(reply);
101             }
102         }
103         else {
104             block();
105         } } }
106
107 private class AnswerAcceptances extends CyclicBehaviour {
108     public void action() {
109         MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);
110         ACLMessage msg = myAgent.receive(mt);
111         if (msg != null) {
112             ACLMessage reply = msg.createReply();
113             reply.setPerformative(ACLMessage.INFORM);
114             reply.setContent(msg.getContent());
115             myAgent.send(reply);
116             try {
117                 Thread.sleep(20000);
118             }
119             catch (Exception e) {
120             }
121         }
122         else {
123             block();
124         } } } }

```

Appendix B

MILP formulation

Notations for parameters:

P	set of jobs, $P = \{1, 2, \dots, n\}$
R	set of machines, $R = \{1, 2, \dots, r\}$
I_j	set of operations of the job j , $I_j = \{1, 2, \dots, I_j \}$, $j \in P$
O_{ij}	operation i of the job j
R_{ij}	set of machines that can perform operation O_{ij} , $R_{ij} \in R$
p_{ij}	processing time of operation i ($i \in I_{ij}$)
$tt_{r_1 r_2}$	transportation time from machine r_1 to r_2
MJ	maximum simultaneous jobs in the shop floor
ci_r	input queue capacity of machine r
d_j	due date of job j , $j = 1, \dots, n$

Notations for variables:

t_{ij}	completion time of operation O_{ij} ($i \in I_j$), $t_{ij} \in N$
μ_{ijr}	a binary variable set to 1 if operation O_{ij} is performed on machine r ; 0, otherwise.
b_{ijkl}	a binary variable set to 1 if operation O_{ij} is performed before operation O_{kl} ; 0, otherwise.
$tr_{ijr_1 r_2}$	a binary variable set to 1 if job j is transported to machine r_2 after performing operation O_{ij} ; 0, otherwise.
w_{ijr}	waiting time of operation O_{ij} in the queue of machine r
wv_{ijklr}	a binary variable set to 1 if operation O_{ij} is waiting for operation O_{kl} in the queue of machine r ; 0, otherwise.
z_{lj}	a binary variable set to 1 if job l and job j are in the shop floor in the same time; 0, otherwise.

Disjunctive constraints: A machine can process one operation at time, and an operation is performed by only one machine.

$$t_{ij} + p_{kl}\mu_{klr} + BMb_{ijkl} \leq t_{kl} + BM, \quad \forall i, k \in I, \forall j, l \in P, \forall r \in R_{ij} \quad (\text{B.1})$$

where BM is a large number.

$$b_{ijkl} + b_{kl ij} \leq 1 \quad \forall i \in I_j, k \in I_l, \forall j, l \in P \quad (\text{B.2})$$

$$\sum_{r \in R_{ij}} \mu_{ijr} = 1 \quad \forall i \in I_j, \forall j \in P \quad (\text{B.3})$$

Precedence constraints: These constraints insure job's production sequence.

$$t_{(i+1)j} \geq t_{ij} + p_{(i+1)j} + w_{(i+1)jr2} + \sum_{r1, r2 \in R} tt_{r1r2} tr_{ijr1r2} \quad \forall i \in I_j, \forall j \in P, \forall r1, r2 \in R_{ij} \quad (\text{B.4})$$

$$\sum_{\substack{r1, r2 \in R \\ r1 \neq r2}} tr_{ijr1r2} \leq 1, \quad \forall i \in I_j, \forall j \in P \quad (\text{B.5})$$

Allocation and transportation relationship: If successive operations of a job are performed on different machines, there is a transportation operation between those two machines.

$$\mu_{ijr1} + \mu_{(i+1)jr2} - 1 \leq tr_{ijr1r2} \quad \forall i \in I_j, \forall j \in P, \forall r1, r2 \in R_{ij}, r1 \neq r2 \quad (\text{B.6})$$

$$\mu_{ijr1} + \mu_{(i+1)jr2} \geq (1 + \varepsilon) tr_{ijr1r2} \quad \forall i \in I_j, \forall j \in P, \forall r1, r2 \in R_{ij}, r1 \neq r2 \quad (\text{B.7})$$

where ε is a small number.

Queue capacity of the machine input and FIFO rule: Each machine has a limited queue capacity.

$$b_{ijkl} + wv_{ijklr} \leq 1 \quad \forall i, k \in I, \forall j, l \in P, \forall r \in R_{ij} \cap R_{kl} \quad (\text{B.8})$$

$$b_{ijkl} - wv_{kl ij r} \geq 0 \quad \forall i \in I_j, \forall k \in I_l, \forall j, l \in P, \forall r \in R_{ij} \cap R_{kl} \quad (\text{B.9})$$

$$wv_{ijklr} + wv_{kl ij r} \leq 1 \quad \forall i \in I_j, \forall k \in I_l, \forall j, l \in P, \forall r \in R_{ij} \cap R_{kl} \quad (\text{B.10})$$

$$t_{ij} - p_{ij} + BMb_{ijkl} + BMwv_{kl ij r} \leq t_{kl} - p_{kl} - w_{klr} + 2BM \quad \forall i \in I_j, \forall k \in I_l, \forall j, l \in P, j \neq l, \forall r \in R_{ij} \cap R_{kl} \quad (\text{B.11})$$

$$w_{klr} \leq \sum_{\substack{i \in I_j \\ j \in P, j \neq l}} p_{ij} wv_{kl ij r} \quad \forall k \in I_l, \forall l \in P, \forall r \in R_{kl} \quad (\text{B.12})$$

$$\mu_{ijr} + \mu_{klr} \geq 2(wv_{ijklr} + wv_{kl ij r}) \quad \forall i \in I_j, \forall k \in I_l, \forall j, l \in P, \forall r \in R_{ij} \cap R_{kl} \quad (\text{B.13})$$

$$t_{ij} + BMb_{ijkl} \leq t_{kl} + BM \quad \forall i \in I_j, \forall k \in I_l, \forall j, l \in P \quad (\text{B.14})$$

$$t_{ij} - p_{ij} \mu_{ijr} + BMb_{ijkl} \leq t_{kl} - p_{kl} \mu_{klr} + BM \quad \forall i \in I_j, \forall k \in I_l, \forall j, l \in P, \forall r \in R \quad (\text{B.15})$$

$$t_{ij} - p_{ij} \mu_{ijr} - w_{ijr} + BMb_{ijkl} \leq t_{kl} - p_{kl} \mu_{klr} - w_{klr} + BM \quad \forall i \in I_j, \forall k \in I_l, \forall j, l \in P, \forall r \in R \quad (\text{B.16})$$

$$\sum_{\substack{l \in P, k \in I_l \\ l \neq j}} wv_{ijklr} \leq c_{ir} - 1, \quad \forall i \in I_j, \forall j \in P, \forall r \in R_{ij} \cap R_{kl} \quad (\text{B.17})$$

Limitation of the number of jobs in the system: The number of simultaneous jobs in the shop floor

can be limited by MJ .

$$\sum_{\substack{l \in P \\ l \neq j}} z_{lj} \leq MJ - 1 \quad \forall j \in P \quad (\text{B.18})$$

$$z_{jl} \geq b_{0luj} + b_{0j0l} - 1 \quad \forall j, l \in P \quad (\text{B.19})$$

$$z_{jl} \leq 1 - b_{0l0j} + b_{ujul} \quad j, l \in P \quad (\text{B.20})$$

$$z_{jl} \geq b_{0l0j} + b_{ujul} - 1 \quad \forall j, l \in P \quad (\text{B.21})$$

Constraints for the type of each variable.

$$t_{ij} \geq p_{ij} \quad \forall i \in I_j, j \in P \quad (\text{B.22})$$

$$b_{ijkl} \in \{0, 1\} \quad \forall i \in I_j, j \in P, \forall k \in I_l, l \in P \quad (\text{B.23})$$

$$tr_{ijrr'} \in \{0, 1\} \quad \forall i \in I_j, j \in P, \forall r, r' \in R_{ij} \quad (\text{B.24})$$

$$\mu_{ijr} \in \{0, 1\} \quad \forall i \in I_j, j \in P, \forall r \in R_{ij} \quad (\text{B.25})$$

A valid inequality: The following cut is added to the model to improve the solving.

$$\mu_{klr} + \mu_{ijr} - (b_{ijkl} + b_{klij}) \leq 1 \quad \forall i \in I_j, k \in I_l, \forall j, l \in P, \forall r \in R \quad (\text{B.26})$$

Quantitative performance: Makespan is the time at which the last job is completed.

$$C_{max} = \max_{\forall i \in I, \forall j \in P} t_{ij} \quad (\text{B.27})$$

Objective function: The objective function chosen is the minimization of the maximum Makespan.

$$\text{Minimize } C_{max} \quad (\text{B.28})$$