An Agile and Adaptive Holonic Architecture for Manufacturing Control

Dissertation submitted to the Faculty of Engineering of University of Porto to obtain the Degree of Doctor of Philosophy in Electrotechnical and Computational Engineering, scientific area of Industrial Automation

Supervision: Prof. Doutor Francisco Restivo

Paulo Jorge Pinto Leitão

Porto, January 2004

Paulo Jorge Pinto Leitão

Department of Electrotechnical Engineering

Polytechnic Institute of Bragança

Quinta Santa Apolónia, Apartado 134

5301-857 Bragança

Portugal

URL: http://www.ipb.pt/∼pleitao

e-mail: pleitao@ipb.pt

To Rute, Raquel and Luís.

Abstract

In the last decades significant changes in the manufacturing environment have been noticed: moving from a local economy towards a global economy, with markets asking for products with high quality at lower costs, highly customised and with short life cycle.

In this environment, the manufacturing enterprises, to avoid the risk to lose competitiveness, search to answer more closely to the customer demands, by improving their flexibility and agility, while maintaining their productivity and quality. Actually, the dynamic response to emergence is becoming a key issue, due to the weak response of the traditional manufacturing control systems to unexpected disturbances, mainly because of the rigidity of their control architectures.

In these circumstances, the challenge is to develop manufacturing control systems with autonomy and intelligence capabilities, fast adaptation to the environment changes, more robustness against the occurrence of disturbances, and easier integration of manufacturing resources and legacy systems. Several architectures using emergent concepts and technologies have been proposed, in particular those based in the holonic manufacturing paradigm.

Holonic manufacturing is a paradigm based in the ideas of the philosopher Arthur Koestler, who proposed the word *holon* to describe a basic unit of organisation in biological and social systems. A holon, as Koestler devised the term, is an identifiable part of a (manufacturing) system that has a unique identity, yet is made up of sub-ordinate parts and in turn is part of a larger whole. The introduction of the holonic manufacturing paradigm allows a new approach to the manufacturing problem, bringing the advantages of modularity, decentralisation, autonomy, scalability, and re-use of software components.

This dissertation intends to develop an agile and adaptive manufacturing control architecture to face the current requirements imposed to the manufacturing enterprises. The architecture proposed in this dissertation addresses the need for the fast reaction to disturbances at the shop floor level, increasing the agility and flexibility of the enterprise, when it works in volatile environments, characterised by the frequent occurrence of unexpected disturbances.

The proposed architecture, designated by ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems), is based in the holonic manufacturing paradigm, build upon autonomous and cooperative holons, allowing the development of manufacturing control applications that present all the features of decentralised and holonic systems.

ADACOR holonic architecture introduces an adaptive control that balances dynamically between a more centralised structure and a more decentralised one, allowing to combine the global production optimisation with agile reaction to unexpected disturbances.

Resumo

Nas últimas décadas têm-se assistido a mudanças significativas no ambiente de fabrico: evoluindo de uma economia local para um economia global, com os mercados a procurar produtos com elevada qualidade a baixos preços, altamente customizados e com um ciclo de vida curto.

Neste ambiente, as empresas de manufactura, para evitar o risco de perda de competitividade, procuram responder às solicitações dos clientes, melhorando a sua flexibilidade e agilidade, mantendo os mesmos índices de productividade e qualidade. Na verdade, a resposta dinâmica à emergência está a tornar-se num assunto chave, devido à fraca resposta a perturbações que os sistemas de controlo de fabrico tradicionais apresentam, principalmente devido à rigidez das suas arquitecturas de controlo.

Nestas circunstâncias, é fundamental o desenvolvimento de sistemas de controlo de fabrico com capacidades de autonomia e inteligência, rápida adaptação às mudanças, maior robustez à ocorrência de perturbações e fácil integração de recursos físicos e sistemas legados. Diversas arquitecturas usando conceitos e tecnologias emergentes têm sido propostas, em particular algumas baseadas no paradigma da produção holónica.

O paradigma da produção holónica é inspirado nas ideias de Arthur Koestler, que propôs a palavra *holon* para descrever uma unidade básica de organizaçõa de sistemas biológicos e sociais. Um holon, de acordo com a definição de Koestler, é uma parte identificável do sistema com identidade única, composta por sub-partes e fazendo simultaneamente parte do todo. A introdução do paradigma da produção holónica permite uma nova abordagem aos sistemas de controlo de fabrico, trazendo vantagens de modularidade, descentralização, autonomia, escalabilidade e re-utilização de componentes.

Esta dissertação pretende desenvolver uma arquitectura de controlo ágil e adaptativa que suporte os requisitos actuais impostos às empresas de manufactura. A arquitectura proposta visa a necessidade de uma reacção rápida a perturbações, ao nível da planta fabril, melhorando a flexibilidade e agilidade da empresa quando esta opera em ambientes voláteis, caracterizados pela ocorrência frequente de perturbações inexperadas.

A arquitectura proposta, designada por ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems), é baseada no paradigma da produção holónica e construída sobre holons autónomos e cooperativos, permitindo o desenvolvimento de aplicações de controlo de fabrico que apresentem todas as caracteristicas dos sistemas descentralizados e holónicos.

A arquitectura holónica ADACOR introduz um controlo adaptativo que balança dinamicamente entre uma estrutura de controlo mais centralizada e uma mais descentralizada, permitindo combinar a optimização da produção com a ágil reacção a perturbações.

Resumé

Au cours des récentes décennies, des changements manifestes sont apparus dans le paysage industriel conduisant à la production personnalisée de masse : évolution d'une économie locale vers une économie globale, avec des marchés exigeant des produits de haute qualité, aux prix les plus bas, très personnalisé et de faible cycle de vie.

Dans un tel environnement, les entreprises manufacturières, sous peine de perdre leur compétitivité, cherchent à répondre au plus près aux demandes de leurs clients, en développant leur flexibilité et leur agilité, tout en maintenant leur productivité et leur qualité. De fait, la réponse dynamique à l'émergence devient une question clé, du fait de la faiblesse de la réponse des systèmes traditionnels de pilotage de production à des perturbations imprévues, en raison de la rigidité de leur architectures de commande.

Dans ces circonstances, la gageure est de développer des systèmes de pilotage de production, dotés de facultés d'autonomie et d'intelligence, adaptables de façon agile et rapide aux changements de l'environnement, plus robustes face aux possibles perturbations et d'une intégration plus aisée au cœur des ressources de production existantes. Plusieurs architectures utilisant les concepts et technologies émergents on été proposés, en particuliers ceux fondés sur le paradigme de la production holonique.

Le paradigme de la production holonique est inspiré des idées du philosophe Arthur Koestler, qui a proposé le mot *holon* pour décrire une unité d'organisation de base dans les systèmes biologiques et sociaux. Un holon, selon la définition de Koestler, est une partie d'un système (ici: de production) qui a une identité unique, bien que composé de sous-composants et qu'appartenant à un ensemble plus large. L'introduction du paradigme de la production holonique offre une nouvelle approche de la question de la production en apportant les avantages de la modularité, de la décentralisation, de l'autonomie, de la facilité de mise à l'échelle et de la ré-utilisation des composants logiciels.

La présente thèse vise à développer une architecture de pilotage de la production, agile et adaptative, pour faire face aux nouvelles contraintes imposées aux entreprises. L'architecture proposée dans cette thèse traite des besoins de réponse rapide aux perturbations au niveau de l'atelier, de façon à augmenter l'agilité et la flexibilité de l'entreprise quand elle travaille dans des environnements volatils, caractérisés par l'apparition fréquente de perturbations inattendues.

L'architecture proposée, dénommée ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) est basée sur le paradigme de la production holonique et construite à partir de holons autonomes et coopératifs; elle permet le développement d'applications de pilotage de production qui présentent toutes les caractéristiques des systèmes holoniques décentralisés.

L'architecture holonique ADACOR introduit une commande adaptative qui s'équilibre dynamiquement entre une structure plus centralisée et une structure plus répartie, permettant ainsi de combiner l'optimisation globale de la production avec la réaction agile aux perturbations imprévues.

Acknowledgements

This thesis is the result of some years of research, where I had opportunity to work closely with many people. I would like to take this opportunity to express my gratitude to all who helped me making the thesis a reality. I would like to specially recognise some of them here.

I would like to express my deeply gratitude to my research supervisor, Professor Francisco Restivo, for the trust he puts in me at the beginning of the PhD long walk. I also would like to thank for his encouragement, precious advises and discussions, that contributed to the elaboration of this thesis, and the improvement of the final result.

My acknowledge to Faculty of Engineering of University of Porto, for the opportunity to the elaboration of this research work.

I would like to express my gratitude to Escola Superior de Tecnologia e Gestão of Polytechnic Institute of Bragança for the availability and support given during this work.

I would like to thank to the program PRODEP (Programa de Desenvolvimento Educativo para Portugal/Programme of Educative Development for Portugal) for the financial support granted (under the measure PRODEP III - 5.3/2000), that allowed the dispense of the teaching activities during the last three years, being crucial to the achievement of the objectives and the completion of this work in due date.

I would like to express my gratitude to Armando W. Colombo for the fruitful cooperation, insights and inspiring discussions during the last research years, even when its overloaded agenda did not allow to spent more time doing the cooperative research.

I am happy to contribute to several European research projects, such as CIM Center of Porto, PASO, MOSCOT, DEDEMAS and Comme Agathe. In this context, I am very grateful for the inspiring and fruitful discussions and all the fun I had with European researchers. Sincere thanks to Raymond Boissier, José Barata, Per Gullander, Pär Klingstam, Ilkka Seilonen, Gerrit Teunis and Stefano Borgo.

I also would like to thank to all colleagues of Polytechnic Institute of Braganca, specially from the Department of Electrotechnical Engineering, for their friendship, support and inspiring discussions, that had enriched my knowledge in this research area; a special thank to José

Augusto Carvalho and Henrique Teixeira.

I would like to thank to Francisco Casais for his availability and support during the implementation phase, that had contributed for the success of the presented solution. Also, for his support when dealing with LateX editor tool: when I were not able to find a way to do something with LateX, he always found quickly a solution to the problem.

Last, but not least, I am deeply grateful to my family and friends for always being there. A special thank to Rute for the unflinching support and for her patience during all these years. Finally for Raquel and Luís, the promise that I will try in the future to spend more time with them.

Bragança, Portugal Paulo Leitão
January 24th, 2004

Glossary

Activity - A well-defined action or set of actions to be executed. In case of a time consuming activity, it is a temporal activity, otherwise it is an atomic activity.

Adaptation - Capability to adjust the control behaviour to the environmental conditions during the system life-cycle, maintaining the pursuance of its objectives.

Agenda (or schedule) - Timetable that describes the planned list of work orders that each manufacturing resource has to execute over the time.

Agility - Capability to react in a short period of time to the occurrence of unexpected disturbances (i.e. production environment changes).

Architecture - The practice (science) of designing and building systems, by indicating the system components, their functions and their interactions.

Disturbance - An unexpected manufacturing disruption that affects the production.

Flexibility - Capability to adapt to new, different or changing environments. In manufacturing world, several flexibility classifications are presented in literature, like mix, changeover, volume, product and sequencing.

Holonic Manufacturing System - One paradigm for the factory of the future that translates to the manufacturing world the concepts developed by Koestler to living organisms and social organisations. Holonic manufacturing is characterised by holarchies of autonomous and cooperative entities, called by holons, that integrates the entire range of manufacturing entities. A holon, as Koestler devised the term, is an identifiable part of a (manufacturing) system that has a unique identity, yet is made up of sub-ordinate parts and in turn is part of a larger whole. The essence of the holonic approach is the capability to decompose a complex problem into stable intermediate sub-problems, using hierarchy structures.

Manufacturing Control - Managing and controlling the physical activities in the factory aiming to execute the manufacturing plans. Normally, a manufacturing control comprises the short-term process plan and the shop floor activities.

Model - Abstract representation of a portion of reality intended to promote its understanding. There are several kind of models: conceptual model (definition of the concepts which are needed to describe things), functional model (description of the functionalities of a thing), process model (description of a process), generic model (description of a thing at an abstract level), meta model (description of a model), etc.

Performance Measurement - Process of using a tool or a procedure to evaluate a concrete efficiency parameter of the system.

Process Plan - Sequence of the individual processing and assembly operations needed to produce the part. The document used to specify the process sequence is the routing sheet that contains the identification of the part to produce, the sequence of operations and associated resource types, and estimated execution times.

Production Control - Encompasses manufacturing control of one or several plants, purchasing, material requirements planning, design, medium and long-term process planning, and other production activities. The production control is also referred to as the production planning and control (PPC).

Re-configurability - Ability to support different manufacturing system configurations, i.e. different production systems scenarios, with a small customisation task.

Robustness - Capability of a control system to remain working correctly and relatively stable, even in presence of disturbances.

Scheduling - Optimal allocation of resources to jobs over the time, where these assignments must obey to a set of constraints that reflect the temporal relationships between jobs and the capacity limitations of the resources.

Acronyms

AARIA	Autonomous Agents at Rock Island Arsenal
ACL	Agent Communication Language
ADACOR	Adaptive Holonic Control Architecture for Distributed Manufacturing Systems
AFPS	ADACOR Factory Plant Supervisor
AGV	Auto-Guided Vehicle
AI	Artificial Intelligent
AMS	Agent Management System
APM	ADACOR Product Manager
AS/RS	Automated Storage / Retrieval System
AUML	Agent Unified Modelling Language
BDI	Belief-Desire-Intention
BMS	Bionic Manufacturing System
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAPP	Computer Aided Process Planning
CCD	Charge Coupled Device
CHAMP	Chalmers Architecture and Methodology for Flexible Production
CIM	Computer Integrated Manufacturing
CLIPS	C Language Integrated Production System
CMU	Cooperative Manufacturing Units
CNC	Computer Numeric Control
CNP	Contract Net Protocol
ComC	Communication Component
CORBA	Common Object Request Broker Architecture
COSIMA	Control Systems for Integrated Manufacturing

CPN	Coloured Petri Net
DAI	Distributed Artificial Intelligence
DARPA	DARPA Agent Markup Language
DCOM	Distributed Component Object Model
DeC	Decision Component
DEDEMAS	Decentralised Decision-Making and Scheduling
DF	Director Facilitator
DNA	Deoxyribo Nucleic Acid
ECNP	Extended Contract Net Protocol
EDD	Earliest Due Date
ERP	Enterprise Resource Planning
ESPRIT	European Strategic Programme for Research in Information Technology
FACE	Flexible Assembly Control Environment
FACT	Factory Activity Control Module
FBD	Function Block Diagram
FC	Factory Co-ordination
FIFO	First In First Out
FIPA	Foundation for Intelligent Physical Agents
FMS	Flexible Manufacturing Systems
GUI	Graphical User Interface
HCBA	Holonic Component-based Architecture
HMS	Holonic Manufacturing System
IEC	International Electrotechnical Commission
IIOP	Internet Inter-ORB Protocol
IL	Instruction List
IMS	Intelligent Manufacturing System
InteRRap	Integration of Reactive Behaviour and Rational Planning
ISO	International Organization for Standardization
IT	Information Technologies
JADE	Java Agent Development Framework
JESS	Java Expert System Shell
JIT	Just in Time
JRMI	Java Remote Method Invocation
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language

LCD	Logical Control Device
LD	Ladder Diagram
LOR	Least Operations Remaining
KB	Knowledge Base
MADEMA	Manufacturing Decision-Making
MAP	Manufacturing Automation Protocol
MES	Manufacturing Execution System
MIT	Massachussetts Institute of Technology
MMS	Manufacturing Message Specification
MOSCOT	Modular Shop Control Toolkit for Flexible Manufacturing
MTBF	Mean Time Between Failures
NC	Numerical Control
OEM	Original Equipment Manufacturer
OIL	Ontology Interchange Language
OPC	OLE for Process Automation Control
ORB	Object Request Broker
OMG	Object Management Group
PAC	Production Activity Control
PASO	Paradigm Independent Shop Control for Smaller Manufacturing Units
PDCH	Partial Dynamic Control Hierarchy
PIC	Physical Interface Component
PID	Proportional, Integrated and Derivative
PLC	Programmable Logical Controller
PMS	Performance Measurement System
PN	Petri Net
PPC	Production Planning and Control
PROSA	Product-Resource-Order-Staff Architecture
RMA	Remote Monitoring Agent
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SCAPI	Shop Control Application Program Interface
SFC	Sequential Function Charts
SL	Semantic Language
SME	Small and Medium Enterprise
SPT	Shortest Processing Time

SST	Shortest Setup Time
ST	Structured Text
TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modelling Language
VMD	Virtual Manufacturing Device
VR	Virtual Resource
WIP	Work in Progress
YAMS	Yet Another Manufacturing System
XML	eXtended Markup Language

Nomenclature

Indexes

i	Index of the part/order.
j	Index of the resource.
k	Index of the operation.
t	Time.

Sets

$\mathcal{P}$	Set of products available in the system.
$\mathcal{R}$	Set of resources.
$\mathcal{PM}$	Set of processing machines.
$\mathcal{TO}$	Set of tools to be loaded in the processing machines.
$\mathcal{H}$	Set of human operators.
$\mathcal{T}$	Set of transporter resources.
$\mathcal{RM}$	Set of raw material.
$\mathcal{A}$	Set of auxiliary resources.
$\mathcal{M}$	Set of mover resources.
$\mathcal{OH}$	Set of operational holons.
$\mathcal{SH}$	Set of supervisor holons.
$\mathcal{PH}$	Set of product holons.
$\mathcal{TH}$	Set of task holons.
$\mathcal{S}_j$	Set of skills of a resource j.
$\mathcal{B}_{ik}$	Set of requirements to execute the operation $o_{ik.}$
$\mathcal{R}_{ik}$	Set of alternatives resources able to execute the operation $o_{ik.}$

Order Parameters

S_i	Start date of order i.

D_i	Due date of order i.
w_i	Priority of order i.
d_{ik}	Processing time to execute the operation o_{ik}.
d_{ikj}	Processing time to execute the operation o_{ik} at resource j.
sd_{ik}	Start date for the operation o_{ik}.
dd_{ik}	Due date for the operation o_{ik}.
b_{ik}	Scheduled start date of operation o_{ik}.
f_{ik}	Scheduled end date of operation o_{ik}.

Process Plan Parameters

Γ_i	Process plan to produce the part i (comprises Θ_i, Ω_i and ρ_{ik}).
Θ_i	Set of operations required to produce the part i.
o_{ik}	Operation k that belong to $\Theta_{i.}$
Ω_i	Order between two operations belonging to Θ_i (precedence between two operations).

Adaptation Parameters

α	Autonomy factor.
ρ	Pheromone parameter, which reflects the occurrence of a disturbance, either local or propagated.
δ	Scope of disturbance occurrence: {local} or {pheromone}.
τ	Reestablishment time (from a disturbance).
t_r	Recovery time (from a physical failure).
t_b	Blocking time. Used during the reaction to disturbances, by the operational holon, to define the temporal window where work orders must be returned to the task holons.
t_p	Polling time. Used during the reaction to disturbances, by the operational holon, to check if the machine has recovered and to re-estimate the time parameters if the machine is not recovered as forecasted.
t_f	Expected time for the occurrence of the next disturbance.

Credits Parameters

π	Credits of the task holon.
μ	Credits of the operational holon.
φ	Contracted value to execute an operation (equal to p_{jik}).

ξ	Penalty value in case of delay in the execution of an operation, for example due to a machine failure.
ν	Value offered by other task holons during a negotiation for release of a time window.

Performance Parameters

C_i	Completation time for the order i.
L_i	Manufacturing lead time for the order i.
T_i	Tardiness for the order i.
Q	Throughput.
pU_j	Percentage of capacity utilisation of resource j.
t_a	Time spent by the system to adapt its behaviour to the disturbance.
lQ	Loss of productivity: percentage of the reduction of the throughput over the steady throughput.
$r(pU)$	Standard deviation of the percentage of utilisation of a resource j over several experiences e.

Resource Allocation Parameters

δ_{ikjt}	Function that represents the allocation of a resource j to the operation o_{ik} at time t: 1 if is allocated, 0 if not (Kronecker delta).
p_{jik}	Price presented by operational holon that represents resource j to execute the operation $o_{ik.}$
C_b	Cost associated to the investiment done to buy the machine.
C_s	Cost associated to the execution of the setup.
C_p	Average cost of production per time unit (due to electricity, maintenance, etc).
σ	Price saturation coefficient.
β	Quotient between the actual load and the capacity of the resource (in case of transporter devices, this parameter represents the actual autonomy of the battery).
γ	Acceptance rate (in terms of number of operations allocated).
C_t	Cost associated to the acquisition of new tools for the execution of the operation.
C_{transp}	Average cost of transport per unit.
L_{actual}	Actual location of the transport device.

L_{source}	Source location of the workstation.
L_{dest}	Destination location of the workstation.
$bidRate$	Heuristic function to evaluate an operation proposal, aiming to select the best proposal.
Cdd_{jik}	Cost related to the due date, by resource j to executed the operation $o_{ik.}$
Cl_{jik}	Cost related to the location, by resource j to executed the operation $o_{ik.}$
Cq	Cost related to award partial quantities.
Cr	Confidence degree of the proponent operational holon.
$nWOdd$	Number of work orders that the operational holon did not fulfil the due date.
$nWOf$	Number of work orders allocated to the operational holon that were cancelled due to disturbances.
$nWOs$	Number of work orders executed by the operational holon with sucess.
w_p	Weight of the price value in evaluation of a proposal.
w_l	Weight of the location cost in evaluation of a proposal.
w_{dd}	Weight of the due date cost in evaluation of a proposal.
w_q	Weight of the partial quantity cost in evaluation of a proposal.
w_c	Weight of the confidence degree of the proponent operational holon.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"It is not the strongest of the species that survives,
nor the most intelligent, but the one most responsive to change."
Charles Darwin

In the last decades significant changes in the manufacturing environment have been noticed: moving from a local economy towards a global economy, with markets demanding for products with high quality at lower costs, highly customised and with short life cycle, leading to mass customisation. In parallel, the continuous evolution of technology often requires the updating and integration of existing systems within new supervisory environments, to avoid their technological obsolescence.

In this worldwide market competition, the companies can no longer be seen acting stand-alone, being forced to reconsider how they are organised. On one hand the companies tend to divide into small sub-companies, belonging or not to the mother company, each one having a specific business core, focusing on the production of a few specialised ranges of products. On the other hand the companies tend to share skills and knowledge, networking together to achieve global production.

The enterprise geographic expansion, through the geographically distributed implantation of factory plants, administrative facilities and sales offices, leads to the concept of distributed production systems, which has impact at all levels of the enterprise, from the inter-enterprise level to the shop floor level.

1.1 The Problem

Companies, to remain competitive, search to answer more close to the customer demands, by improving their flexibility and agility while maintaining their productivity and quality.

The agility and flexibility[1] are related with the capability of adaptation to the stochastic and volatile manufacturing environment. These competitiveness vectors require the ability to maintain goals in face of internal and external unpredictable disturbances. The weak reaction to disturbances, with new jobs arriving, certain resources becoming unavailable and additional resources being introduced to the system, leads to deviations from the initial plans and causes delays and no-operative situations.

Charles Darwin, in his book *"The Origin of Species"*, developed a theory of evolution where he believed that species change over a long period of time, evolving to suit their environment. He stated that the species that will survive to evolution and changes in the environment are not the strongest or the most intelligent, but those that are more responsive to change. Translating this theory to the manufacturing world, the companies better prepared to survive are those that better respond to emergent and volatile environments, by adapting dynamically their behaviour.

The traditional manufacturing control systems do not exhibit this capability of adaptation and evolution in terms of production control. In fact, the centralised and hierarchical control approaches present good production optimisation but a weak response to change, mainly because of the rigidity and centralisation of the control structure. On the other hand, heterarchical-like manufacturing control approaches present a good response to change and unpredictable disturbances, but due to the partial knowledge about the system, the global production optimisation may be degraded.

In these circumstances, the challenge is to develop manufacturing control systems with autonomy and intelligence capabilities, agile and fast adaptation to the environment changes, and more robust against the occurrence of disturbances, and easier integration of manufacturing resources and legacy systems.

Holonic Manufacturing Systems (HMS) [HMS, 2004] is one paradigm for the factory of the future, that translates to the manufacturing world the concepts developed by Arthur Koestler to living organisms and social organisations [Koestler, 1969]. Holonic manufacturing is characterised by holarchies of autonomous and cooperative entities, called holons, that represent the entire range of manufacturing entities. A holon, as Koestler devised the term, is an identifiable part of a (manufacturing) system that has a unique identity, yet is made up of sub-ordinate parts and in turn is part of a larger whole. The introduction of holonic manufacturing paradigm

[1]These concepts will be defined in chapter 3.

allows a new approach to the manufacturing problem, bringing the advantages of modularity, decentralisation, autonomy and scalability.

In spite of its promising perspective and the research developed by the holonic community, such as referred in [Christensen, 1994, Brussel et al., 1998, Bussman and McFarlane, 1999, Höpf, 2002, Brennan et al., 2002] and others compiled in [JASS, 2001, Deen, 2003], the holonic manufacturing achievements leave some important open questions: how to achieve global optimisation in decentralised systems, how to evolve the production control structure to adapt to change, how to specify formally the dynamic behaviour of holonic systems, how to introduce learning and self-organisation capabilities, how to integrate automation resources, how to develop holonic-based control applications, etc.

1.2 Objectives and Contributions

The objective of this dissertation is to contribute to the improvement of the manufacturing control systems performance, addressing the fast reaction to disturbances at the shop floor level, thus increasing the agility and flexibility of the enterprise when it works in volatile environments, characterised by the frequent occurrence of unexpected disturbances.

The thesis to be developed in this work can be expressed in the following statement:

The introduction of holonic production control, allowing the dynamic re-configuration of the control structure and the capability to have hierarchy in decentralised systems, improves the manufacturing system performance, with special emphasis in scenarios of change and unexpected disturbances.

To sustain this thesis, it is proposed an architecture, designated by ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems), which is based in the holonic manufacturing paradigm and in the following main foundations:

- Decentralised systems, comprising a community of autonomous and cooperative holons, taking advantage of modularity, decentralisation, agility, flexibility, robustness, scalability and components re-use.

- Supervisor entities to introduce hierarchy in decentralised systems.

- Self-organisation capability associated to the holons, which allows the dynamic evolution and re-configuration of the organisational control structure, combining the global production optimisation with the agile reaction to unpredictable disturbances.

Since architecture is the practice (science) of designing and building systems, by indicating the system components, their functions and their interactions, the ADACOR holonic architecture, as illustrated in Figure 1.1, comprises the ADACOR components, interactions, functional models and disturbance handling model.

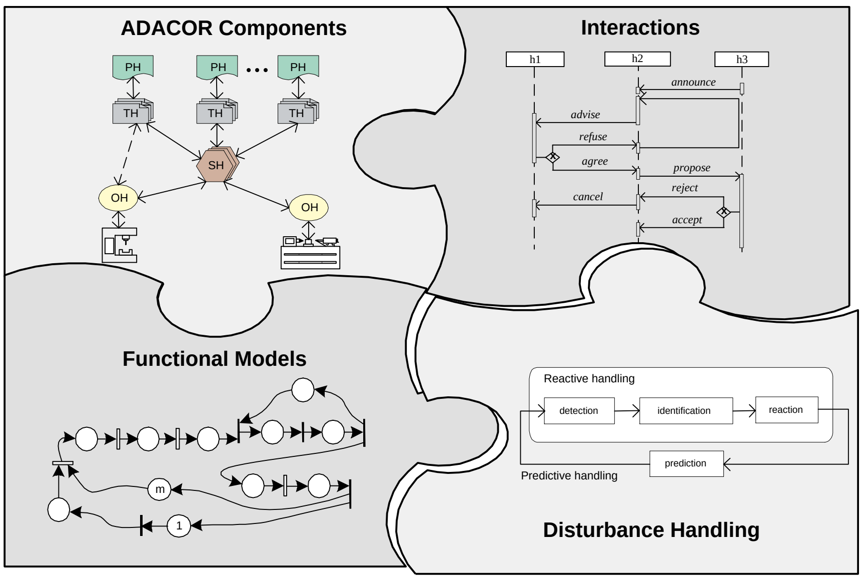


Figure 1.1: ADACOR Holonic Architecture

The first issue is related to the definition of the distributed and autonomous **components** belonging to the architecture, comprising the description of the types of holons and their characteristics, the architecture of a generic ADACOR holon, and how the adaptive production control is shared between ADACOR holons.

The second issue defines the **interactions** between ADACOR holons leading to the manufacturing control functions, i.e. the short-term process planning, scheduling and plan execution functions.

The third issue is concerned to **model the dynamic behaviour** of each type of ADACOR holon and the synchronisation between the individual models, and to the formal validation of those models.

The fourth issue is related to mechanisms for the distributed (re-)scheduling and dynamic **adaptation to disturbances**, supporting the agile, reactive and predictive response to the unexpect manufacturing disturbances.

All these issues together define a holonic control architecture whose main achievement is an adaptive production control approach that balances between a hierarchical architecture and a more flat architecture, in order to provide agility combined with global production optimisation.

1.3 Dissertation Organisation

This dissertation is organised in seven chapters, that initially describes the context, challenges and state-of-the-art of the distributed manufacturing control systems, then the proposed holonic manufacturing control architecture to face the described challenges, and at last the validation of the proposed architecture through the implementation into a case study.

In chapter 2, entitled *"Distributed Manufacturing Control Systems: A State-of-the-Art"*, the state-of-the-art of distributed manufacturing control systems is reviewed. Initially, the manufacturing systems are reviewed, with special attention to their classification and to the evolution of the manufacturing paradigms. Then, the flexible manufacturing is analysed, focusing on the Flexible Manufacturing Systems and Computer Integrated Manufacturing concepts.

The analysis of manufacturing control systems taxonomy, functions and requirements are also discussed, as well as some existing manufacturing control approaches, focusing on the hierarchical manufacturing control architectures and the agent-based manufacturing research work. At last, the holonic manufacturing paradigm and related research work is described.

In chapter 3, entitled *"An Adaptive Holonic Control Architecture"*, the ADACOR holonic manufacturing control architecture is presented, aiming to improve the agility, flexibility and reaction to unexpected disturbances at shop floor level. Along this chapter, the components of the system, the ontology used by the architecture components, the interactions between the components and the architecture of a generic ADACOR holon are described.

This chapter also introduces an adaptive production control approach that distributes the control between the different coordination levels, and balances between stationary (presenting a similar hierarchical control structure) and transient (presenting a quite similar heterarchical control structure) control states. To support this adaptive production control, the self-organisation concept inherent to each ADACOR holon is described, which is driven essentially by the autonomy factor and propagation mechanisms.

In chapter 4, entitled *"Modelling of Dynamic Behaviour of ADACOR Holons"*, the specification of the ADACOR holonic system is discussed. The dynamic behaviour of each ADACOR holon class is formally modelled using a Petri Net formalism tailored for production and control modelling purposes. Also in this chapter, the formal validation of the structural and behavioural specifications of the Petri net models elaborated for the ADACOR holon is performed, allowing to verify the correctness of these models and the system specifications.

In chapter 5, entitled *"ADACOR Disturbance Management"*, the several steps associated to the disturbance management are described, namely the detection of symptoms, identification of disturbances, mechanisms for reaction to disturbances and prediction of future unforeseen disturbances. A special attention is devoted to the prediction of occurrence of future disturbances that extends the traditional reaction mechanisms, which could minimise the unpredictable effects of the disturbance.

The chapter 6, entitled *"Implementation and Experimental Validation"*, describes the implementation of a prototype to validate the ADACOR concepts. Initially, a procedure to analyse and to evaluate manufacturing control systems, and the implementation of the ADACOR holonic control concepts into a prototype will be described, the last one focusing mainly the development platform and the implementation of the ADACOR holon classes.

The experimental case study used to test the ADACOR concepts is also described, defining the production system and manufacturing scenarios. At last, the experimental results will be analysed and some conclusions about the validation of the concepts proposed in the ADACOR architecture are elaborated.

Chapter 7, *"Conclusions and Future Work"*, rounds up the dissertation with the conclusions and the main contributions of this research work. At the end, it is elaborated an overview of planned further developments related to the approach presented here, being suggested some guidelines, within the context of future research trends in manufacturing control domain.

Additional appendixes describe some issues not considered as the major focus, but which may complement the understanding of the dissertation.

Chapter 2

Distributed Manufacturing Control Systems: A State-of-the-Art

"No thought is too old or too absurd to increase our knowledge."
Paul Feyerabend

Manufacturing systems involve activities related to the production of goods using manufacturing resources and knowledge, according to the external demands and subject to the environmental context, e.g. social and economic aspects. Nowadays, markets demand products with high quality at lower costs, highly customised and with short life-cycle.

In this scenario, the increase of competitiveness expressed in more productivity, more quality, more agility, more flexibility and better adaptation to unexpected disturbances is crucial for an enterprise staying in the business. Aiming to increase the competitiveness, some manufacturing enterprises tended to divide themselves into small sub-enterprises, belonging or not to the mother enterprise, each one having a specific business core, and being specialised in the production of a small range of products. The enterprise geographic expansion, through the geographical distribution of factory plants, administrative and sales offices, led to the concept of distributed production systems, which has impact at all levels of the enterprise, from the inter-enterprise level to the shop floor level.

More recently, in opposite to what we described above, the competitiveness is reached by cooperation between the enterprises. This situation provides the opportunity for Small and

Medium Enterprises (SME)[1] to improve their competitiveness within the global economy, participating in supply chains and forming virtual enterprises and e-alliances to fulfil specific customer demands.

Another way to achieve increased competitiveness is to use innovative technologies, through the introduction of industrial automation systems combined with information technologies. The choice, design and integration of adequate technologies in the system are essential since the introduction of emergent technologies by itself does not solve the problems. This trend is due to the great development of technologies that involve microprocessors, robots, numerical control machines, communication networks, artificial intelligence, etc.

The control system plays a critical role in increasing the performance parameters of a manufacturing system. Traditionally, these systems were implemented using centralised and hierarchical control approaches, presenting good responses in terms of throughput due to their production optimisation capabilities. In the actual manufacturing environment, the performance must also take in consideration the flexibility and agility of the control system.

The heterarchical control approaches introduces good response to the flexibility and agility requirements, but degrade the production optimisation. The current challenges are the development of manufacturing control systems that combine the hierarchical and heterarchical approaches, fulfilling the requirements imposed by the current manufacturing environment, using new paradigms for the factory of the future, such as holonic and bionic manufacturing.

The purpose of this chapter is to analyse and contextualise the distributed manufacturing control systems, reviewing their state-of-the-art. Initially, the manufacturing systems are analysed, and their classification and the historical evolution of the manufacturing paradigms are reviewed. Then, the concept of flexible manufacturing is presented, by defining the several types of flexibility found in manufacturing domain, and describing the available automation technologies and the flexible manufacturing system, computer integrated manufacturing and distributing manufacturing concepts.

Afterwards, the manufacturing control functions are reviewed, and the main requirements of the next generation of manufacturing control systems are introduced. Some manufacturing control architectures using traditional approaches are then presented. Agent-based manufacturing is considered next, and the basic concepts related to agent technology and some research work on agent-based manufacturing are reviewed. At last, the holonic manufacturing paradigm,

[1]The definition of SME is not unique or consensual, depending from country to country and from sector to sector. As example, the Commission of the European Communities considers that a SME is made up of enterprises which have fewer than 250 employees and which have either an annual turnover not exceeding 50 million euro, or an annual balance sheet total not exceeding 43 million euro.

which is one of the new paradigms for the factory of the future, is described, focusing on the description of the basic holonic concepts and the holonic manufacturing research work.

2.1 Manufacturing Systems

A production enterprise is an organisation whose core business is focused in the production of products. The production can be defined as the transformation process that converts raw materials or semi-finished products into finished products that have value in the market, using manual labour and machinery, and usually carried out systematically [Groover, 1987].

A production enterprise requires the integration of three main elements: product, process and business. The product vector is related to the product development and design activities, the process vector related to how to produce the products and the business vector is related to distribution, marketing and service infrastructure. The focus of this research work is in the process element.

2.1.1 Manufacturing Process Model

The production industries can be classified according to the type of products produced: manufacturing industries, which are typically identified with the production of discrete items that can be individual recognised, counted and defined in form, weight and features, as in the case of production of automobiles, computers and televisions, and process industries, which are typically identified with the production of goods involving a continuous production process, as in the case of production of energy and paper. In this work, the focus will be in the manufacturing industries requirements.

In an abstract level, the production process of a manufacturing industry, can be modelled considering a platform that comprises machinery, tools, knowledge and human labour, as illustrated in Figure 2.1. Nowadays, knowledge is a prominent production factor, together with traditional production factors, such as capital, labour and raw material.

The production process has as inputs raw materials, information and energy. The guidelines that support the decision of how to produce are the organisational strategies, product demands and external disturbances. The organisational strategies define the guidelines of production, such as the production type and the long/medium term production plan. During the transformation process, subjected to environment, quality and safety constraints, waste is generated due to the material transformation process, to the failures occurred in the machines and to the quality control rejections. The variation in product demand and the external disturbances requires the introduction of corrective actions in the planning and control system to maintain the production

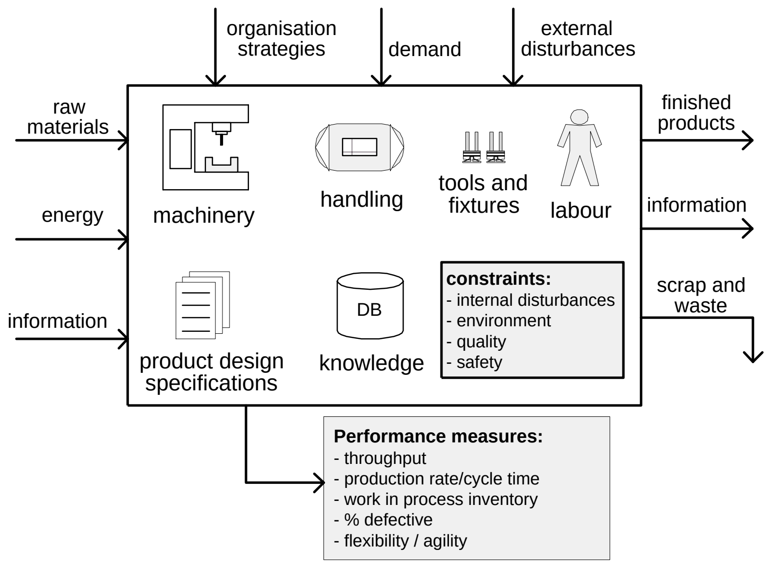


Figure 2.1: Abstract Model of a Manufacturing System (Adapted from [Black, 1991])

strategic guidelines.

The outputs of the production process are the finished products that will be delivered to the market according to the customer demands.

2.1.2 Classification of Manufacturing Systems

Manufacturing system can be classified according to production type, production layout and production volume.

The types of production, in terms of production orders, are usually divided into:

- make-to-stock, where the production is done for stock, based in forecast orders, such as in the high volume textile and shoe industry;
- assembly-to-order, where final products are only assembled after receiving a customer order, such as the automobile industry;
- make-to-order, where the production of the product starts after receiving a customer order, such as in the case of production of machine tools;
- engineer-to-order, which is an extension of make-to-order type, where one-of-a-kind products are designed and manufactured according to the customer specifications, such as in the space electronics.

A manufacturing system can also be classified according to the production volume. Under this vector it is possible to find three production types [Groover, 1987]: job shop, batch and mass production.

The job shop production is characterised by the production of small quantities, often one-of-a-kind, of a great variety of products. Typically, the equipment used in this production type must be flexible and general purpose to support the great variety of products. As examples of job shop production, it is possible to mention the production of machine tools, molds and aircrafts.

The batch production involves the production of lots of medium sized quantities of the same product, that has a regular but not so high demand. The equipment used in batch production is general purpose specially designed for higher production rates, for example a tool machine equipped with special fixtures designed to increase the machine production rate. As examples of items produced using batch production it is possible to find electronics equipments and furniture manufacturing.

The mass production is related to the specialised production of one or a small number of products, each one with high production rates. The equipment and the factory plant used in mass production is completely dedicated to the production of a particular product. Examples of items produced using this type of production are screws and light bulbs.

Another possible classification of a manufacturing system is according to the physical plant layout. The factory plant layout refers to the disposition of the physical facilities in a production plant. In manufacturing industries, producing discrete items, it is possible to find three main production layouts, Figure 2.2: fixed position, product flow layout and process layout.

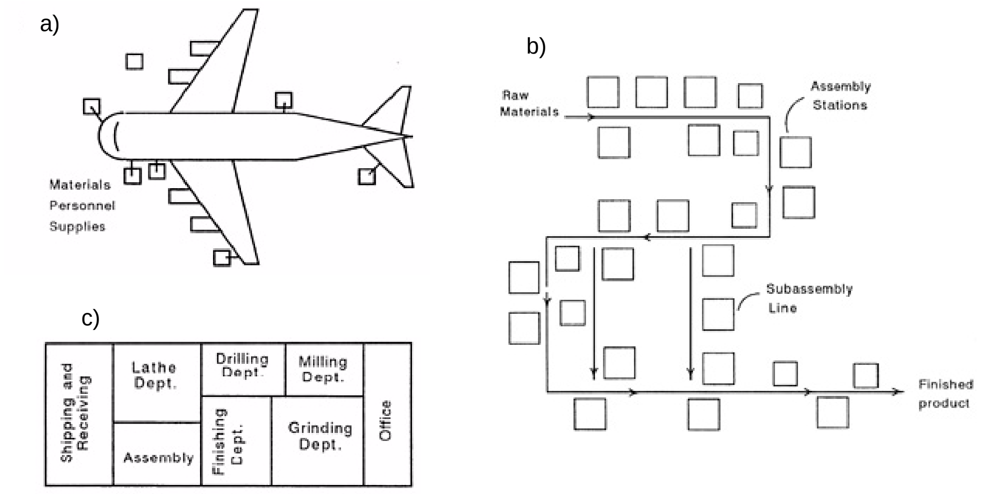


Figure 2.2: Types of Production Layouts: a) fixed position layout, b) product flow layout, c) process layout (From [Groover, 1987])

In the fixed position layout, the product is fixed due to its size or weight, and the machines and operators go to the product to execute the operations needed to produce that product.

In the product flow layout[2], the equipment are placed through a product flow line, in order to minimise the transport time between machines. The transport between the workstations can be done manually, using automatic conveyors, robots or AGVs (Auto Guided Vehicle). This layout presents a reduced material transport effort, small work in progress and simple production planning and control system, being adequate to the mass production type. However, it presents small flexibility in product changes and requires high investment, due to the need to duplicate equipment through the flow line.

In the process layout[3], the machines are grouped according to the manufacturing process, i.e. grouping the machines that can execute similar operations. The parts visit the several groups according to the specified operation sequence. This layout is adequate for the job shop and batch production types, presenting good flexibility and low investment in equipment with no need to duplicate equipment. However, it presents as disadvantages low efficiency in the material transportation and higher complexity in the production planning and control systems.

2.1.3 Historical Evolution of Manufacturing Paradigms

The manufacturing environment is in permanent change adapting to the customer demands, advances in information and automation technologies and economical trends. During the last century, several paradigms and organisational concepts were introduced aiming to bring more competitiveness to the production enterprises.

According to the Merriam-Webster dictionary [Merriam-Webster, 2003], a manufacturing paradigm can be defined as a philosophical and theoretical manufacturing framework of a scientific school or discipline within which theories, principles, laws, generalisations and experiments are formulated.

Before the 20th century, the craft production was the dominant type of production, characterised by skilled workers that used general purpose tools to produce exactly what the customers asked for, being the production level close to the *one-at-a-time*. As a historical example, a sword maker only produced one sword at a time, each one customised for the client, with periodic assessments of weight and balance, appearance, and details prior to delivery.

The industrial revolution introduced machinery in production, helping in the first phase the craft production to be more productive, by using machinery to support some craftsman work.

In the beginning of 20th century, Henry Ford decided to build a car that everybody could

[2]Also known as flow line.

[3]Also known as job shop or cellular shop.

own and drive. However, at that time most cars were customised for the client or built one at a time in limited quantities, following the craft production type. Based in the Taylor's theories, he introduced in 1913 at Highland Park plant in Michigan, the revolutionary concept of mass production, characterised by the production of the same product in large scale using a rigid assembly line to produce a car composed by identical interchangeable parts, Figure 2.3. At the time, everybody could have a Ford T car of any colour as far as it was black!

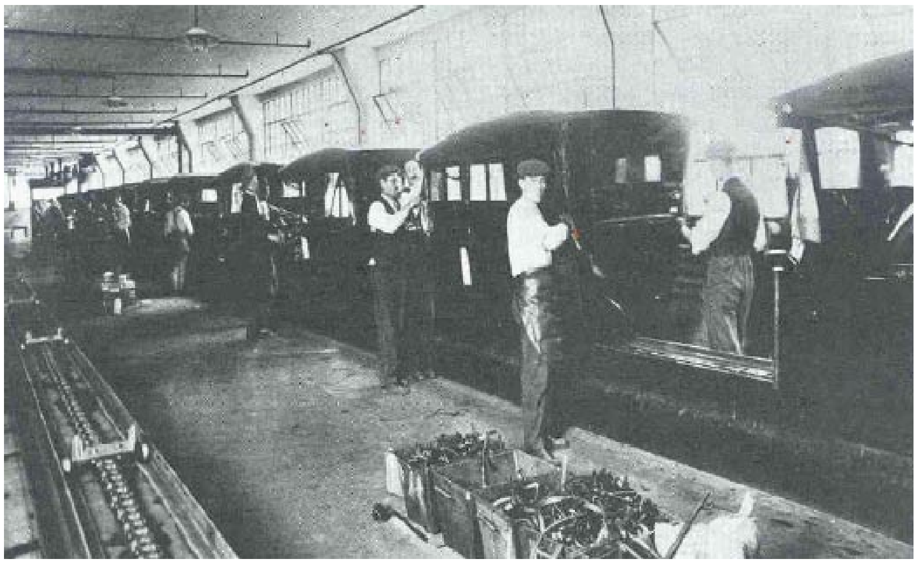


Figure 2.3: Ford's Assembly Line

With the introduction of the production assembly line, the task cycle for the average Ford assembler (i.e. the amount of time that the operator works before repeating the same operations), was reduced from 514 minutes to 2,3 minutes [Womack et al., 1990]. Lately, he had further cut the time from 2,3 minutes to 1,2 minutes with the moving line which brought the car to the stationary worker.

The mass production model requires stability and control in the input variables, markets, and the labour force. In the 1970s and 1980s, these parameters became less stable, with common economic fluctuations, increase of the consumer power, homogeneity of the market eroded and start of the global competition. Additionally, the globalisation of markets brought to the production companies the need to become strong competitively, in order to fulfil the requirements of the market for the reduction of prices, better product quality, minimum time of delivery and diversity of offer. The mass production, idealised by Henry Ford was a strap down system, incapable to treat variations in the type of product. This rigidity started to be an obstacle and with the world-wide competitiveness the mass manufacturing became viable only for some products.

The Just in Time (JIT) philosophy was introduced by Japanese company Toyota Motor

Inc. in its production system, supervised by its engineer chief Taichii Ohno, after studying the production of the Ford model A car [Womack et al., 1990]. JIT consists in having the right material at right place at right time, eliminating stocks, and using very simple control and scheduling systems, such as the Kanban cards system. The implementation of JIT principles had supported the Japan's ascendancy in the automotive world.

Since the late 1970's the Toyota Production System has been studied, specially by US companies, and their principles have been gathered together as the basis of the lean production concept [Womack et al., 1990], which main idea is to eliminate waste in all activities, achieving manufacturing products with less of time to design, less inventory, less defects and reducing the setups.

A concise definition of lean production is *"Lean Production is a system of work organisation that strives to deliver high quality and low cost products through the efficient use of resources and the elimination of waste"* [4].

An example of lean production is an automobile production line with capability to produce several variants of a car to meet the demands of a specific market segment.

In the eighties, the companies addressed technologies and paradigms to achieve flexibility. However, in the nineties they were challenged by the need to increase agility. The agile manufacturing, introduced by the Iacocca Institute at Lehigh University, is the ability to adapt quickly and profitably to continuous and unexpected changes in the manufacturing environment [Kidd, 1994]. It presents continuous improvement, rapid response, quality improvement, social responsibility, and total customer focus.

Agility impacts the entire manufacturing organisation, including product design, customer relations and logistics, as well as production, and it has been expressed as having four underlying principles [Goldman et al., 1995]: deliver value to the customer, ability to react to changes, value of human knowledge and skills, and ability to constitute virtual partnerships. While the first three principles can be found in the lean production paradigm, the fourth principle makes the difference between lean and agile manufacturing: in agile manufacturing the companies form temporary alliances with other companies, even competitors, to react to unexpected situations, with mutual benefits for both companies [Kidd, 1994]. Finally, in agile manufacturing it is important to consider that human factors and organisational knowledge are just as important as advanced technology. Additionally, work within agile organisation occurs concurrently rather than sequentially.

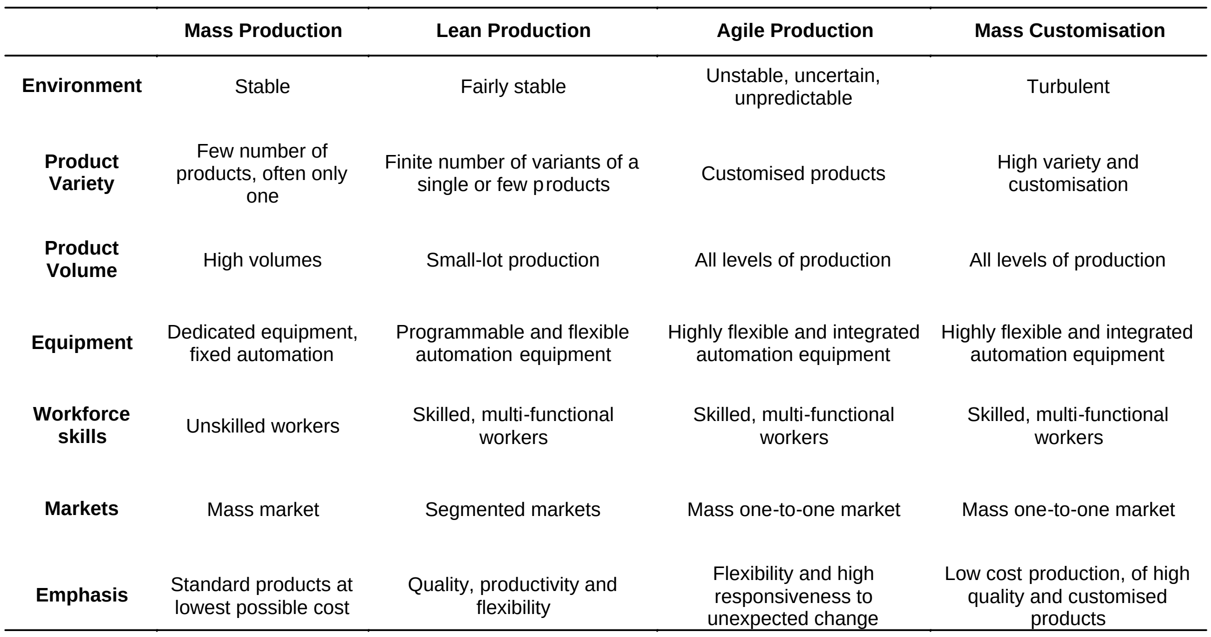
The ideas of lean and agile manufacturing are mainly viable for companies starting from

[4]http://www.lir.msu.edu/piers/programs/leanproduction.htm

scratch its business or companies prepared to support the significative financial risk of build a new production system. Those risks are acceptable for large companies, like Toyota or Bull, but they are often unaffordable for SMEs.

Nowadays, the customer demand for specific and customised products leads to the concept of mass customisation, which focuses on satisfying the individual customer needs. While in lean production the focus as in the elimination of the waste in the process, in mass customisation the focus is in the elimination of the waste in products by eliminating the features unwanted by customers.

Mass customisation requires the increase of flexibility and agility, using flexible processes (automation technologies, such as CNC machines, robots and CAD systems) and flexible organisational structures to produce multiple variations of often customised products at the price of standardised mass products. As an example, Ford's truck plant in Kentucky offers 2,5 million variant of trucks to its customers [Swamidass, 2000]. Another illustrative example is related to the Smart car manufacturer, which offers more than a half million different configurations to its customers[5].



	Mass Production	**Lean Production**	**Agile Production**	**Mass Customisation**
Environment	Stable	Fairly stable	Unstable, uncertain, unpredictable	Turbulent
Product Variety	Few number of products, often only one	Finite number of variants of a single or few products	Customised products	High variety and customisation
Product Volume	High volumes	Small-lot production	All levels of production	All levels of production
Equipment	Dedicated equipment, fixed automation	Programmable and flexible automation equipment	Highly flexible and integrated automation equipment	Highly flexible and integrated automation equipment
Workforce skills	Unskilled workers	Skilled, multi-functional workers	Skilled, multi-functional workers	Skilled, multi-functional workers
Markets	Mass market	Segmented markets	Mass one-to-one market	Mass one-to-one market
Emphasis	Standard products at lowest possible cost	Quality, productivity and flexibility	Flexibility and high responsiveness to unexpected change	Low cost production, of high quality and customised products

Figure 2.4: Comparision between Manufacturing Paradigms

Figure 2.4 presents a brief comparison of the main manufacturing paradigms, resuming the characteristics of each one under different categories, such as the product volume, product

[5]Smart is an European car manufacturer, 100% owned by Daimler-Chrysler, which offers to its customer the possibility to configure the car that he wants to buy, using the Internet (hppt://www.smart.com/).

variety, product cost and workforce skills. The significant differences are reflected in the emphasis of each paradigm and is illustrated in Figure 2.5, that shows the evolution of them in terms of product variety and volume.

In spite of the current trend to mass customisation, it is possible to find several examples of mass production, such as the Telco 1010 truck production line in India, which produces thousands of trucks with no variation in design [Swamidass, 2000]. Also, the craft production is nowadays visible in the cases where artisans build products for the customer specifications in specific market niches, and often sell in traditional markets.

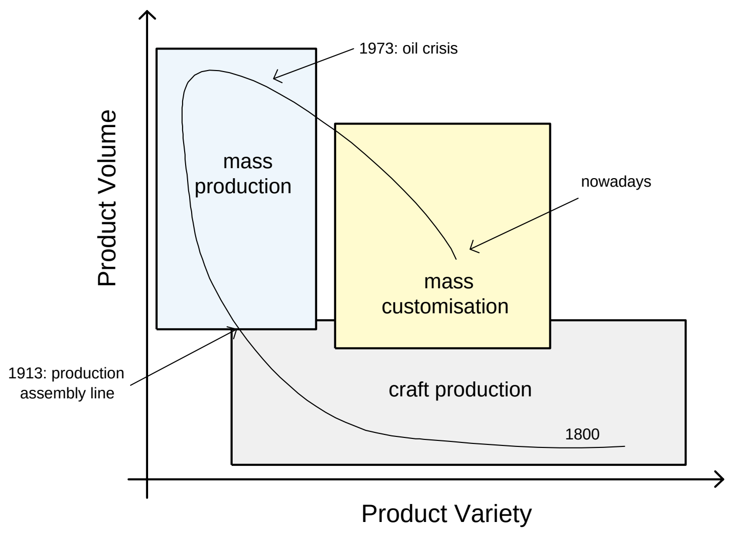


Figure 2.5: Evolution of Manufacturing Paradigms

The conclusion is that in the 21st century, companies are going to operate in a dynamic and challenging environment that requires new approaches to manufacturing. Mass customisation is a general trend that is more and more widespread, seeming to be as the production paradigm for the factory of the future. From the manufacturing point of view, much work must be done to develop adequate manufacturing systems meeting the new requirements, since traditional solutions don't seem to be able to face the demands of mass customisation.

2.2 Flexible Manufacturing

Manufacturing enterprises have two basic alternatives to face the problem of a variable and customised demand: or build manufacturing plants with excess capacity and stock of products in inventory to smooth fluctuations in demand, or increase the flexibility of their manufacturing plants, to deal with the production volume and variety. In spite of its implementation complexity, the use of flexibility in manufacturing is the preferred solution.

2.2.1 Types of Flexibility

According to the Merriam-Webster dictionary, flexibility is the capability to adapt to new, different, or changing requirements. Several flexibility classifications are presented in the literature, as discussed by [Gerwin, 1993, Browne et al., 1984], identifying different types of flexibility, such as mix, changeover, volume, product and sequencing.

The mix flexibility is concerned with the capability of a system to handle a range of products or variants, supported by the execution of fast setups in the process. As example of high mix flexibility, a mobile phone producer that offers 8 different models, each one allowing to choose one from 6 different colours.

The changeover flexibility is the capability to change quickly the production system in order to be able to offer new products. As an example of low changeover flexibility, in 1926 the Ford Motor Co. shut down its production for an entire year when changing from production of the Model T to the Model A.

The volume flexibility is the capability to deal with production volume variability, facing the demand. An example of low volume flexibility is the automobile and shoe industries: the variation of the production volume requires the increase or decrease of the number of shifts.

The product flexibility is the capability to modify rapidly the product design. As an example of high product flexibility, a company manufacturing general purpose machine-tools has normally the ability to change slightly the features of the product (such as the material type and dimensions).

The sequencing flexibility is the capability to support alternative sequences for the production plan execution by using resources that have capability to execute different operations, organised in a proper way. A dedicated assembly line is typically an example of low sequencing flexibility and a manufacturing system using a cell-based layout is typically an example of high sequencing flexibility.

The improvement of flexibility is essentially achieved by using programmable automation technologies, such as computer numerical control machine-tools, flexible manufacturing systems and computer aided technologies. Other factors, like a well trained workforce and the production process design, also contribute for the increase of flexibility.

2.2.2 Flexible Automation Technologies

The major automation technologies in this area are the programmable logic controllers, industrial robots, numerical control machines, automatic guided vehicle systems and automatic storage systems.

PLC stands for Programmable Logic Controller, and it is an electronic and programmable device, to control and monitor the production processes in industrial environment and in real time. The first PLC was introduced in the late 1960's, to replace complicated relay-based control systems.

The industrial robots, Figure 2.6, allow executing repetitive operations normally performed by human operators. The word *Robot* is derived from a satiric theatre play, written by Karel Capek in 1921, who used it to designate *labour force*. There are several definitions for robot, such as the one from the American Robotic Institute, which defines *"a robot as a reprogramable and multi-functional manipulator, designed to handle materials, parts, tools or special devices, in variable movements programmed to execute several tasks"*, and [Groover, 1987] that defines industrial robot as a *"multi-application and programmable machine, having certain anthropomorphic features"*.

The introduction of robots allows to increase the productivity (with no interruptions, absenteeism, etc.) and to increase the robustness, speed and resistence to hostile environments. The main industrial applications of robotic systems are painting, welding, assembly or material handling operations.

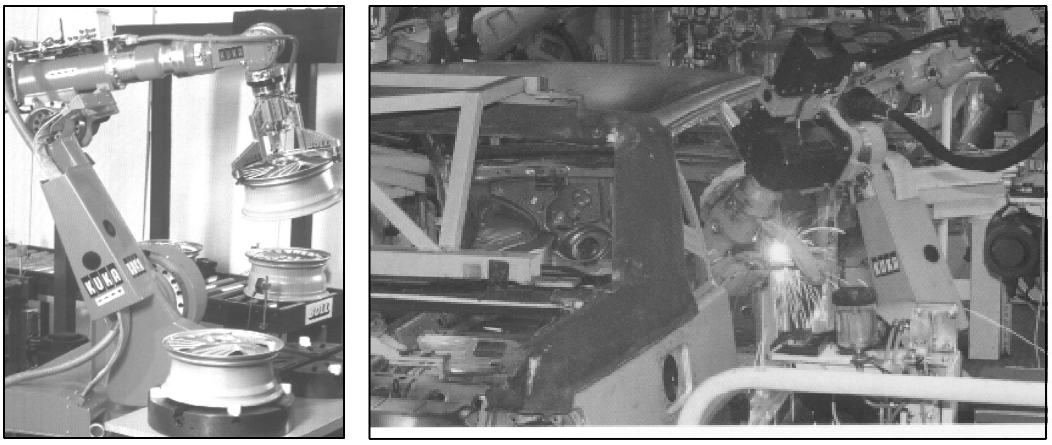


Figure 2.6: Industrial Robot in Materials Handling and Welding Tasks

The numerical control (NC) machines, Figure 2.7, which appeared in industry in the fifties after a successful demonstration in the Massachusetts Institute of Technology (MIT), is a form of programmable automation in which the processing equipment is controlled by programs that allow the execution of sequence of complex operations inside the machine, without the help of human operators. A NC machine is, in functional terms, similar to a conventional machine, with differences related to the way as the machine functions and movements are controlled.

These machines have the ability to execute different machining operations, with an appro-

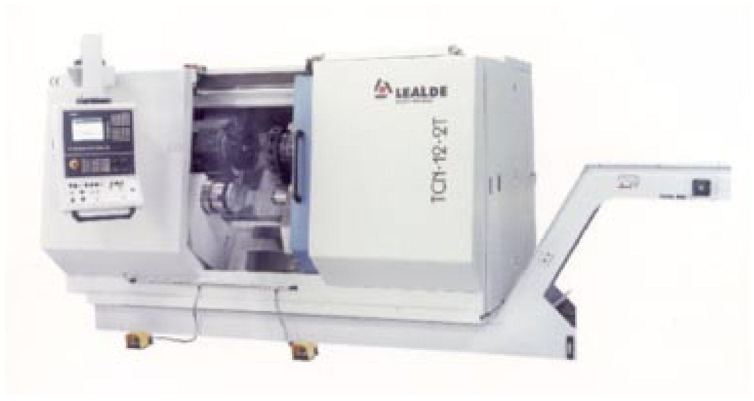


Figure 2.7: Computer Numerical Control Machine Tool

priated set-up and a machining program, supported by an internal tools magazine. The use of numerical control machines allows increasing the quality of produced products and the productivity. Lately, in the seventies, appeared the Computer Numerical Control (CNC) machine tool that uses dedicated micro-computers as control unit and have capabilities for storage and local edition of several machining programs, interpolation execution, compensation of tools dimension, and communications interface.

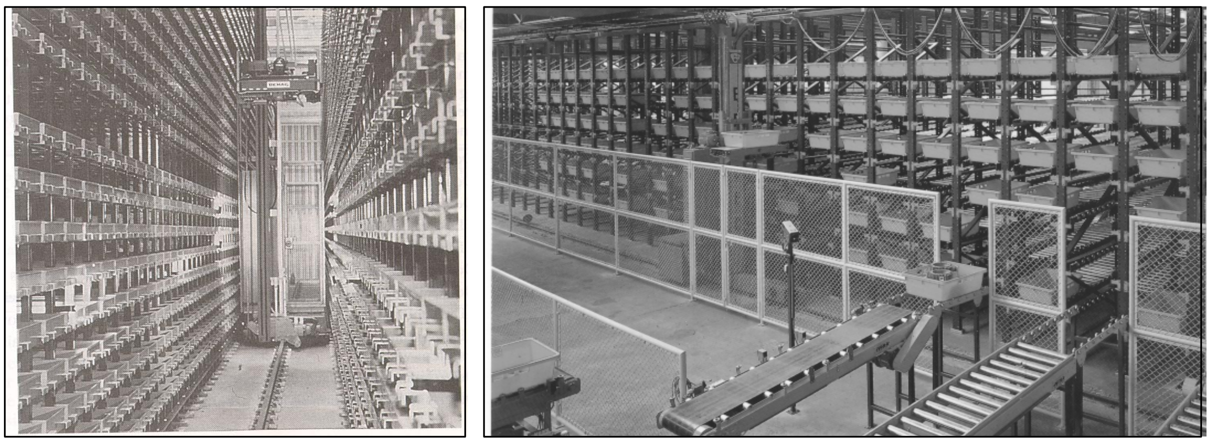


Figure 2.8: Automatic Storage and Retrieval Systems

An AGV is an intelligent, flexible and versatile material transportation system, with programming capabilities for motion, path selection and positioning. Each vehicle is powered by batteries, and is controlled by a microprocessor, which is used to guide the vehicle to follow a pre-defined path, and to correct the trajectory if the vehicle deviates from the initial path. An AGV can load and unload automatically materials, and can be modified, adapted and re-designed in order to operate in hostile environments. An AGV is capable to move in one or both directions of a path and travel between several buildings of the same plant (opening and closing automatically the doors) or between several floors (using lifts).

An Automated Storage/Retrieval System (AS/RS), Figure 2.8, is a system for temporary storage of materials, where the movements are executed by automatic devices and the management executed using information technologies, without human intervention. The AS/RS systems are widely used in industry, to store raw materials, intermediate parts, finished products, tools and grippers, parts for recover and waste, work in progress and office material. The use of AS/RS systems allows increasing the storage capacity through the growth in height and the narrowness of the aisles, the optimisation of the storage spaces and volumes (illumination and heating economy, etc.) and the integration in the logistic chain, avoiding ruptures between the flows, and management in real time.

2.2.3 Flexible Manufacturing Systems

The Flexible Manufacturing System (FMS) concept combines the efficiency of product flow layout with the flexibility of process layout, allowing to reduce the transport time, the investment in equipment and the setup times, Figure 2.9. A FMS comprises a set of work stations, interconnected by a transport and material handling system, and controlled by a integrated computational system [Groover, 1987, Upton, 1992].

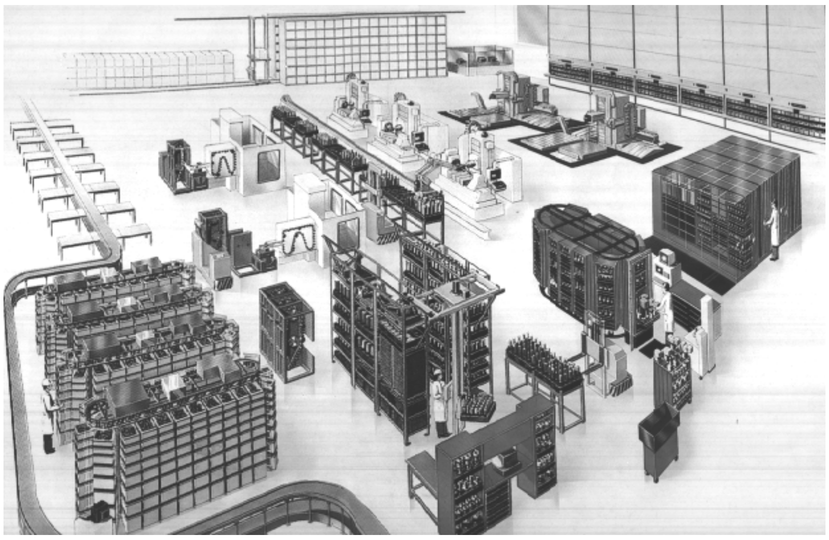


Figure 2.9: Flexible Manufacturing System

As illustrated in Figure 2.10, these systems fill the gap between the mass production and the dedicated NC machine production, with the ability to process simultaneously a variety of different part types.

The FMS systems present several significant advantages [Rembold et al., 1993, Ranky, 1990]:

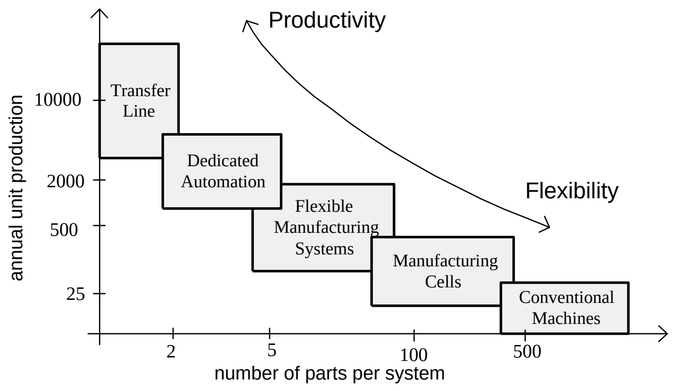


Figure 2.10: Flexible Manufacturing Systems Context [Upton, 1992]

an increase of productivity (2-3,5 times), decrease of production costs (50%), reduction of inventory and stocks (85%), increase of quality, decrease of response time, products customised to the customer requirements, etc. The reduction of inventory can be enough to justify the investment in necessary hardware and software to build a flexible manufacturing system [Upton, 1992].

In spite of the main objective of the FMS being to achieve the flexibility, one of the main disadvantage is its inflexibility to the introduction of new products. FMS are flexible while they are producing a known range of products, becoming difficult the introduction of new product families, due to the complexity of automatically execute the necessary adjustments.

FMS can present several layouts: flow line, loop, ladder, openfield and centered robot. In the flow line configuration, the materials flow between the workstations disposed in a line, being appropriated for systems in which the material progress is well defined. In loop configuration the materials flow between stations, such as the previous configuration, with the difference that the input station is coincident with the output station. The ladder configuration is similar to the loop configuration, presenting the advantage of alternative paths, in order to reduce the transport times.

The openfield configuration divides the shop floor in cells, each one is responsible for the execution of a specific range of operations. The workstations or autonomous cells are normally manufacturing cells (that comprises mainly NC machines and robots), assembly cells, inspection cells, etc. The formation of autonomous cells is done according to the group technology concept, which is characterised by grouping the items in families with similar features[6].

At last, the centered robot configuration is typically used in applications where the robot is the central element of the production process, executing all the material handling tasks.

[6]There are two types of features to be considered in group technology: drawing attributes, such as geometry and size, and process attributes, such as the sequence of operations to produce the product.

2.2.4 Computer Integrated Manufacturing

The automatisation of the production activities to solve partial and specific problems, in a stand-alone way, creates automation islands, which leads to information redundancy and to a non-optimisation of the resources usage. The solution for this problem requires the need to integrate those automation islands.

The Computer Integrated Manufacturing (CIM) paradigm, popular in the eighties, consists of the integration of the enterprise activities, related with the production, through the use of information technologies, such as databases and networks, which allows the exchange and sharing of data [Rembold et al., 1993].

In the beginning, the integration only dealt with the engineering and production activities, and evolved in a next step to all activities related with the production. The last step was to integrate the enterprise systems with the suppliers and customers systems.

The main advantages of the CIM paradigm can be listed in the following [Ranky, 1990, Rembold et al., 1993]:

- **Increase of productivity**: the elimination of information redundancy leads to a better management and control of the resources, with improvements of 40 to 70%.
- **Increase of flexibility**: due to the information sharing it is possible the decentralised control leading to a faster response to external and internal disturbances.
- **Increase of quality**: the integration of automatised systems allows reducing the number of failures due to the guarantee of no duplication of information. The integrated management allows the execution of quality control, retaining immediately the products with defects. With CIM system it is possible to increase 2 to 5 times the quality.
- **Reduction of design time**: sharing the information by the several teams responsible for the product design allows a reduction of 15 to 30% in the design time.
- **Reduction of the work in progress (WIP)**: an optimised management using the information integration, allows a reduction of 30 to 60% of the work in progress.

The CIM paradigm also aims the integration of several computer-aided technologies that support the production systems, such as Computer Aided Design (CAD), Computer Aided Engineering (CAE), Computer Aided Manufacturing (CAM) and Computer Aided Process Planning (CAPP).

The CAD technologies use computational resources to aid the design activity, using specialised graphical systems, to create, update and document a design project in terms of engineering. The usage of CAD tools allows the increase of project design productivity, an easy visualisation of the projects and their components (for example the project drawings), a reduction of the development time, an increase of the design quality, and a re-use of old developed projects.

Using CAE technologies, such as a Finite Elements Analysis tool, the analysis and evaluation of the mathematic models created during the design, make possible to verify if the product obey to the demanded mechanical and structural characteristics.

The process planning acts as interface between the project and manufacturing phases, through the specification of manufacturing process details. The CAPP technologies support the definition of the sequence of operations (e.g. processing, assembly and inspection), necessary to produce the product. The main steps in the elaboration of the process plan are: raw material selection, determination of the operations sequence, selection of the type of machines that will execute the operations, selection of tools, fixtures and inspection equipments, determination of machining parameters (such as cut speed, feed rate and cut deep), and determination of manufacturing times (setup times, processing times, manufacturing time).

The manual elaboration of machining programs is a task extremely tiring that takes long time and is susceptive of errors due to human faults. The CAM technologies allow the automatic generation of machining programs, using a post-processor previously configured for each machine. The use of these tools allows the faster development of machining programs and the reduction of design errors.

The concurrent engineering concept aims to reduce the time to produce a product respecting the quality and due date specifications, and requires a parallel and cooperative approach to the design of the product and process, using computer aided tools, in opposite to the traditional design practices, which are sequential.

The concurrent engineering presents several benefits, the more important being the reduction of the manufacturing costs that can reach 50% and the reduction of product design time in 50%. As an example, Rolls-Royce using concurrent engineering reduced the time to develop its engines by 30% and reduced the weight in some instances by 25%.

The CIM paradigm is not the sum of these components but the integration of them into an operating system that satisfies the enterprise business strategies and objectives.

In spite of its objectives and described advantages, the implementation of the CIM concept didn't achieve good results, due mainly to the technological, heterogeneity, social and economical problems [Leitão, 1996].

The technological problems are related to the complexity to automatise and integrate some processes. The heterogeneous problems are due to the proprietary protocols from each equipment and technology, making more complex the integration of different systems. The implementation of the CIM concept is very expensive bringing economical problems. The social problems appear because normally the introduction of automation causes or seems to cause the increase of unemployment rate and generates new jobs that can not be taken by these workers. Additionally, and due to the CIM centralised approach it is difficult to expand and reconfigure a process for new products.

2.2.5 Distributed Manufacturing

Manufacturing industry by the end of 20th century was characterised by dynamic enterprises operating in a global scale, each one being made up of a number of autonomous production units or facilities cooperating among themselves. A distributed manufacturing system can be defined as a production system that is geographically distributed, acting in a cooperative way in order to work as a whole.

Supply chain is one concept associated to distributed manufacturing, which deals with the management of materials, information and financial flows in a network, consisting of suppliers, manufacturers, distributors and customers. The goal of supply-chain management is to have the right product in the right place, at the right price, at the right time, and in the right condition [Harrison, 1992]. Supply chains span from raw materials, to manufacturing, distribution, transportation, warehousing, and product sales. As the responsibilities are divided into different enterprises, the maintenance of a continuous control of the production flow is more complex.

The worldwide market competition implies that manufacturing enterprises can no longer be seen acting stand-alone or within rigid supply chains, forcing them to reconsider how they are organised. The Virtual Enterprise is a paradigm that can be defined as a temporary alliance of enterprises that come together to share skills and resources in order to better respond to business opportunities and whose co-operation is supported by computer networks [Camarinha-Matos and Afsarmanesh, 1999]. The term Virtual Enterprise is used because in spite of having the attributes of an enterprise, they are not a permanent organisation.

Nowadays, namely in automobile industry, other forms of enterprise organisation are emerged, such as the Extended Enterprise, e-Alliances or Smart Organisation.

A car OEM (Original Equipment Manufacturer) and its 1st tier suppliers constitutes an Extended Enterprise, in the sense that most of the component design, manufacturing and sometimes assembly is in charge of these suppliers. Second tier suppliers are now organising themselves as e-alliances, either because their customers need to reduce their suppliers base, or because they joint together by their own initiative to take advantage of a larger scale. Smart organisation is an enterprise that has design and/or manufacturing capabilities and works for multiple customers with strong requirements in terms of e-business systems and CAD systems.

The several types of distributed manufacturing systems present structures and features, that can be modelled and related through a layered approach, illustrated by Figure 2.11, which exhibits a fractal structure [Leitão and Restivo, 1999].

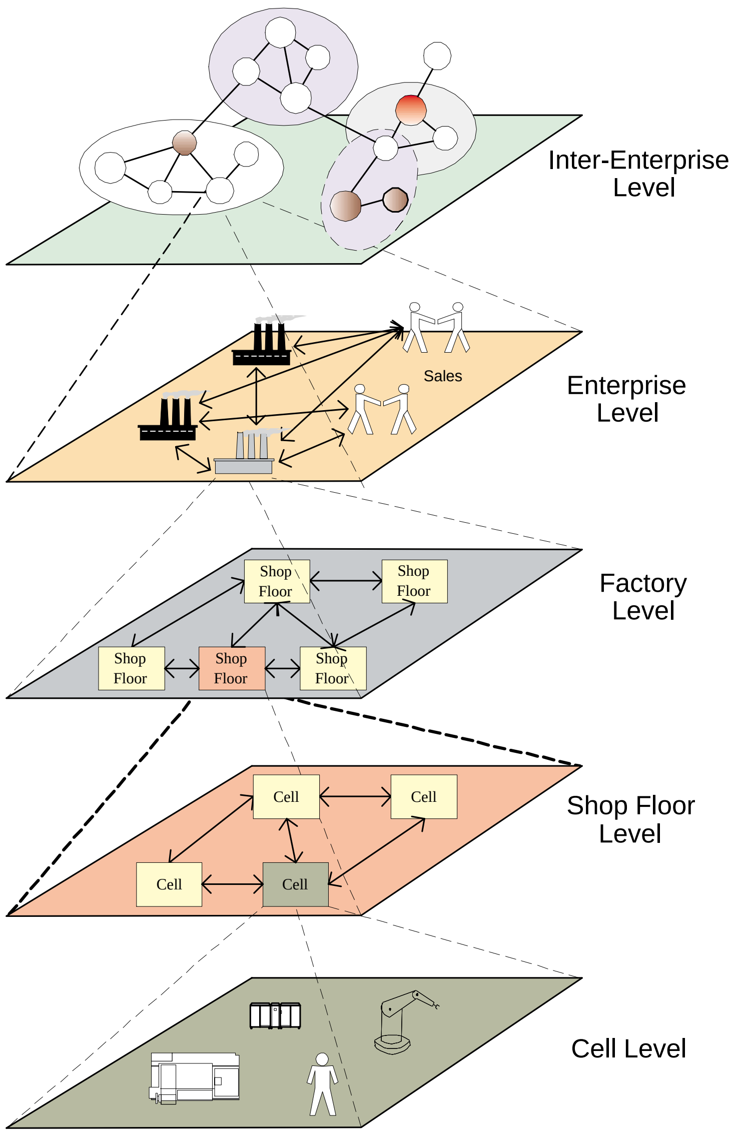


Figure 2.11: The Layer Approach to Distributed Manufacturing

This model represents fractal layers with similar interaction models, but different actors and requirements. The lower the layer is the higher are the temporal restrictions and the complexity of integration with physical resources. On the other hand, the higher is the level the higher are the inter-operability problems and the need for common ontologies.

The highest layer, the **inter-enterprise** layer, represents the interaction between distributed enterprises, acting together in order to achieve a common objective. A similar scenario is found within each manufacturing enterprise. Zooming into an enterprise shows another distributed manufacturing layer, the **enterprise** layer, where it is possible to find the co-operation between geographically distributed entities, such as sales offices and production sites, in a multi-site environment. Zooming down a production site leads to the **factory** layer, where the distributed manufacturing control within the site can be found. In this layer, the entities are distributed through shop floor areas, working together and in co-operation, in order to accomplish all the orders allocated to the production site. Zooming down a shop floor area shows the **shop floor** layer, where it is possible to find the co-operation between different cells, such as manufacturing, assembly and transport, organised in a flexible manufacturing system. Finally, a similar environment is found in the **cell** layer, with the interaction between equipments and humans belonging to a cell.

2.3 Manufacturing Control Systems

Control is a key factor in an automated production system, being required at different stages: low and high level.

At the low-level, the automation devices, such as industrial robots and NC machines, require control techniques that regulate its behaviour according to a specific objective. At this level, time-based control techniques such as Proportional, Derivative, Integral and On/Off techniques, which are commonly used either alone or in some combination like the well known PID (Proportional, Integrative and Derivative), and even intelligent control techniques, such as fuzzy logic, are often used to design and implement appropriate control algorithms on digital or analog devices. The control functions presented at this level are not discussed in this document.

The high-level control is concerned to coordinate the manufacturing resources activities aiming to produce the desired products, such as in the case of flexible manufacturing control systems. Algorithms at this level are used to decide what to produce, how much to produce, when production is to be finished, how and when to use the resources or make them available, when to release jobs into the factory, which jobs to release, job routing, and job/operation sequencing [Baker, 1998].

2.3.1 Terminology

At this point it is necessary to clarify some different terms used indistinctly to refer this high-level factory control.

The shop floor control is concerned with the problem of monitoring the production progress of the product, as it is being processed, assembled, transported and inspected in the factory. Despite the several definitions that ranges from the simple monitoring to those that includes scheduling, dispatching and execution (monitor and reaction), in this document will be used the last definition. Shop floor control is often referred also as Manufacturing Execution Systems (MES).

The manufacturing control is concerned with managing and controlling the physical activities in the factory aiming to execute the manufacturing plans. Normally, a manufacturing control comprises the short-term process plan and the shop floor activities.

The production control encompasses manufacturing control of one or several plants, purchasing, material requirements planning, design, medium and long-term process planning, and other production activities. The production control is also referred as the production planning and control (PPC) systems.

The focus of this document is in the manufacturing control, that is responsible to control the physical execution of the manufacturing plans, provided by the manufacturing planning activity, and to monitor the progress of the product as it being processed, assembled, moved, and inspected in the factory.

The manufacturing control systems comprise the following main functions, as represented in the Figure 2.12:

- Process related functions.
- Scheduling.
- Plan execution (dispatching, monitoring and reaction to disturbances).
- Pathological state handling (deadlock handling, etc).

The process related functions address mainly the short-term process planning, which is highly dependent of the type of manufacturing process, such as assembly, machining, sheet metal processing and continuous process.

The scheduling is concerned with the assignment of operation to resources, within a shorter temporal horizon and respecting a specific criteria. The scheduling algorithm uses the manufacturing plans provided by the short-term process planning and the capacity of the available resources at the factory plant.

The plan execution performs the final assignment of resources to the orders based on the current state of the manufacturing system and the schedule plans. The execution of manufacturing plans is subject to deviations, due to machine failures or parts delayed by suppliers. In this case, it is necessary to take the necessary corrective actions to complete the order on time and minimising the disturbances.

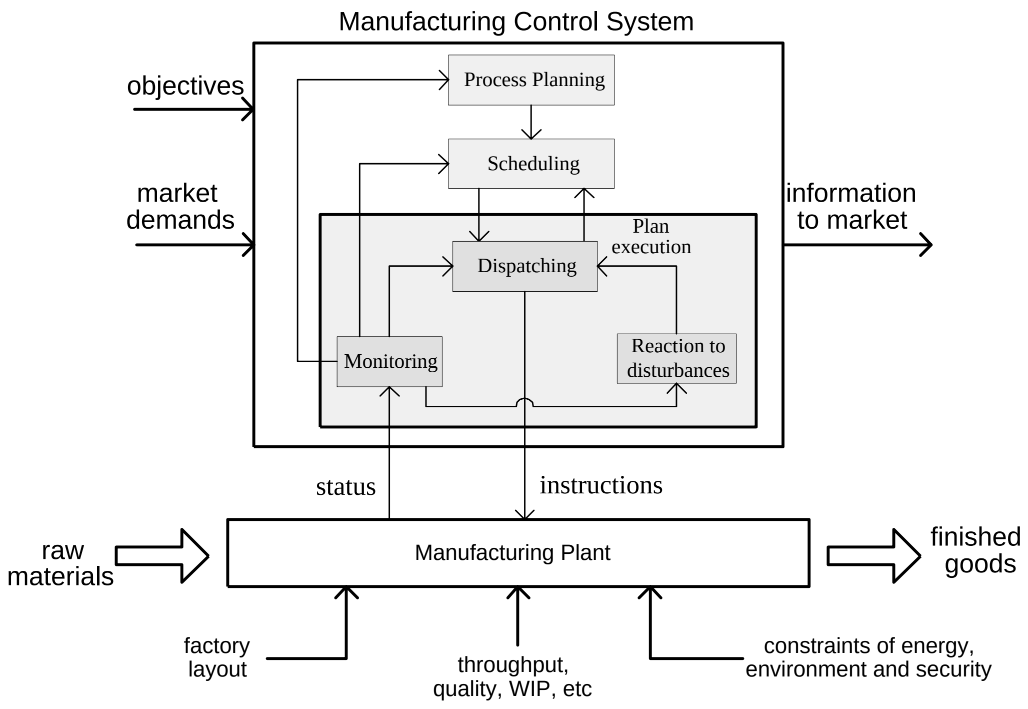


Figure 2.12: Manufacturing Control Model

The pathological state handling intends to keep the system in a safe state, in order to avoid and/or recover from undesirable system states, such as deadlock.

The resource allocation related functions (essentially the scheduling and plan execution) are dependent on the logistical characteristics of the manufacturing system (job shop, flow line, flexible production system, AS/RS, etc.) and on the logistical manufacturing goals (throughput, delivery date, flow time or work in progress).

2.3.2 Short-term Process Planning

The short-term process planning aims to optimise the production process, planning the sequence of operations process and defining the resource allocation based in the available resources. According to the process type it is possible to have fixed routing or alternative process plans, which allows dynamic routing.

The execution of the part P_i ($\mathcal{P}_i \in \mathcal{P}$, which is the set of products) comprises a set of one or more operations

$$\Theta_i = \{o_{i1}, o_{i2}, o_{i3}, ..., o_{in}\}$$

that are partially ordered according to their precedences, represented by

$$\Omega_i = \{< o_{ik}, o_{ij} > | o_{ik} \prec o_{ij}\}$$

i.e. o_{ik} must precede o_{ij}. Each operation o_{ik} requires d_{ik} time units, is no-preemptive and has a set of specifications, $\mathcal{B}_{ik} = \{B_{ikz} | z \in I\}$, that a machine should satisfy to be able to execute the operation.

Being $\mathcal{R}$ the set of available resources at the manufacturing system, a resource R_j ($R_j \in \mathcal{R}$) has abilities to execute the operation o_{ik} if

$$B_{ik} \subseteq S_j \Leftrightarrow \forall x \in B_{ik} \Rightarrow x \in S_j$$

where $\mathcal{S}_j = \{S_{jz} | z \in I\}$ is the set of skills of a resource R_j. The previous expression implies that a resource only has abilities to execute an operation if fulfils all the requirements presented by the operation. The set of resources $\mathcal{R}_{ik}$, which is a sub-set of $\mathcal{R}$ ($\mathcal{R}_{ik} \subseteq \mathcal{R}$), is build considering all the resources that have ability to execute the operation o_{ik}.

A graphical and mathematical representation of process planning information can be done using graph theory as illustrated in [Cho and Wysk, 1995, Marapoulos, 1995] or other tool, such as the Non-linear Process Plan [Kruth et al., 1996]. As an example, [Cho and Wysk, 1995] presents a AND/OR based graph to represent the set of operations and their precedence relationship, as exemplified in Figure 2.13. A node in the graph is one of the five different types: operation, split-or, split-and, joint-or and joint-and.

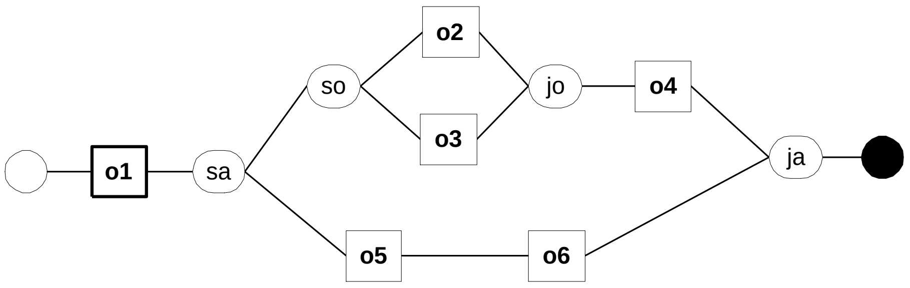


Figure 2.13: A Process Plan Representation Example

All paths following a *split-and* type node must be processed in any sequence, being necessary the execution of both paths. A *joint-and* type node is required to bring multiple paths back together after a *split-and* type node. Only one path following a *split-or* type node must be selected to execution, representing operations alternatives. A *joint-or* type node is required to bring multiple paths together after a *split-or* type node.

2.3.3 Scheduling

Scheduling can be defined as the optimal allocation of resources over the time to jobs, where these assignments must obey to a set of constraints that reflect the temporal relationships between jobs and the capacity limitations of the resources.

Scheduling Problem

The scheduling problem can be formulated as a general optimisation problem that is subject to a set of constraints.

Consider a set of parts P_i, each one containing a process plan giving the set of operations (Θ_i) to be executed in the shop floor, the sequence of operations, the processing time of each operation required to be executed by an available resource ($d_{ik} : o_{ik} \times \mathcal{R} \rightarrow \mathbb{R}^+$), and a due date ($D_i : \mathcal{P} \rightarrow \mathbb{R}^+$). The objective is to produce a schedule plan that allocates operations to the available shop floor resources, minimising (or maximising) a performance measure. Examples of objective functions are the maximisation of the throughput, the minimisation of the work in progress inventory, the minimisation of the manufacturing lead time and the minimisation of the average order tardiness.

Consider also the function δ_{ikjt} ($X \rightarrow \{0, 1\}$) that represents the allocation of the operation o_{ik} to the resource R_j at time unit t (1 if the resource is allocated, 0 if not). The manufacturing scheduling is subject to a set of constraints:

- the operations are non-preemptive,
- each resource can only process one operation at time ($\sum_{k=1}^{m} \delta_{ikj} = 1 (j = 1, 2, ..., n)$),
- a part can be processed by only one resource at time ($\sum_{j=1}^{n} \delta_{ikj} = 1 (k = 1, 2, ..., m)$).

Small scheduling problems can be solved using some simple algorithms to obtain optimal scheduling solutions. However, the manufacturing scheduling becames a complex combinatorial problem, more specifically a non-polynomial (NP) problems, for larger scheduling problems. For a generic n jobs and m machines problem, the number of scheduling solutions is given by $(!n)^m$. As an example, in the case of existence of one machine to process four jobs, there are 4! (24) possible sequences, but if the number of jobs to process increases to 10, the number of possible sequences will be 10! (3628800), being harder to find the best sequence. The problem becomes even harder if instead of one machine, the problem comprises three different machines, where there will be $10!^3$ ($4{,}78 * 10^{19}$) possible sequences. Considering a computer equipped with a 3 GHz processor and that the anaysis of each solution requires only one processing cycle, it is necessary 505 years (!) to analyse all the possible solutions. More complex is the case of 12 jobs and 10 machines, where the size of the search space is approximately $6,36 \times 10^{86}$, which is larger than the number of particules in the Universe[7].

[7]According to Arthur Eddington in his book Mathematical Theory of Relativity (1923, London, Cambridge

Scheduling Strategies

The scheduling problem has been widely studied and referred in literature, mainly due to its highly combinatorial aspects, its dynamic nature and its applicability in manufacturing systems [Shen and Norrie, 1998]. Some referenced methods are heuristics, constraint satisfaction techniques, neighbourhood search techniques and genetic algorithms.

For simple and small scheduling problems, mathematical programming, such as linear programming, is applied to find the optimal solution. Those algorithms may require a huge amount of time to achieve the optimal solution, and are not adequate to problems with large dimension and complexity.

Since the experience shows that it is not important to have the best solution but a satisfactory fast solution, there are scheduling heuristics, such as EDD (Earliest Due Date) and SPT (Shortest Processing Time), which allow a good approach to the ideal solution, in a shorter period of time. However, heuristics are not adequate to the systematically improvement of the generated schedules.

The Constraint Satisfaction approach formulates the problem with constraints using an appropriate constraint programming language [Russel and Norvig, 1995]. This approach, quite analogous to mathematical programming, reduces the search by focusing on specific constraints.

Another approach is based on Lagrangian relaxation, as described by [Bongaerts, 1998, Gou et al., 1998, Luh et al., 2000]. Based on mathematical optimisation techniques, the capacity constraints of a scheduling problem can be relaxed and replaced by a penalty cost. The relaxation of the capacity constraints is quite similar to the mechanism that runs the free-market economy. This approach yields high performance, but uses considerable calculation time.

Other researchers use neighbourhood search techniques, which consist in finding, iteratively, a new solution in the neighbourhood of an existing solution. These techniques use for example simulation annealing [Kirkpatrick et al., 1983] or taboo search [Glover and Laguna, 1997] to perform the stochastic search.

The previous scheduling strategies considers the manufacturing scheduling as static and deterministic. However, the industrial manufacturing scheduling is subject to a dynamic environment, with new jobs arriving continuously to the system, certain resources becoming unavailable and additional resources introduced.

The reactive scheduling deals with the dynamic and stochastic nature of the problem, and

University press), the number of particles in the universe is $\approx 3.1495 \times 10^{79}$. This value is the largest specific integer ever thought to have a unique and tangible relationship to the physical world (all larger numbers in physics are estimates and approximations).

considers disturbances in the environment. The methods used in reactive scheduling do not foresee the stochastic effects, but react to the occurrence of disturbances by restarting the scheduling process from scratch. The proactive scheduling methods consider the risk for disturbances in its schedule, instead of reacting to them.

The distributed scheduling means that the scheduling algorithm is distributed over a number of entities that combine their calculation power and their local knowledge to optimise the global performance [Bongaerts, 1998]. The major advantages of the distributed scheduling are the improvement of reaction to disturbances and the parallel computation. Some algorithms are based on existing centralised algorithms, like dispatching rules or neighbourhood techniques, and others based on emergent behaviour, like market-based and behaviour based algorithms. Some other algorithms are based on contract net protocol [Smith, 1980].

Agent technology approaches have recently been used to solve the manufacturing scheduling problem, mainly using algorithms based on previous described centralised techniques, or other distributed behaviour approaches, such as market-based or contract net protocol. Examples of these types of manufacturing scheduling are described in [Bongaerts, 1998, Parunak et al., 1998, Maturana et al., 1996].

2.3.4 Plan Execution

Since the schedule is achieved, it is necessary to execute the achieved plan in the factory plant. The plan execution process comprises mainly the dispatching, monitoring and reaction to disturbances functions.

Dispatching

The schedule implementation takes into account the current status of the production system [Bauer et al., 1991]. A dispatching algorithm decides how to use a manufacturing resource only upon the availability of the factory plant resources. The dispatching rule determines which job a resource will work on next. This sequencing decision can be made based on the job's due-date, the customer priority, minimisation of set-ups, the shortest processing time, or any other possible rules or heuristics.

The dispatching based in priority rules is known by scheduling heuristics and are commonly used in the scheduling problem. There are a lot of priority rules, such as the already referred EDD and SPT, and also the FIFO (First In First Out), SST (Shortest Set-up Time) and LOR (Least Operations Remaining). A more elaborated rule combines some of those priority rules into only one method, each one having a specific weight.

The advantages of dispatching rules come from its simplicity, because they only require local information and do not require significative computation power.

Monitoring

An important aspect of the factory operation is to have a detailed and up-to-minute knowledge about the work in progress and the status of the process. The monitoring can be done in two different ways: passive and active.

In the passive monitoring, the initiative belongs to the entity that wants to know some specific information, such as the current status of an order or the capacity of a production site. The active monitoring involves the subscription of events for the future notification in case of their occurrence.

Reaction to Disturbances

The occurrence of unexpected disturbances can lead to deviations from the initial plan, reducing the system productivity due to the machine/system inactivity. In this case, the system should respond dynamically and quickly to the disturbance, using appropriate mechanisms according to the type of disturbance.

The dynamic reaction to disturbance plays an important role in the manufacturing control systems, since nowadays their performance is also evaluated in terms of agility and flexibility.

2.3.5 Deadlock Handling

The manufacturing systems exhibit a high degree of resource sharing, with the parts competing for the access to them. Deadlock is a highly unfavourable situation, and occurs when the parts in a set request access to machines held by other parts in the same set [Fanti et al., 1997].

The necessary and sufficient conditions, which should be fulfilled simultaneously, to provoke a deadlock, are [Silberschatz and Peterson, 1991]:

- Mutual exclusion: at least one resource in the system cannot be shared.
- No pre-emption: processes cannot force other processes to give up their resources.
- Hold and wait: a process holds a resource while waiting to acquire other resources.
- Circular wait: a closed chain of processes, in which each process is waiting for a resource held for a resource held by the next process in the chain.

A valid deadlock mechanism should guarantee that one of the above conditions does not hold. The deadlock mechanisms can be classified as: deadlock prevention, deadlock avoidance, and deadlock detection and recovery.

The deadlock prevention methods intend to design a deadlock-free resource allocation system, ensuring that the entire set of necessary conditions for deadlock cannot be satisfied simultaneously [Reveliotis and Ferreira, 1996]. Generally, the application of these methods leads to low resource utilisation.

The first and second conditions cannot be avoided in the manufacturing context, because it is normal to have non-sharable resources and the interruption of an operation can produce scrap.

The third condition can be avoided if all needed resources are requested and allocated before order begins processing. These mechanisms, called *all or nothing allocation*, lead to low resource utilisation, and orders requesting several scarce resources will never be able to start processing.

The fourth condition can be avoided using a numbering sequence system in which each resource has a unique sequence number and the orders shall request and allocate resources in sequence of increasing sequence number. The assignment of the sequence numbers to the manufacturing resources will safeguard the utilisation rate of the resources and is probably the most well-known deadlock prevention mechanism.

The deadlock avoidance methods prevent the system to enter in a deadlock state, making use of information about the resources currently available, the resources currently allocated to each order, and the future requests and releases of each order, to decide if the current request can be satisfied or must wait to avoid a possible future deadlock. The well-known example of this type of deadlock mechanism is the Banker's algorithm [Haberman, 1969].

The Petri Nets models has been used to develop some deadlock avoidance techniques, such as described by [Viswanadham et al., 1990], and [Banaszak and Krogh, 1990]. The usage of Petri nets has advantages in terms that the deadlock states can be explicitly modelled in the Petri net model and the process routing flexibility can be easily expressed in the model. However, they are difficult to be applied to real-time control since the model complexity increases exponentially as the number of resources becomes larger [Yoon and Lee, 2000].

Deadlock detection and recovery mechanisms allow the system to evolve into a deadlock state, being then detected by a monitoring mechanism that initiates a recovery procedure. This recovery procedure rolls the system back to a safe state, by pre-empting resources from deadlocked processes. Normally, these methods are not applicable in manufacturing systems, mainly due to the impossibility to have pre-emption of the operations.

2.3.6 Requirements of Manufacturing Control Systems

The manufacturing systems are typically **heterogeneous environments**, comprising heterogeneous hardware and software applications, being the functions, knowledge, skills and operations distributed. In these environments, the manufacturing systems should be able to act cooperatively to achieve common goals. The manufacturing control systems should be based in a set of **autonomous components** in order to make easier its organisation into cooperative, dynamic and distributed structures, able to share skills and knowledge to achieve common goals.

The control systems should be **expandable**, being possible the addition of new components, without the need of re-design, re-programming and re-initialisation of the other components. The control system should be **re-configurable**, adapting dynamically to configuration changes, without stopping or re-starting the process.

The manufacturing systems are **stochastic and dynamic environments** with certain resources becoming unavailable and additional resources introduced, new jobs arriving continuously to the system, and new products introduced in the system, as well new regulations (such as quality and safety specifications). The traditional manufacturing control systems have low **capacity to adapt and react to the dynamic changes** of its environment, such as machine breakdowns and market volatility, mainly due to the rigidity of their control architecture. Next generation of manufacturing control systems should be able to react rapidly to the occurrence of disturbances, in order to minimise their effects, and also to create plans that anticipate the future occurrence of disturbances.

The introduction of **learning mechanisms** plays an important role in the improvement of the control system performance, dealing with the industrial environment unpredictability and evolution.

The control system should be **portable or platform independent**, i.e. capable of being used on different vendor platforms.

The manufacturing control system should provide mechanisms to **integrate automation devices**, such as robots or machine-tools, both to send commands and instructions, and to collect information. Also it should provide interfaces to integrate the existing legacy systems, such as scheduling systems and production planning systems.

The product diversity and market volatility require the **system flexibility**, providing alternative processing workstations, alternative transport paths, alternative execution plans, etc.

The implementation of new control systems is seldom done from scratch, requiring a huge amount of time and effort to develop new applications. The **re-use** of past or previous solutions (knowledge, know-how or components) allows to simplify the solution development.

The data translation is related to the fact that each distributed entity represents and inter-

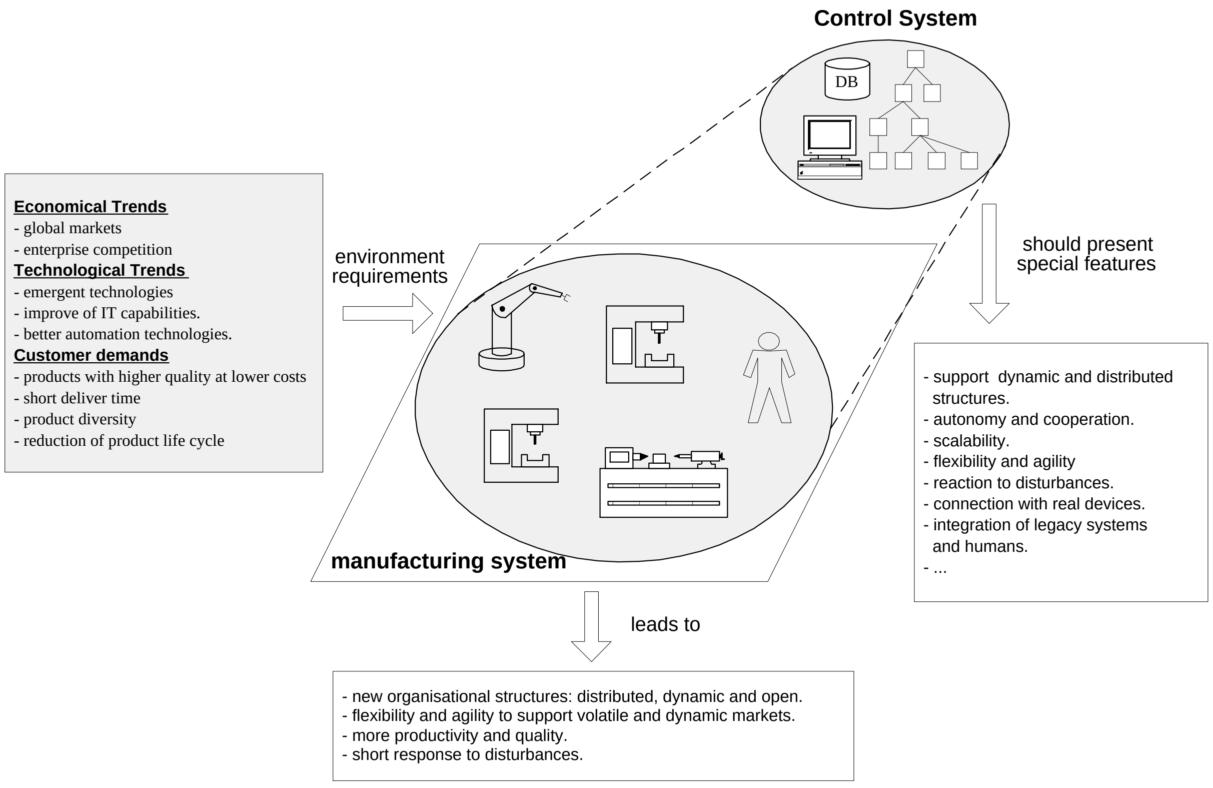

Control System
DB
Economical Trends
- global markets
- enterprise competition
Technological Trends
- emergent technologies
- improve of IT capabilities.
- better automation technologies.
Customer demands
- products with higher quality at lower costs
- short deliver time
- product diversity
- reduction of product life cycle
environment requirements
should present special features
- support dynamic and distributed structures.
- autonomy and cooperation.
- scalability.
- flexibility and agility
- reaction to disturbances.
- connection with real devices.
- integration of legacy systems and humans.
- ...
manufacturing system
leads to
- new organisational structures: distributed, dynamic and open.
- flexibility and agility to support volatile and dynamic markets.
- more productivity and quality.
- short response to disturbances.

Figure 2.14: Requirements and Features of Manufacturing Control Systems

prets the information in a different way. The **exchange of data** in a distributed environment requires the utilisation of standards protocols and ontologies, to support the interaction between distributed entities.

An important question when handling manufacturing control systems, which increases the problem complexity, is how to **integrate the process planning with the scheduling and plan execution** capabilities. This requires an integrated view of all manufacturing control functions, in order to have a simultaneously optimisation of the process planning and the manufacturing scheduling taking in consideration the current availability of the manufacturing resources.

Manufacturing systems also present specific features that differ from other domains, which increase the complexity of the control systems. As an example, the occurrence of failures in a manufacturing device, such as collision and failure in the compressed air, leads frequently to the temporary inactivity of the manufacturing devices, requiring external intervention to solve and/or re-start the device. Additionally, the manufacturing operations are **non-preemptive**, which makes even difficult the re-scheduling process in case of machine failure.

In certain situations, the manufacturing control systems present **real time constraints**, mainly at the machine controller level. A real time control system is a system that should produce a response within a well-defined limit of time. The response after this temporal limit causes degradation or bad functioning of the system.

2.4 Traditional Manufacturing Control Approaches

The control structure assumes a crucial importance in the final performance of the manufacturing control system. According to most of authors (such as [Diltis et al., 1991]), the control architectures can be classified as centralised, hierarchical and heterarchical. In this section will be analysed the traditional manufacturing control approaches, mainly those based in the centralised and hierarchical control structures.

2.4.1 Centralised and Hierarchical Control Structures

The centralised architecture was the first to be used and is characterised by a single decision node, where all the planning and processing information functions are concentrated [Diltis et al., 1991]. This architecture has the advantage of a better management and control optimisation, although presenting some important disadvantages, in terms of speed of response, control complexity, tolerance to faults and expandability, specially for large systems.

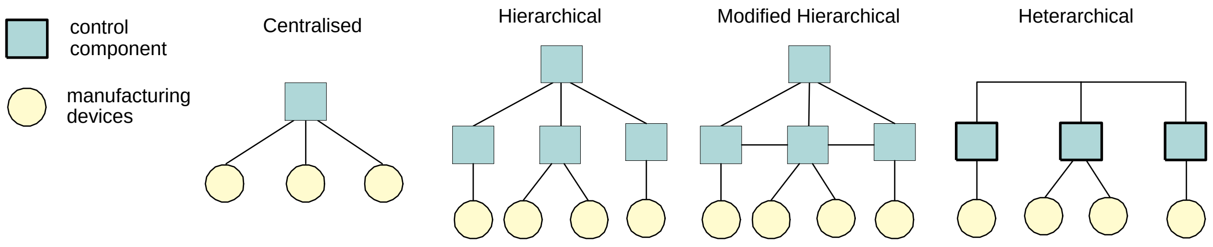


Figure 2.15: Control Structures (Adapted from [Diltis et al., 1991])

In the hierarchical architecture, a complex problem is decomposed in several simpler and smaller problems, and distributed among multiple control layers. This architecture is characterised by the existence of some control levels, distributed in a tree structure, allowing the distribution of decision-making among these hierarchical levels. The relations between hierarchical levels are based on the master-slave concept. The main advantages of this architecture are the robustness, the predictability and the efficiency that are better than in centralised architectures. However, the appearance of disturbances in the system reduces significantly its performance.

The modified or proper hierarchical architecture tries to improve the response to disturbances, maintaining all the features of hierarchical architecture and adding the interaction between modules at the same hierarchical level. The expandability of the system is easier than the hierarchical architecture due to the interaction at same control level feature.

2.4.2 Research Work using Traditional Control Structures

Some research in the area of cell and shop floor control was carried out using the traditional control architectures, namely the centralised or hierarchical control architectures.

The ESPRIT project 447, COSIMA (Control Systems for Integrated Manufacturing), has developed functional software architecture for cell and shop floor levels [Bauer et al., 1991]. It consists of five well-defined functional modules, grouped into the Production Activity Control (PAC) concept, which controls one manufacturing cell. The PAC architecture modules are the scheduler, the dispatcher, the monitor, the producers and the movers, as illustrated in Figure 2.16.

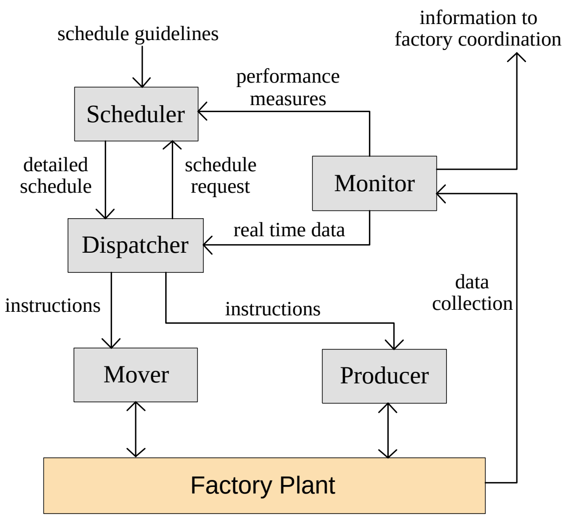


Figure 2.16: PAC Architecture [Bauer et al., 1991]

The scheduler handles the plans of manufacturing resources according to the long term tactical plans and resources capacities. The dispatcher is the heart of control system and acts in real time control over the manufacturing environment, and the monitor collects shop floor data to give a logical view of actual states in the manufacturing environment. The producers and movers modules control the shop floor resources. The co-ordination between PAC systems is performed by the FC (Factory Co-ordination) module. The FC module consists of a control task and a production environment design task. The control task is similar to the PAC, in which the movers and producers are the PAC modules. The production environment design task contains process planning, layout maintenance and analysis of the manufacturing systems.

The CHAMP (Chalmers Architecture and Methodology for Flexible Production) architecture, developed at Chalmers University of Technology, is a modified hierarchical reference architecture, to model the control software of manufacturing cells. It is elaborated upon the experiences made from implementations of control systems based on the PAC architecture and its extensions, PAC+ and PAC++ [Andersson, 1997]. CHAMP architecture includes functions for scheduling, dispatching, resource control, monitoring and error handling [Gullander, 1999]. The main features of this architecture are the separation of product and resource information, and the physical separation of generic functions from specific functions of the products and the resources currently in use. The architecture introduces also a generic message-passing structure.

FACE (Flexible Assembly Control Environment) is an architecture, developed at the Royal Institute of Technology, that aims to simplify and speed up programming and control of flexible automatic assembly cells [Onori, 1996]. The FACE architecture consists in the following modules: off-line control, on-line control, error recovery and databases. The off-line control module manages the generation of motion programs for robots included in the assembly cell. The on-line control module is responsible for the management and control of the system and comprises the setup, order planning and dispatcher modules. The error recovery module is used when a failure in the assembly cell is detected, applying a combination of active recovery and re-scheduling techniques.

Manufacturing Decision Making (MADEMA) was introduced by the Massachusetts Institute of Technology aiming to develop decision-making models at the cell level [Chryssolouris, 1987]. It comprises the simulator, decision-making and database modules. The simulator module simulates the production under different kinds of scenarios and cell organisations. The decision-making module determines which alternatives are available, analysing the consequences and selecting the best alternative. This approach has the possibility to test, in an off-line mode, the response of a manufacturing system due to a certain decision. However, this can be viewed as a weak point because the decision-making mechanism can only be used in off-line mode.

RapidCIM was a joint venture project between Texas A&M University, Penn State University and Systems Modelling Corporation, aiming to facilitate the process of developing full-automated computer controllers for flexible manufacturing systems, generating quickly the application code. The basic components of the RapidCIM concept are the shop floor architecture (based in the hierarchical control structure), factory and process plan model, formal models of execution and associated tools (for development of execution software) and simulation for real time control [RapidCIM, 2003].

CIM Centre of Porto manufacturing cell controller architecture was developed and implemented for its flexible manufacturing cell. It uses a modified hierarchical architecture approach,

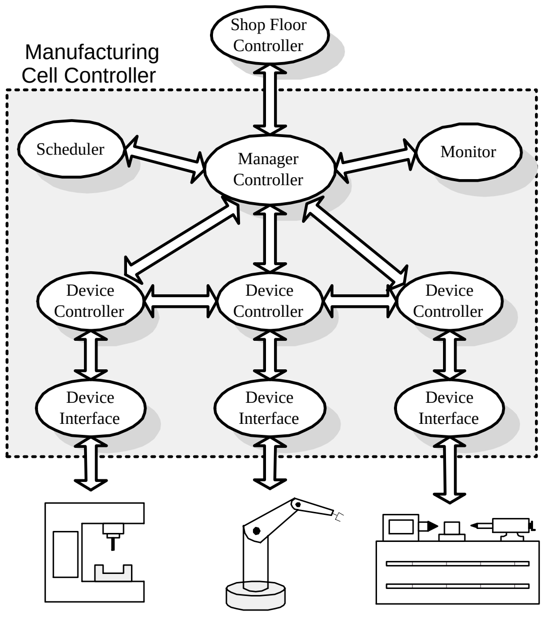


Figure 2.17: Manufacturing Cell Controller Architecture at CIM Centre of Porto

Figure 2.17, comprising a set of modules, whose brain is the manager module, which is responsible for the control and the supervision of the production process of the manufacturing cell [Quintas and Leitão, 1997, Leitão, 1996]. Each real device has a customised module, designated by device controller, that has the responsibility for the local control of the machine, and for the execution of the jobs requested by the high level module. The interface between the cell controller and each of the industrial machines is implemented using the MMS (Manufacturing Message Specification) communication protocol [ISO/IEC9506-1, 1992].

The MOSCOT (Modular Shop Control Toolkit for Flexible Manufacturing) architecture, developed upon the ideas and concepts resulted from ESPRIT Project 20920 - PASO (Paradigm Independent Shop Control for Smaller Manufacturing Sites), is characterised by two main parts: a kernel and a set of shells [Teunis et al., 1998]. A kernel contains the common modules (objects) to all shop floor applications; the shells are developed or customised according to the specifications of each application. Other innovation associated to this architecture is the SCAPI (Shop Control Application Program Interface), which acts like an operating system for shop control applications developers.

FACT (Factory Activity Control Model) architecture, developed at University of Twente, introduces a generic concept for shop floor control. The architecture contains six basic modules, distributed over two hierarchical levels [Arentsen, 1995]: scheduling, dispatching, monitoring and diagnosis modules belongs to the cell level, and the the workstation control and auxiliary station control belongs to the station level. These modules describe the planning, control and monitoring functions required for an efficient production plan execution.

2.5 Agent-based Manufacturing Control

The traditional manufacturing control systems don't support efficiently the current requirements imposed to the manufacturing systems. With the increase of powerful, inexpensive and widely available computational resources, the architectures evolved from centralised to distributed and dynamic approaches, requiring the need for some degree of autonomy to enable components to respond dynamically to changes.

The heterarchical architecture, also designated by autonomous agent approach in the agent domain, is characterised by the high level of autonomy and co-operation, being the client-server structure with fixed relations no more applied [Diltis et al., 1991]. These features allow a high performance against disturbances, being the global optimisation reduced, because the decision-making is local and autonomous, without a global view of the system. The expandability of the system is an easier task, because it is enough to modify only the functioning of some modules or add new modules to the control system.

The agent technology introduces functionalities that support efficiently the distributed manufacturing system needs, such as modularity, decentralisation, and dynamic and complex structures characteristics, for what agents are well suited to solve [Parunak, 1998]. Agent-based approaches have been applied in many different areas, such as electronic commerce, e-business, air traffic control, process control and telecommunications, besides manufacturing.

2.5.1 What is an Agent?

The multi-agent system paradigm derives from the Distributed Artificial Intelligence (DAI) field, being characterised by decentralisation and parallel execution of activities based on autonomous entities, called agents.

The definition of agent concept is neither unique nor consensual [Russel and Norvig, 1995, Wooldridge and Jennings, 1995, Ferber, 1999, Luck et al., 2003]. Despite the some definitions and interpretations for agents, in this work an agent will be defined as being:

> *An autonomous component, that represents physical or logical objects in the system, capable to act in order to achieve its goals, and being able to interact with other agents, when it doesn't possess knowledge and skills to reach alone its objectives.*

In the automation and manufacturing domain, an agent can represent physical resources, such as machine tools, robots, auto-guided vehicles and products, and logical objects, such as the schedulers and orders.

A frequently question related to the application of agents is if agents are a new approach or just a new way to look old concepts. Agents allow a new approach to the problems, both in the design and implementation phases, due to their characteristics of autonomy, decentralisation and decision capabilities. Additionally, the required software to develop agents is shorter and simpler than the software required by the centralised approach, leading to easier development, debug and maintenance [Parunak, 1996].

The agent-based and object-oriented approaches present some similarities that allow some confusion, mainly because agents can be implemented using object-oriented programming languages, such as Java and C++. The first difference between the two approaches is that objects are passive in their behaviour, acting only in reaction to an external stimulus, such as the reception of a message or the invocation of a method. The second difference, and perhaps the more significant difference, is that the objects encapsulate a set of attributes and a set of services (methods), not possessing the capacity of choice about the execution of the services requested by other object. On the other hand, agents have ability to decide if they accept or reject the service execution, based in their knowledge and skills. This difference can be summarised in the following slogan: objects do it for free; agents do it for money. At last, in the object-oriented approach, the messages are necessarily related with the invocation of object methods, while in the agent-based approach, the agents can receive messages that are not requests for services execution, but they can consist in information requests or in information sending.

2.5.2 Agent Typologies

A typology refers to the study of types of entities. There are several approaches to the classification of the agents, such as presented by [Wooldridge and Jennings, 1995, Nwana, 1996, Ferber, 1999].

[Wooldridge and Jennings, 1995] classify the agents according to the following types: deliberative, reactive and hybrid. Deliberative agents are characterised by their goal-oriented behaviour, knowledge representation, reasoning model and the planning process that aims the generation of correct and optimal sequences of actions, having the capability of anticipation. The reactive agents do not have internal knowledge representation and operate in a stimulus-response manner aiming to produce robust actions in contrast with deliberative agents that aim to produce optimal actions. This reactive behaviour is described as situation-action rules. The hybrid agents combine the best features of deliberative and reactive agents, achieving fast response and generation of optimal sequences of actions.

[Ferber, 1999] presents a quite similar classification, by considerating reactive and cognitive types of agents, the cognitive agents being similar to the deliberative agents presented by

[Wooldridge and Jennings, 1995] classification.

[Nwana, 1996] extends the previous classification by introducing several dimensions to classify the agents: mobility, deliberative or reactive and ideal attributes. In the mobility dimension, the agents are classified in relation to their ability to move around some network, being possible to have static and mobile agents. The mobile agents, are capable to move around a network, interact with foreign hosts, gather information on behalf of its owner and come back home having performed the duties set by its user. The second dimension is related to the capacity of response of the agents and is similar to the [Wooldridge and Jennings, 1995] classification. The third dimension is related to the characteristics that an agent should exhibit. Examples of these attributes are the autonomy, co-operation, learning and versatility. The combination of these attributes leads to several types of agents, such as collaborative agents, interface agents or smart agents.

Other researchers classify agents under other parameters, such as what they do, what they are and what technologies they use. Additionally, in other classifications, the social aspects of the agents are also considered.

A well known cognitive agent type is the BDI (Belief-Desire-Intention) architecture, which origin lies in a theory of human practical reasoning, focusing particulary the role of intentions in practical reasoning [Wooldridge, 2002]. In the BDI agents, the decision making depends from the manipulation of beliefs, desires and intentions of the agents.

The development of reactive agents is simpler than the cognitive agents [Ferber, 1999], easy to understand and more robust and fault-tolerant than the other agent types [Nwana, 1996]. However, reactive agents are incapable of foreseing what is going to happen and thus of anticipating the future by planning what action to take [Ferber, 1999].

2.5.3 Agent Technologies

A multi-agent system can be defined as a set of agents that represent the objects of a system, capable to interact, in order to achieve their individual goals. In multi-agent systems, since each agent has a partial view of the system, the agents need to be able to communicate, in order to achieve a pre-defined goal or solve a problem. The interaction between agents requires that the agents can understand themselves, using a proper agent communication language, ontologies and interaction protocols.

In volatile and dynamic scenarios, where it is difficult to foresee future events, agents must learn to adapt their behaviour to those dynamic environments, improving their performance. Learning contributes for the intelligent of an agent, by acquiring new knowledge and skills, which will be used in the future to take better decisions.

Agent Communication Languages

Agent Communication Languages (ACL) intend to make transparent the data exchange between distributed holons, being crucial to standardise the messages used during the communication act. The two major current agent communication languages are KQML (Knowledge Query and Manipulation Language) [Finin et al., 1993] and FIPA-ACL (Foundation for Intelligent Physical Agents - Agent Communication Language)[8] [Labrou et al., 1999], which are both based on the speech act theory [Searle, 1969]. Speech acts designate all intentional actions carried out in the course of a communication, being the elementary units that make possible to establish a conversation between agents.

KQML and FIPA-ACL are almost identical with respect to their basic concepts and principles they observe.

The KQML language is probably the best-known and mature agent communication language, which is both a message format and a message-handling protocol to support run-time knowledge exchange among agents, but is indifferent to the format of the information itself [Finin et al., 1993, Ferber, 1999].

The FIPA-ACL describes every communicative act with both a narrative form and formal semantics based on modal logic, and it also includes a normative description of a set of high-level interaction protocols, such as requesting information and contract-net [Labrou et al., 1999].

FIPA-ACL and KQML uses different names to represent the type of communication act. They use also, different names for some reserved primitives. The FIPA architecture defines an AMS (Agent Management System) that specifies services that manage agent communities, eliminating the need for register/unregister, recommend, recruit, broker and (un)advertise primitives used in KQML.

Ontologies

In multi-agent systems, the communication between cooperative agents requires a common understanding of the concepts of their knowledge domain.

The term ontology is vague and not precise. [Gruber, 1995] defines ontology as a specification of a conceptualisation. [Guarino, 1998] extends the previous definition, saying that an ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualisation of the world. Besides these differences, it is

[8]FIPA is an organisation formed in 1996 to produce software standards for heterogeneous and interacting agents and agent-based systems. FIPA-ACL is the agent communication language specified by FIPA organisation.

consensual that the purpose of ontologies is to create shared understanding between co-operative agents, enabling the exchange of knowledge and the capability to reuse that knowledge.

In the context of this document, an ontology defines the vocabulary that will be used in the communication between agents, and the knowledge relating to these terms. This knowledge includes the definition of the concepts, their attributes, relationships between the concepts and constraints. The use of standard ontologies is crucial in the development of holonic and agent-based manufacturing control applications, because is a step toward the integration of heterogeneous agents.

KIF (Knowledge Interchange Format) is a language that allows a user to develop ontologies. The KIF language, based on first-order logic, allows two agents with different knowledge base, to inter-operate, through the translation of each knowledge base into KIF format, that will be shared. Typically, when an agent receives a knowledge base in KIF, it converts the data into its own internal form. When the agent needs to communicate with another agent, it maps its internal data structures into KIF. The KIF is not efficient as a specialised representation for knowledge but it is a programmer-readable language and thereby facilitates the independent development of knowledge-manipulation programs [KIF, 2003].

The best-known KIF ontology is Ontolingua [Farquhar et al., 1996]. Ontolingua intends to provide a common platform in which ontologies developed by different groups can be shared [Wooldridge, 2002]. Ontolingua consists on a library of ontologies, expressed in the Ontolingua ontology definition language, which is based on KIF, on a set of tools for analysing and editing the ontologies, and a set of translators for converting Ontolingua sources into forms acceptable to implemented knowledge representation systems.

Other languages have been developed to define ontologies, such as the DAML+OIL (DARPA Agent Markup Language + Ontology Interchange Language) [Horrocks et al., 2001], DOLCE [Gangemi et al., 2002] and Loom [Loom, 2004].

Learning Mechanisms

Learning can be defined as a way to acquire knowledge and skills to respond to the dynamic evolution of the environment and to improve the system ability to act in the future. The idea beyond learning is that perceptions received should be used not only for acting, but also for improving the ability to behave optimally in the future to achieve goals.

Learning is normally adopted when it brings benefits to the manufacturing control context, in result of a decision-making process or by the observation of the environment, allowing to adjust the decision parameters or even to update the behaviour rules.

According to [Russel and Norvig, 1995] the learning techniques can range from trivial memorisation of experience to the creation of entire scientific theories. Additionally, the learning techniques are dependent on the language in which the learned knowledge is represented, such as relational descriptions, if-then rules and decision trees. Several classifications for the learning techniques are presented in the literature, such as discussed by [Russel and Norvig, 1995, Kubat et al., 1996, Sen and Weiss, 1999, Monostori et al., 1996, Goldman and Rosenschein, 1996]. A possible approach is to classify the learning techniques according to five main categories: rote learning, supervised learning, reinforcement learning, unsupervised learning and hybrid learning.

Rote learning involves the simple memorisation of incoming information for later use. This type of learning does not need to do any processing to understand or interpret the information supplied by the environment, the learned information being recalled when the same problem instance arises. The other categories of learning involve the understanding and interpretation of returned information, to create new knowledge.

Supervised learning is related to techniques where the learner is provided with feedback information (a set of examples and information) describing the desired activity; the objective is to match this desired action as closely as possible. As example of this type of learning is the inductive learning, also called learning from examples, where the goal is to construct a formula that matches all the given positive examples and no negative example.

The reinforcement learning algorithms are quite similar to supervised learning, using reward feedbacks, known as the reinforcement signals, to learn a successful function that allow to automatically determine the ideal behaviour within a specific context, in order to maximise its performance. The system receives occasional positive or negative rewards, indicating how successful was the decision, rather than being told the correct action.

The unsupervised learning uses techniques to learn new knowledge without labelled classes, optimisation criterium, feedback signal, or any other information rather than raw data. In contrast to the supervised learning, the learner is not presented with an explicit set of examples showing the desired input/output relations. Examples of this type of learning are clustering and discovery techniques. Clustering techniques attempt to classify objects looking for classes based on feature description. The learning by observation or discovery acquires knowledge without a teacher, by discovering new concepts merely from unstructured observation or by planning and performing experiments in the environment.

The hybrid learning includes the learning algorithms that do not immediately fall into the either unsupervised or supervised categories. Examples of this type of learning are the learning by analogy, case-based reasoning and explanation-based knowledge. Learning by analogy is based on applying existing knowledge to a new situation based on similarities between them. The case-based reasoning approach solves a new problem by modifying old solutions until they meet with the requirements of a new problem. The major problem with the inductive learning is the large number of training examples, both positive and negative, required to teach a concept of low complexity. The explanation-based learning technique uses an explicitly defined knowledge domain to extract general rules from single examples by explaining and generalising the explanation.

2.5.4 Research Work in Agent-based Manufacturing

The application of agent technology in the manufacturing field has being carried out by several research teams, in different application domains, such as enterprise integration and manufacturing planning and control [Shen and Norrie, 1999].

In the area of enterprise integration, the agent-based technology provides a natural way to design and implement open and distributed environments. CIIMPLEX [Peng et al., 1998] uses agents to represent the communication services, such as name server, facilitator and gateway, aiming the integration of planning and execution manufacturing systems, in real scenarios, involving legacy systems (execution system and scheduler system).

The HOLOS architecture, [Rabelo and Camarinha-Matos, 1994], supports multi-agent based dynamic scheduling. The flexibility of the architecture both from the organisational point of view and from the control one is supported by negotiation techniques and the dynamic formation of consortia of manufacturing resources. One of the important aspect of this work is the tandem agent architecture to support integration with legacy systems. This work was later extended to distributed multi-site manufacturing systems and virtual enterprises in the framework of the MASSYVE project [Rabelo et al., 1999].

MetaMorph I [Maturana and Norrie, 1996] uses an agent federation centred in the mediator approach, supporting the change of form, structure and activity, in order to adapt dynamically to emerging tasks and environment change. Agents represent manufacturing devices and products, and the mediators are used to coordinate the interactions between agents. The approach supports dynamic clustering and cloning, and learning. MetaMorph II [Shen et al., 1998] uses a hybrid architecture combining the mediator and autonomous agents approaches for the integration of enterprise activities, such as planning, scheduling and execution, with the activities of their customers, suppliers and partners. In this approach, the manufacturing system is, in a first step, organised at the highest level through sub-system mediators. Each sub-system itself can be an agent-based system, in which the agents may be autonomous agents. Some of these agents may also be able to communicate directly with other sub-systems or with agents in other sub-systems.

DEDEMAS (Decentralised Decision-Making and Scheduling), [Tönshoff et al., 2000], supports the integration of distributed systems providing mechanisms for decentralised decision making and scheduling covering both multi-site operations of one company and its chain of external suppliers. DEDEMAS is designed to be an intelligent and decentralised mediation system for order scheduling and monitoring in multi-site, supply chain and virtual enterprise cases.

In the domain of manufacturing planning and control, the agent technology introduced the possibility to develop modular and flexible applications to support the manufacturing systems complexity, flexibility and decentralisation. [Duffie and Piper, 1986] had been of the first ones to discuss and to introduce the heterarchical control approach, using agents to represent physical resources, parts and human operators, and implementing scheduling oriented to the parts.

CORTES [Sadeh and Fox, 1989] uses micro-opportunistic techniques to solve the scheduling problems and Constrained Heuristic Search techniques for the decision-making related to the scheduling. The agents execute scheduling and monitoring for a set of resources.

IFCF [Lin and Solberg, 1992] uses a market-based control model to implement resource allocation and distributed scheduling. The market-based mechanism uses multiple step negotiation, allowing the real time coordination of agents. The agents represents physical resources, parts, databases and communication processes.

YAMS (Yet Another Manufacturing System) [Parunak, 1998] applies a contract net technique to a hierarchical model of manufacturing system, including agents to represent the shop floor. AARIA (Autonomous Agents at Rock Island Arsenal) intends to control a production system to fulfil incoming tasks in due time, focusing in the dynamic scheduling, dynamic re-configuration and in the control of manufacturing systems that fulfil the deliver dates [Parunak et al., 1998]. The manufacturing resources, process and operations are encapsulated as agents using an autonomous agent approach.

MASCADA developed manufacturing control mechanisms to support the production change and disturbance, safeguarding and/or maximising the production systems throughput, where the disturbances reduce the effectiveness of the plans/schedules that are generated initially [Valckenaers et al., 1999, Peeters et al., 1999]. The approach uses a pro-active disturbance handling mechanism and uses autonomous and intelligent agents to represent the factory components.

[Bussmann and Schild, 2001] uses agent technology to design a flexible and robust production system for large series manufacturing. The approach uses agents to represent machines and workpieces, implementing dynamic resource allocation with the objective of continuous optimisation of the throughput.

PABADIS [Sauter and Massotte, 2001] uses the concept of CMUs (Co-operative Manufacturing Units) to provide significant functions to the production process in automation control, encapsulating residential, products and shop floor management as agents. For the products it was used the mobile agent technology. The approach comprises centralised (for the connection with Enterprise Resource Planning (ERP) systems) and decentralised components. [Maturana et al., 2002] describes an industrial application, which uses a multi-agent architecture to automate a chilled water system. The approach is developed according to the FIPA specifications and uses dynamic decision-making organisations based on agents to plan, commit and execute control tasks.

Schneider Electric GmbH, Industrial Automation, in co-operation with DaimlerChrysler AG, Research and Technology, Berlin, had developed and implemented a heterogeneous agent-oriented collaborative control system, called FactoryBroker^{TM}, adequate to control widely distributed and heterogeneous devices in environments that are prone to disruptions and where hard real-time constraints are crucial [Colombo et al., 2004].

The agent technology was applied in other manufacturing domains besides the enterprise integration and the manufacturing control. As an example, [Barata and Camarinha-Matos, 2003] focus on the shop floor re-engineering, using agents to represent the physical components which are agregated into consortia regulated by contracts, to agilise the shop floor life cycle.

2.6 Holonic Manufacturing Systems

To face the requirements of operating on a global scale and to meet the needs of an ever more demanding consumer market, an international collaborative research program in manufacturing, called Intelligent Manufacturing Systems (IMS), was started in the beginning of nineties. Within the IMS programme, several paradigms for the factory of the future were developed, such as holonic, bionic and fractal manufacturing systems.

These new distributed manufacturing control paradigms have the ability to respond promptly and correctly to external changes, and they differ from conventional approaches due to their inherent capability to adapt to changes without external interventions. These theories present similar concepts and characteristics but with different origins: mathematics for the fractal factory [Warneke, 1993], nature for bionic and genetic manufacturing systems [Okino, 1993] and social organisation for holonic manufacturing systems [Brussel et al., 1998]. In spite of the similarity of concepts and characteristics, these paradigms emphasis a different set of issues and characteristics, reviewed by [Tharumarajah et al., 1996].

These paradigms suggest the idea that manufacturing systems will continue to need a hierarchical structure besides the increased autonomy assigned to individual entities. They also advise that hierarchy is needed in order to guarantee the inter-entities conflict resolution and to maintain the overall system coherence and objectivity resulting from the individual and autonomous attitude of the entities [Sousa et al., 1999]. The concepts of each paradigm are yet quite general and are currently being developed by different research communities.

2.6.1 Paradigms for the Factory of Future

Bionic Manufacturing

The Bionic Manufacturing Systems (BMS) is a paradigm developed in the framework of the IMS programme, within the Next Generation of Manufacturing Systems initiative. BMS, introduced by [Okino, 1993], tries to translate to the manufacturing world the concepts based on structures and behaviours observed in biology.

In the biological systems, the most complex living forms belong to a certain hierarchical structure, and simultaneously they present autonomy, dynamic behaviour and social harmony, based in the self-organisation concept and evolution mechanisms.

The bionic manufacturing is developed under these ideas, and assumes that the manufacturing companies can be built upon open, autonomous, co-operative and adaptative entities, which can evolve. The cell, organ or living being are modelled in BMS by the *modelon* concept, which is composed of lower level *modelons*, forming a hierarchical structure.

The biological cells are autonomous and control their local behaviour based on the surrounding environment and their genetic code: *static* information related to the genetic data found in DNA (Deoxyribo Nucleic Acid)[9], and *learned* information found in brain neurons. By analogy to biological cells, the *modelons* have a set of properties and behaviours represented by a code with static information and adaptive information. The notion of DNA inheritance is translated to manufacturing world by the properties and behaviours that each higher level *modelon* passes to low level *modelons*.

In biology, *enzymes* are responsible for the regulation and control of the system. *Enzymes* are modelled in manufacturing context by supervisors, which also play an organisational and structural role in the co-operation process within the BMS, influencing the relations between

[9]All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serve as a model for the whole organism. A chromosome consists of genes, blocks of DNA. Each gene encodes a particular protein. Basically, it can be said that each gene encodes a trait, for example color of eyes. Thus, DNA, contained in the cells chromosomes, carries the genetic information about the organism characteristics, such as the eye or skin colour. Moreover, each individual organism has its own composition that makes it unique, inheriting the genetic composition from its parents through the genetic information in their chromosomes (DNA and genes).

modelons, imposing self-division or aggregation, in order to adapt and react to the requirements imposed by the environment [Sousa et al., 1999].

Fractal Factory

The Fractal Factory paradigm is based on mathematical fractal concept and theory of chaos, introducing a set of new concepts which aim to solve the organisation lack of flexibility to react to external and/or internal changes [Warneke, 1993]. The Fractal Factory is an open system, which consists of independent self-similar units, so called fractals. The fractal manufacturing uses the ideas of mathematical chaos: the companies could be composed of small components, the fractal, which have the capacity to react and adapt quickly to the new environment changes. The explosion of fractal objects into other fractal objects, has the particularity of generating objects which possess organisational structure and objectives similar to the original ones. A fractal object has the following features [Warneke, 1993]:

- *self-organised*, which means that it doesn't need external intervention to reorganise itself.
- *self-similar*, which means that one object in a fractal company is similar to other object. In other words, self-similar means that all objects contain a set of similar components and share a set of objectives and visions.
- *self-optimised*, which means that continuously increase its performance.

For [Warneke, 1993] the factory of the future will present a different dynamic organisational structure, being project oriented in contrast with the traditional function oriented organisation. This approach implies that the organisational structure will encapsulate the process and the technology. The fractal factory will not have a predefined organisation, but a set of resources with static capabilities, and a very dynamic set of projects (or tasks), which combine dynamically in multiple alternatives, in order to react quickly and efficiently to the external requirements. The conceptual behaviour of the system is described by the introduction of a new project in the system, which initiates a very dynamic process, responsible for resource/task negotiation, leading to constant changes in the enterprise structure and organisation.

2.6.2 Basic Concepts of Holonic Manufacturing

In middle of sixties, Arthur Koestler [Koestler, 1969] introduced the word holon to describe the basic unit of organisation in living organisms and social organisations, based on Herbert Simon theories and on his observations. Simon observed that complex systems are hierarchical systems formed by intermediate stable forms (see the parable of the watchmakers[10]), which do not exist as auto-sufficient and non-interactive elements but, on the contrary, they are simultaneously a part and a whole. Koestler concluded that parts and wholes do not exist in domain of life, and proposed the word holon to represent this hybrid nature, being a combination of the Greek word *holos*, which means *whole*, and the suffix *on*, which means *particle* [HMS, 2004]. Holons combines the whole and the part, being simultaneously self-contained wholes to their subordinated parts, and dependent parts when seen from higher levels (Janus effect). The property that a holon can be part of another holon, allows to break a holon into several others holons, which in turn can be broken into further holons, allowing the reduction of the problem complexity.

Koestler also identified two important characteristics of a holon [HMS, 2004]: autonomy, where the stability of the holons result from their ability to act autonomously in case of unpredictable circumstances, and cooperation, which is the ability to have holons cooperating, transforming these holons into effective components of bigger wholes.

The Holonic Manufacturing System (HMS) is a research initiative to develop next generation of manufacturing system, included as part of the international IMS programme. HMS is a paradigm that translates to the manufacturing world the concepts developed by Koestler to living organisms and social organisations, mainly those that are complex hierarchical systems formed by intermediate stable forms.

A holon is defined by the HMS consortium [HMS, 2004] as

> *An autonomous and cooperative building block of a manufacturing system for transforming, transporting, storing and/or validating information and physical objects.*

A holon can represent a physical or logical activity, such as a robot, a machine, an order, a flexible manufacturing system, or even an human operator [Leitão and Restivo, 1999]. The holon has information about itself and the environment, containing an information processing part and a physical processing part when the holon represents a physical device, such as an industrial robot [Winkler and Mey, 1994].

A holarchy is defined as a system of holons, organised in a hierarchical structure, cooperating to achieve the system goals, by combining their individual skills and knowledge. Each holarchy

[10]The parable tells the story of two excellent watchmakers Tempus and Hora. While Hora is getting richer and richer, Tempus is getting poorer and poorer. A team of analysts makes a visit to both shops and noticed the following. Both watches consists of 1000 parts, but Tempus designed his watch such that, when he had to put down a partly assembled watch it immediately fell into pieces and had to be reassembled from the basic elements. Hora had designed his watches so that he could put together subassemblies of about ten components each. Ten of these subassemblies could be put together to make a larger sub-assembly. Finally, ten of the larger subassemblies constituted the whole watch. Each subassembly could be put down without falling apart.

has fixed rules and directives, and a holon can dynamically belong to multiple holarchies at the same time, which is an important difference to the traditional concept of hierarchies. The holons can integrate themselves into a holarchy and, at the same time, to preserve their autonomy and individuality. A HMS is a holarchy that integrates the entire range of manufacturing elements, such as machines, products, parts and automated guided-vehicles. In HMS, the holon's behaviours and activities are determined through cooperation with other holons, as opposed to being determined by a centralised mechanism.

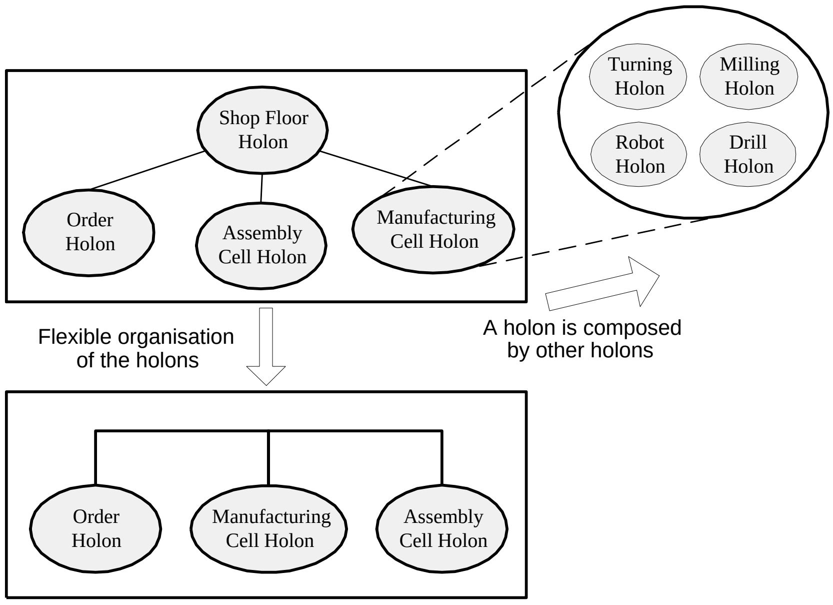


Figure 2.18: Holonic Features: Adaptation and Holon Expansion

A holon can be part of another holon, or a holon can be broken into several others holons, which in turn can be broken into further holons. An example, illustrated in Figure 2.18, is the representation of a shop floor control system, which has a holon designated by *manufacturing cell holon*, constituted by a set of other holons: *robot*, *milling, turning* and *drill holons.*

HMS can be used to implement control structures that combines the advantages of hierarchical and heterarchical approaches, showing the reactivity against disturbances presented in heterarchical control, and the high and predictable performance presented in hierarchical control [Bongaerts et al., 1998].

The implementation of the holonic manufacturing concepts can be done using the agent technology, which is appropriate to implement the modularity, decentralisation, re-use and complex structures characteristics. The use of agent technology addresses the high-level of abstraction [Marík et al., 2002]. At the lowest real time control level, the interconnection with physical devices is required, making it able to read data from sensors and to send actions to actuators. Currently, the lowest real time control is usually carried out by industrial PLC's with control programs developed using IEC 1131-3 programming languages[11]. The IEC 61499[12] is an extension of the IEC 1131-3 function block diagrams developed within the holonic research [Holobloc, 2003], presenting enough potential to replace in future the IEC 1131-3 based programs, only waiting that industrial PLC's will support their specifications.

IEC 61499 defines a new way to model the control and execution of algorithms in distributed control systems, by encapsulating and reusing software modules [Holobloc, 2003]. Within this model, the old concept of function block is used in order to make a clear distinction between the event and the triggered algorithms, making easier the verification of time properties [Vyatkin and Hanisch, 2003]. A function block, which is the fundamental unit of software encapsulation and reuse in IEC 61499, encapsulates the control algorithm with physical interfaces, communications, human interfaces, monitoring and diagnostics, and information technology-based services. Function blocks are able to provide a software solution to a small problem, such as the control of a valve, or control of a major unit of a plant, such as a complete production line. The advantages of using the function blocks concept are the improved software productivity through re-use of standard solutions (function block libraries) and the improvement in design flexibility through the ability of plug-and-play devices from different vendors.

2.6.3 Agent-Based and Holonic Manufacturing

The agent-based and holonic manufacturing paradigms were developed under the same fundamental principles of autonomy and cooperation, exploring the distribution and decentralisation of entities and functions. Although the similarity of the holon and agent concepts, some discussion is carried out to identify their differences, such as described in [Marík et al., 2002]. Here, some distinctions are pointed out.

In terms of origin, the agents have their roots in the computer science (artificial intelligence area) and the holons in the computer integrated manufacturing domain, focusing the problems associated to the flexible manufacturing systems. In conceptual terms, the holon is a concept and an agent is both a concept and a technology, being possible to implement the holon concept

[11]IEC 1131 consists of five parts: General information, Equipment and test requirements, PLC programming languages, User guidelines, and Communications. IEC 1131-3 is the international standard for programmable controller programming languages. As such, it specifies the syntax, semantics and display for the following suite of PLC programming languages: Ladder Diagram (LD), Sequential Function Charts (SFC), Function Block Diagram (FBD), Structured Text (ST) and Instruction List (IL). The ladder logic diagrams are the most popular and used programmable controller programming language.

[12]Also known as function blocks.

and holonic manufacturing systems using agent technology.

In terms of integration of physical resources, agents normally represent software components and the focus is not the integration of physical resources. In the manufacturing world this issue is critical and the holon concept supports the integration of physical resources, based on the feature that a holon comprises logical and physical components. In terms of human integration, the human interface is automatically embedded into each holon, while in the agent approach, the human interface is represented by a separated agent. The same occurs for the graphical user interfaces.

In terms of real time control, an agent cannot guarantee the real time constraints, while the holons must meet the hard real-time constraints required to achieve reliable system operation. At last, exploring the principle that a holon can represent simultaneously a whole and a part of the whole, the holon can be composed by several lower-level holons; an agent doesn't supports this feature [Bongaerts, 1998].

These differences tend to be reduced because at the moment, the agent and holonic communities are converging rapidly. This convergence is reflected in the efforts done by FIPA and HMS consortiums to introduce guidelines and specifications that support the holonic requirements in the FIPA specifications, through the FIPA Product Design and Manufacturing working group [Marík et al., 2003].

2.6.4 Research Work on Holonic Manufacturing

The introduction of holonic manufacturing concepts has being carried out by several research teams, in different application domains, such as manufacturing scheduling and control, materials handling, machine controllers and assembly systems.

In the manufacturing planning domain, [Hasegawa et al., 1994] present one of the first holonic approach to the manufacturing planning and scheduling systems. They propose a holonic planning and scheduling system for a simulated robotic assembly test bed and coordination mechanisms for scheduling, based on Lagrangian relaxation. The IntaPS project [Denkena et al., 2002] presents an approach for integrated process planning and production control. The IntaPS architecture consists of two main components, which link information systems of earlier stages of product development and the resources on the shop floor. This link is realised by decentralised planning on shop floor level and by rough level process planning.

In the manufacturing scheduling and control domain, [Sousa and Ramos, 1999] propose a dynamic scheduling system supported by a holonic approach, using forward and backward influence in the negotiation leading to the task allocation, to handle the temporal constraints and to solve conflicts. The architecture is composed by holons to represent resources, tasks, planning systems, etc. [Gou et al., 1998] define a scheduling algorithm based upon Lagrange relaxation concepts. It requires a centralised coordination that guides the individual holons to improve their schedule. [Markus et al., 1996] propose a market model to solve dynamic order processing and scheduling problems, such as conflicts between local scheduling agents, resolved by negotiation simple terms of tasks, due dates and prices. [Heikkilä et al., 1997] propose an holonic approach for manufacturing scheduling and control in a manufacturing cell. [Sugimura et al., 1996] model the manufacturing operations using an object-oriented approach and propose a real time scheduling mechanism for assembly lines.

[Brennan et al., 1997] use a hybrid control architecture, designated by PDCH (Partial Dynamic Control Hierarchy), combining the hierarchical and heterarchical approaches, for manufacturing control. Later, PDCH was used with IEC 61499 Function Block model to achieve a general approach for dynamic and intelligent reconfiguration of real-time distributed manufacturing control systems [Brennan et al., 2002]. [Fisher, 1999] uses a holonic approach for planning and control, builded upon the InteRRap (Integration of Reactive behaviour and Rational Planning) hybrid agent architecture [Müller, 1996], consisting of the production planning and control, shop floor control system, flexible cell control, autonomous systems and machine controller levels. The architecture uses agents to represent holonic manufacturing components, forming a multi-agent system organised in a hierarchical structure based in rules. [Tönshoff and Winkler, 1996] introduce the holonic concepts for the shop floor control.

In the materials handling domain, [Christensen, 1994] describes the initial ideas of holonic manufacturing systems, applying it to the materials handling domain. [Fletcher et al., 2002] present a holonic approach to inventory management developed at the Rockwell Automation Research Center Prague, where each item of inventory is treated as a holon. An application of that holonic approach, focusing mainly the transportation of workpieces among different manufacturing cells using AGVs, is developed using a multi-agent material handling system.

PROSA (Product-Resource-Order-Staff Architecture) [Brussel et al., 1998, Wyns, 1999] is a holonic reference architecture for manufacturing systems, which uses holons to represent products, resources, orders and logical activities. This architecture is based in three types of basic holons: product, order and resource. The resource holon contains the resource and an information processing part that controls the resource. The product holon holds the process and product knowledge, and contains all information about the product. The order holon represents the tasks in manufacturing systems. Additionally, there are the staff holons, whose mission is to assist and advice the basic holons; an example of staff holon is the scheduler. This architecture combines the predictability and the robustness of the hierarchical control with the high reaction to disturbances of heterarchical control.

Holonic concepts have been also introduced in other manufacturing areas. As an example, [McFarlane et al., 1995] apply holonic control to continuous processes such as steel manufacturing. In the machine controller domain, [Tanaya et al., 1997] apply holonic concepts in the development of machine controllers, which are more flexible and open than traditional NC technology. This holonic behaviour is supported by advanced planning, execution and monitoring actions. [Wang and Norrie, 2001] apply holonic concepts to distributed process planning and use function blocks as a new CNC controller language, allowing an automatic control application generation. [Arai et al., 2001] present a holonic assembly system that addresses sudden changes and breakdowns of assembly devices using the Plug and Produce concept. [Fletcher et al., 2003] describe the implementation of a holonic packing cell using JACK Intelligent Agents^{TM}platform. [Chirn and McFarlane, 2000] present a specific holonic control system architecture, HCBA (Holonic Component-based Architecture), to enable a smooth migration between the available standard control hardware and the system needed to implement holonic control.

2.7 Lessons Learned and Trends in Manufacturing Control

During the last decades, the manufacturing emerged from the mass production, idealised by Henry Ford, to the era of mass customisation, presented at the moment. In the current environment, the manufacturing enterprises must consider new product design, manufacturing and management strategies, asking for new systems to perform the control and supervision of distributed manufacturing. The adoption of new manufacturing concepts combined with the implementation of emergent technologies, is the answer to improve productivity, quality and product diversity, and to the decrease of price and delivery time.

The CIM paradigm was a promising approach aiming to integrate the automation islands through a centralised approach of the information system. However, this objective is not realistic because the experience obtained with the few experiences in the implementation of CIM systems was not satisfactory, mainly due to technical, heterogeneity, economical and social problems.

New emergent paradigms for distributed manufacturing systems, like the holonic manufacturing, seems to fulfill the gap left by the CIM approaches and the current economical, technological and market trends. In contrast to the heterarchical shop floor control and scheduling approaches, the holonic control solutions proposed present a compromise between hierarchical and heterarchical methods, by introducing hierarchy in decentralised systems.

In spite of its promising perspective, the holonic manufacturing achievements leave some important open questions.

Related to the control, it is possible to verify that some approaches use purely heterarchical architectures, while others use holonic architectures combining the hierarchical and heterarchical architectures. The application of holonic architectures by itself does not solve the manufacturing problems, being necessary to combine this type of architectures with mechanisms that allow the dynamic structure re-configuration and mechanisms to react quickly and efficiently to disturbances, minimising the effects of the disturbance, both in terms of time and in terms of propagation to other entities. In other words, the open question is related to how it is achieved the global production optimisation in decentralised systems, since the holonic manufacturing approach is based in autonomous entities. In case of formation of hierarchies to achieve global optimisation, an other set of open questions are related with how temporary hierarchies are dynamically formed, managed and removed.

In order to integrate different holarchies, another set of open questions is related with the definition of common ontologies to support inter-operability and knowledge sharing during the interaction processes. This inter-operability has two different levels: inter-operability within the same control platform and a more complex inter-operability related to the integration of different (distributed) control platforms.

In order to adapt to disturbances, the implementation of self-organisation capabilities and the integration of planning, scheduling and plan execution functions, are yet far from trivial. The definition of how the learning capabilities of each holon should be improved to support the manufacturing evolution and emergency, also remains an open challenge.

The connection with physical devices still present some gaps in the developed approaches, mainly due to the heterogeneousity of the manufacturing devices.

The formal modelling and validation of the holonic systems specifications assumes a critical role, with little attention devoted to it within the holonic comunity. This is particularly true in the specification of the dynamic behaviour of holonic systems and in the specification of the interaction between distributed holons to reach the manufacturing control functions.

Holonic manufacturing systems are frequently referred as performing well in presence of disturbances but there has been little reported evidence of that. An open question is to have a performance measurement methodology that evaluates and compares the increase of flexibility and agility when handling the occurrence of unexpected disturbances.

In the next chapter, a holonic manufacturing control architecture will be introduced, addressing the requirements and challenges described in this chapter. The proposed architecture focus on the the agile reaction to disturbances maintaining the same levels of production optimisation, through the use of dynamic control structure re-organisation.

Chapter 3

An Adaptive Holonic Control Architecture

"The significant problems we face cannot be solved at the same level of thinking we were at when we created them."
Albert Einstein

The current manufacturing control systems respond weakly to the emergent challenges faced by the manufacturing systems, since their capability to adapt with agility to unexpected internal disturbances[1] and to external environment volatility is poor. This weakness is mainly due to the rigidity of the control architectures.

The manufacturing research community is answering to this challenge by developing manufacturing control systems that present more agility and flexibility, and higher robustness against disturbances.

In the manufacturing context, and as defined in the previous chapter, the system flexibility is the capability to adapt to new, different or changing environments. On the other hand, agility is related to the ability to react to production environment changes in short time. According to the APICS Dictionary [APICS, 1995], the robustness of the manufacturing control system is related with the capability to remain working correctly and relatively stable with a minimum of variation, even in presence of factors that influence the operation.

[1]In the context of this work, a manufacturing disturbance can be defined as an unexpected disruption that affects the production.

The architectures of these new manufacturing control systems must have the ability to adapt to change, but must not result in systems that are *inflexible, fragile and difficult to maintain* [Bussmann, 1998], being composed of partial dynamic hierarchies that can preserve the stability of hierarchy while providing the dynamic flexibility of heterarchical control approaches.

The ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) holonic architecture addresses the agile reaction to emergence and change, increasing the agility and flexibility of manufacturing control systems, specially those located in volatile environments characterised by the frequent occurrence of disturbances.

The focus of ADACOR architecture is the shop floor level, according to the layer approach described in section 2.2.5, and specially the flexible manufacturing systems organised in job shop production type, characterised by concurrent and asynchronous processes with non-preemptive operations and alternative routings.

The proposed adaptive architecture intends to be as decentralised as possible and as centralised as necessary, i.e. using a centralised approach when the objective is the optimisation, and a more heterarchical approach in presence of unexpected events and modifications. In these circumstances, ADACOR proposes the decomposition of manufacturing control functions into a community of autonomous and cooperative entities, taking advantage of modularity, decentralisation, agility, flexibility, robustness and scalability. The introduction of supervisor entities allows the presence of hierarchy in decentralised systems, to achieve global production optimisation. The introduction of self-organisation capabilities associated to the distributed entities, allows the dynamic evolution and re-configuration of the organisational control structure, combining the global production optimisation with the agile reaction to unexpected disturbances.

As referred, besides the flexibility and agility, ADACOR aims to improve the adaptation characteristics of manufacturing control systems. According to the Merriam-Webster dictionary [Merriam-Webster, 2003], adaptation can be defined as:

1. *the act or process of adapting;*

2. *adjustment to environmental conditions: as*

    - *adjustment of a sense organ to the intensity or quality of stimulation;*

    - *modification of an organism or its parts that makes it more fit for existence under the conditions of its environment.*

From the Britannia Encyclopaedia [Britannica, 2003], adaptation is *"in biology, the process by which an animal or plant becomes fitted to its environment; it is the result of natural selection acting upon heritable variation. Even the simpler organisms must be adapted in a great variety* *of ways: in their structure, physiology, and genetics; in their locomotion or dispersal; in their means of defense and attack; in their reproduction, ...”*

In this work, in accordance to the previous definitions, adaptation means the capability to adjust the control behaviour to the environmental conditions during the system life-cycle, maintaining the pursuit of its objectives. The more adaptable a system is, the more agile is the reaction to unforeseen events or situations. The adaptation can be performed in different ways, either by changing the control system structure or adjusting decision parameters.

In this chapter the main concepts of the ADACOR architecture will be described, mainly the components of the system, the adaptive control approach, the ontology used by the architecture components, the interactions between the components to perform the manufacturing control functions and the architecture of a generic ADACOR holon.

3.1 Architecture Components

The holonic manufacturing paradigm is well suited to deal with manufacturing control problems in a distributed manner. The ADACOR architecture is based on the holonic manufacturing system paradigm, using the advantages of its main concepts, such as presence of distributed structures with autonomous and cooperative entities, and the partial dynamic hierarchies forming intermediate stable forms. The use of holonic concepts is complemented with the introduction of some ideas derived from bionic manufacturing systems, such as the role of supervision and the dynamic evolution of the system.

ADACOR architecture is build upon a set of autonomous and cooperative entities, designated by holons, to support the distribution of skills and knowledge, and to improve the capability of adaptation to changing environments. Each holon is a representation of a manufacturing component that can be either a physical resource (numerical control machines, robots, programmable controllers, pallets, etc.) or a logic entity (products, orders, etc.). The manufacturing components are holonified to implement a behaviour that represents the manufacturing component objectives and functionalities, each holon being responsible for carrying out different specific functionalities.

3.1.1 ADACOR Holon Classes

ADACOR architecture groups the manufacturing holons into product, task, operational and supervisor holon classes [Leitão and Restivo, 2003a], as illustrated in Figure 3.1. This generalisation is based in grouping the factory components according to their functions and objectives, like in the specialisation concept from the object-oriented paradigm.

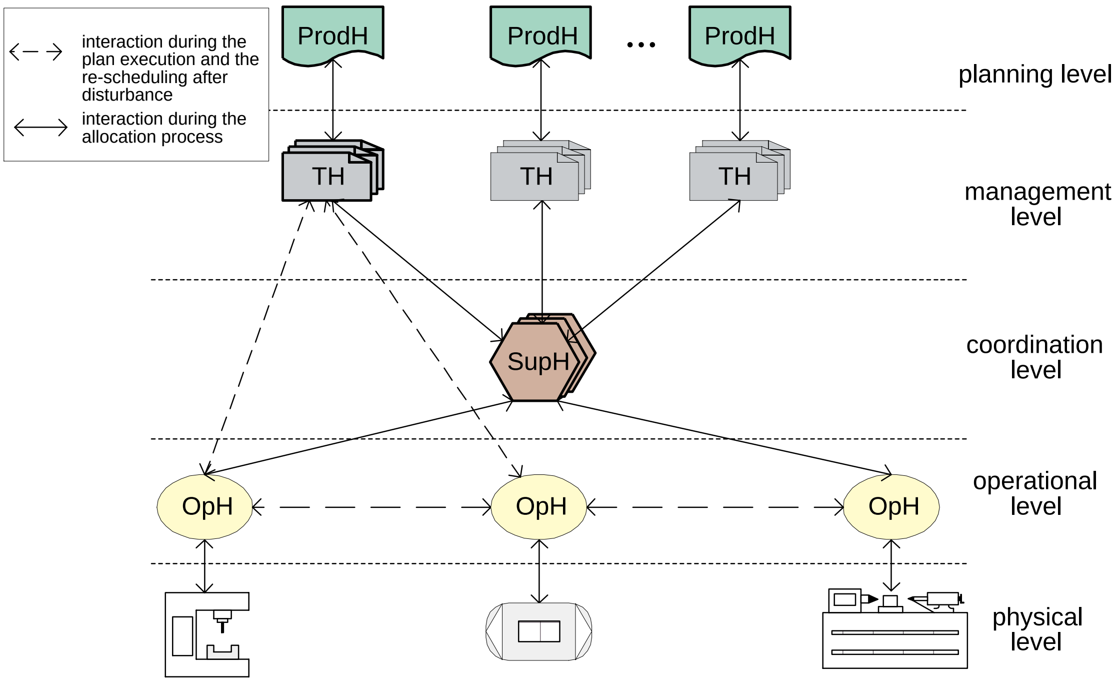

interaction during the plan execution and the re-scheduling after disturbance
interaction during the allocation process
ProdH
ProdH
...
ProdH
planning level
TH
TH
TH
management level
SupH
coordination level
OpH
OpH
OpH
operational level
physical level

Figure 3.1: ADACOR Holon Classes

The supervisor holon is inspired in biological systems and presents different characteristics from the staff holons defined in PROSA reference architecture [Brussel et al., 1998], while the product, task and operational holons are quite similar to the PROSA product, order and resource holons.

Each product available to be produced in the factory plant is represented by a product holon that contains all knowledge related to the product and is responsible for the short-term process planning. The product holon is not the major focus of the architecture, since this work addresses the shop floor level, and it is only necessary to be the bridge between the shop floor and the planning level. In the ADACOR architecture, the product holon also contributes to the integration of all the manufacturing control functions, i.e. the planning, scheduling and plan execution.

Each production order launched to the shop floor to execute a product (or sub-product) is represented by a task holon, which is responsible to manage its execution, containing the dynamic information about the production order.

The operational holons represent the physical resources available in the shop floor, such as operators, robots and numerical control machines, managing their behaviours according to the resource goals and skills [Leitão and Restivo, 2002b]. The operational holon manages the agenda of the resource, i.e. the planned list of work orders that the manufacturing resource has to execute over the time.

The supervisor holon introduces coordination and global optimisation in decentralised control approaches and is responsible for the group formation and coordination.

3.1.2 Supervisor Role

The organisation of holons in stable hierarchies is frequently associated to the global performance optimisation in normal operation. The presence of different hierarchy levels requires the existence of coordinating entities, to combine synergies, to aggregate the skills of each member of the group and to offer the combined services to other entities in the manufacturing system.

In ADACOR architecture, this role is executed by the supervisor holon, which major function is the elaboration of optimised production plans for the holons under its coordination domain. The supervisor holons are also responsible for the constitution of groups and their dynamic evolution according to the environment context, based in pre-defined clusters of holons.

The set-up of a group consists in the addition of holons to a group coordinated by a supervisor holon. The intention to create the group can come from the need to combine synergies aiming to optimise the production or from the existence of geographical constraints. Once a new member is added to the group, the supervisor holon aggregates the information related to the new member using the tuple {holon, skills}, where the *skills* parameter is the list of skills that the holon contributes to the group. These groups can be formed to build a shop floor, a manufacturing cell, or a machine equipped with a set of tools, assuming the supervisor holon the role of group coordinator.

The dynamic evolution of the group is supported by changes in the logical control structure and modification of the group constitution (through the addition or remotion of members), being regulated by pre-defined rules. Its efficiency is dependent of a well-defined set of rules that try to imitate the desired adaptation mechanism.

In ADACOR, an operational holon can be made of a set of several operational or supervisor holons, allowing to build fractal holarchies, as illustrated in Figure 3.2. In this case, the supervisor holon acts as the logic component, and the several operational holons act as the physical part of the holon.

As an example, a manufacturing cell can be represented by an operational holon that constitutes of several other operational holons, each one representing a manufacturing resource, and one supervisor holon representing the manufacturing cell controller. Additionally, each one of these operational holons, that represent a manufacturing resource, could be constituted of several other operational holons, such as the numerical control machine itself and the several tools stored in its tool magazine.

The high flexibility of this control structure organisation, creates the foundations to support

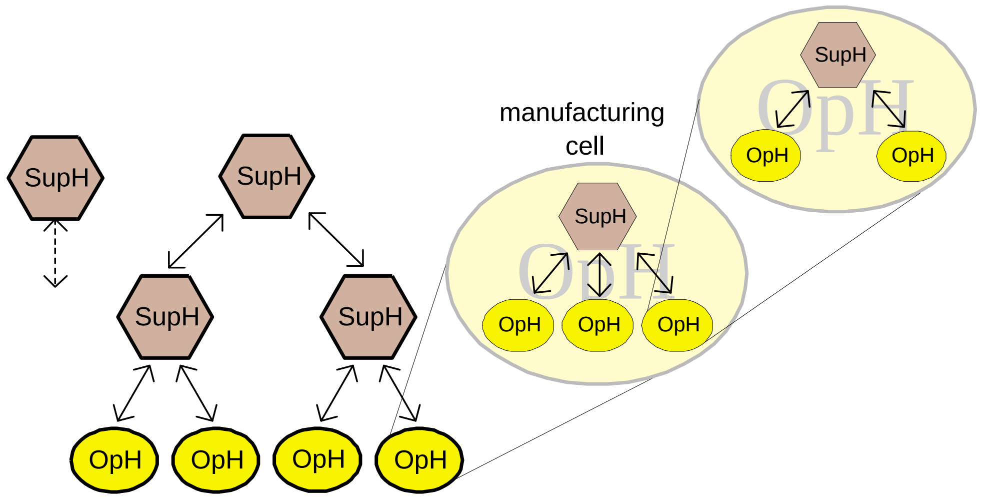


Figure 3.2: "Fractal" Feature in ADACOR Approach

the combination of the global optimisation with the agile reaction to unexpected disturbances [Leitão and Restivo, 2003b]. Additionally, it allows a modular development through the encapsulation of several functions or manufacturing components.

3.1.3 Characteristics and Motivation of ADACOR Holons

A holon, like an agent, acts based in the local knowledge, by sensing the manufacturing environment, triggering the reasoning process, which selects the proper actions to be executed and that will affect the manufacturing environment. Each holon in ADACOR architecture possess the following main features to act in the volatile manufacturing environment: autonomy, cooperation, reactivity and pro-activity behaviour, adaptation and learning.

A holon is **autonomous**, since it can operate without the direct intervention of external entities, and has some kind of control over its behaviour. Having its own objectives, knowledge and skills, each holon has the capability to reason in order to take decisions about its activities. Each ADACOR holon possesses only a partial view of the system, needing to **cooperate** with the other holons in order to achieve its goals or to get additional information about the system, sharing knowledge in order to transform local knowledge into global knowledge.

The **reactivity** and **pro-activity behaviours** are crucial in the design of the holons. ADACOR holons perceive their environment and response quickly to changes that occur on it, reacting to the stimulus provided by the environment. In spite of the predominant reactive behaviour, ADACOR holons do not simply act in response to their environment, but are also able to take the initiative, for example elaborating plans for the product execution and predicting the occurrence of future disturbances.

The holons in ADACOR architecture can be organised in different organisational structures, from a more hierarchical structure to a more decentralised structure, having the capability to dynamically re-organise into different organisational structures. The **self-organisation** associated to each holon contributes for the dynamic adaptation of the system to unexpected disturbances, namely changing in customer demands, economical trends or machine breakdowns. The degree of efficiency of the self-organisation capability is dependent on how the learning mechanisms are implemented. The **learning** capability associated to ADACOR holons allows the acquisition of new knowledge, improving the holon's ability to act in future and to support the dynamic evolution of the environment where it is placed.

ADACOR holons use the **plug and produce** concept, being then possible to add a new element to the system without the need to re-initialise and re-programme the system, allowing high flexibility to adapt to the system re-configuration. Each holon is autonomous and can enter or leave the system at any time, being only necessary that the system as a whole does not feel that disturbance[2].

Table 3.1: Evolution of Credits during the Holon Life Cycle

Phase	**Task Holon**	**Operational Holon**
Operation allocation process.	Contracts the operation execution by ξ and the penalty by φ.	Contracts the operation execution by ξ and the penalty by φ.
Finish of an operation with success.	Pays the value ξ to the OpH ($\pi \leftarrow \pi - \xi$).	Increases the total credits by ξ ($\mu \leftarrow \mu + \xi$).
End of an operation with delay.	Pays the value ξ and receives the value φ from the OpH ($\pi \leftarrow \pi - \xi + \varphi$).	Decreases the total credits by φ and increase by ξ ($\mu \leftarrow \mu + \xi - \varphi$).
Operation cancelled (delay, failure, etc.)	Receives the value φ from the OpH ($\pi \leftarrow \pi + \varphi$).	Decreases the total credits by φ ($\mu \leftarrow \mu - \varphi$).

The **motivation** of ADACOR holons to execute the manufacturing actions is regulated by a

[2]When a holon enters in the system it announces and offers its services to the other holons. When a holon leaves the system, the other holons should be able to found alternative plans to overcome the missed holon.

credits system. Table 3.1 summarises the evolution of the credits of task and operational holons during their life cycles.

When the task holon is launched, it receives a fund to execute the production order (π) and a penalty value for delay. The task holon manages the costs to produce its production order, in order to guarantee that they never exceed the initial fund.

During the interaction to allocate the work orders, the task holons try to pay as less as possible and the operational holons try to receive as more as possible. After the negotiation, the task holon accepts to pay a price of φ credits to the operational holon that will execute the work order and to receive a penalty of ξ credits from the operational holon if it does not fulfil the contracted due date.

The global performance of the operational holons in terms of credits is given by the sum of rewards received minus the penalties paid for the delays. These rewards and penalties reflect the trust in the holon and are taken in consideration during the allocation process.

3.1.4 Identification of ADACOR Manufacturing Holons

The effort to develop an appropriated methodology to identify holons, meeting the production control requirements, has led to some research work in this area, described in [Ritter et al., 2002, Brussel et al., 1999, Bussmann et al., 2000]. Questions like which manufacturing components should be considered as holons, and how to holonify those manufacturing components, are frequently asked by those who develop holonic applications.

The procedure to identify the ADACOR holons, from the manufacturing components, is based in the generalisation and aggregation concepts [Leitão and Restivo, 2003a].

The generalisation concept means that a certain holon can inherit the attributes and methods of one more general holon. As an example, it is possible to define several specialised holons, such as the producer, transporter, mover and maintenance holons, from the operational holon. Each one of those holons extends the operational holon by the addition of more specialised attributes and methods.

Besides the generalisation, the identification procedure uses the physical and logical aggregation of manufacturing entities [Leitão and Restivo, 2003a]. The aggregation concept means that it is possible to aggregate holons to form a larger holon with its own identity, knowledge and skills. Examples of this concept are found in the aggregation of sensors, actuators and tools into a manufacturing machine holon, or in the aggregation of several manufacturing machines and transport machines into a flexible manufacturing system holon.

The following sections will describe how to identify the several ADACOR holon classes, by analysing the specifications of the manufacturing system, as it is summarised in the Figure 3.3.

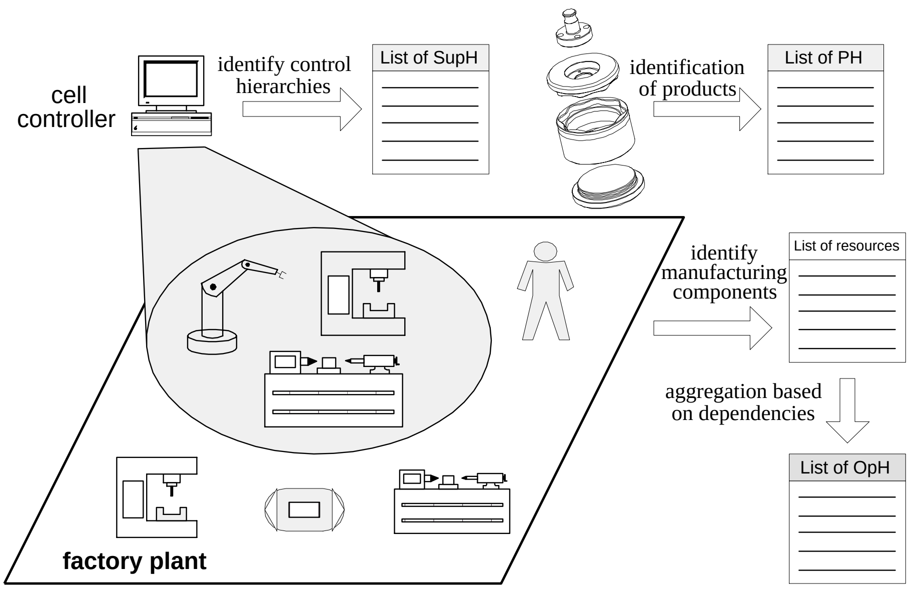

cell controller
identify control hierarchies
List of SupH
identification of products
List of PH
identify manufacturing components
List of resources
aggregation based on dependencies
List of OpH
factory plant

Figure 3.3: Holonification Methodology

Identification of the Operational Holons

The shop floor resources can be classified into:

- processing machines $\mathcal{PM} = \{pm_w | w \in I\}$, such as numerical control machines, responsible for the execution of processing actions.
- operators $\mathcal{H} = \{h_w | w \in I\}$, such as human operators, responsible for the execution of manufacturing activities.
- transporter resources $\mathcal{T} = \{t_w | w \in I\}$, such as AGVs, responsible for the transport of parts between work stations.
- mover resources $\mathcal{M} = \{m_w | w \in I\}$, such as robots, responsible for handling the parts inside a workstation.
- tools $\mathcal{TO} = \{to_w | w \in I\}$, such as cutting tools, used by processing machines, or grippers, used by mover resources.
- batches of raw material $\mathcal{RM} = \{rm_w | w \in I\}$, such as pallets containing the parts required to produce the product.
- auxiliary resources $\mathcal{A} = \{a_w | w \in I\}$, such as CCD (Charge Coupled Device) cameras and buffers.

As the operational holons represent the manufacturing resources presented in the manufacturing system, the first step in the proposed procedure is to identify the shop floor resources that are located at the manufacturing system, finding the set of resources $\mathcal{R}$, by building the sets $\mathcal{PM}$, $\mathcal{H}$, $\mathcal{T}$, $\mathcal{M}$, $\mathcal{A}$, $\mathcal{TO}$ and $\mathcal{RM}$.

For each resource object it is necessary to identify the set of skills and behaviours. The set of skills is defined by storing the list of features of the resource according to the tuple $\{name, value\}$. As an example, the element $\{spindleSpeed, 4000\}$ represents that the splindle of a resource can operate with a speed of 4000 rotations per minute. The behaviour is defined according to the type of resource and to the roles that it will perform in a certain group or collaboration.

The next step is related to the identification of the interactions between the identified resources, in order to extract the dependencies between the objects. Aggregating the objects according to the identified dependencies, for example, cutting tools in a machine and AGVs in a transport system, it is possible to build $\mathcal{R}'$, which is a sub-set of $\mathcal{R}$, that contains the list of resources aggregated by their dependencies, according to some aggregation level.

$$\forall r_x \in \mathcal{R}, r_x \notin \mathcal{R}' \Rightarrow \exists r_y \in \mathcal{R}' : r_x \prec r_y$$

where $\prec$ represents the dependency between two components.

From the set $\mathcal{R}'$ it is possible to extract a list of operational holons, called $\mathcal{OH}$, by mapping each $\mathcal{R}'$ object into an operational holon, indicating its attributes.

Identification of the Product Holons

The identification of the ADACOR product holons requires the identification of all the products and sub-products manufactured by the factory plant, building the set $\mathcal{P}$. The complete identification of a product object requires the description of the product structure (i.e. relations with other sub-products) and the description of the process plan.

The representation of the product structure is done using a set of elements formatted according to the tuple {*product*, *quantity*, *time*}, where *product* refers the name of a sub-product, *quantity* is the number of instances of the sub-product type, and *time* is the period of time required to execute the sub-product.

The reference to the product process plan requires the identification of the list of operations, $\Theta_i = \{o_{i1}, o_{i2}, o_{i3}, ..., o_{in}\}$, partially ordered according to their precedences, $\Omega_i = \{< o_{ij}, o_{ik} > |o_{ij} < o_{ik}\}$. Additionally, it is necessary to identify for each operation o_{ik}, the set of requirements and constraints that a machine should satisfy to execute the operation, $\mathcal{B}_{ik} = \{B_{ikw} | w \in I\}$.

The $\mathcal{PH}$ set is build by using a product holon to represent each element of $\mathcal{P}$, containing the product data model defined for each product object.

Identification of the Task Holons

The task holons are created dynamically according to the needs to produce the products, being not possible to define them at the design phase. When a product holon receives a request to produce the product that it represents, it launches a task holon, passing the due date, a process plan that defines the sequence of operations to execute the product and the fund to execute the production order.

Identification of the Supervisor Holons

The identification of the supervisor holons requires the analysis of the control system specifications, mainly the coordination levels of the control system. The $\mathcal{SH}$ set, which represents the list of supervisor holons, is built by assigning a supervisor holon to each identified coordinator object, such as a shop floor controller or a manufacturing cell controller.

The representation of the coordination domain associated to each supervisor holon, is based in the set of objects defined according to the tuple $\{holon, \mathcal{S}\}$, where *holon* parameter is the name of the holon that belongs to the coordination domain and $\mathcal{S}$ is the set of skills associated to the holon. These organisational structures are dynamically updated during the system life-cycle, with new operational and/or supervisor holons joining or leaving the group. The coordination relationships are maintained by the supervisor holon by adding or removing relationship instances of the tuple $\{holon, \mathcal{S}\}$.

Logical Dependencies

The last step is to identify the logic dependencies between the identified holons. These logic dependencies allow to remove objects from the previous identified sets, integrating them with other objects. As an example, a task holon can have a logical dependency with a part or a pallet, if the part or pallet is associated exclusively to the task holon.

The total number of manufacturing holons identified in the design phase will be the sum of the cardinality of $\mathcal{OH}$, $\mathcal{PH}$ and $\mathcal{SH}$ sets.

3.2 Adaptation to Change

The control architecture is a key factor for the performance of the manufacturing control system, assuming a critical role in the system performance in terms of response to change and capability to learn. The use of heterarchical control architectures introduces good reaction to disturbances but degrades the global production optimisation; on the other hand, the hierarchical approach presents good global optimisation but weak reaction to disturbances.

The challenge is to develop a dynamic and adaptive control approach that improves the agility and reaction to unexpected disturbances without compromising the global optimisation.

The self-organisation capability, inspired in biological systems, is the key concept to support the adaptative production control mechanism. In ADACOR approach, the adaptation is achieved by the self-organisation of ADACOR holons present in the system, contributing to the dynamic adaptation of the whole system. A small example to show the main idea of ADACOR proposal is illustrated in Figure 3.4.

In normal operation the production plan is generated by a centralised entity that introduces global optimisation. When a disturbance occurs, some work orders have to be re-allocated to other resources. This fast re-schedule is achieved by the direct interaction between the distributed entities. After this transient phase, the centralised entity enters again into action, eventually optimising the achieved schedule.

ADACOR makes this adaptive control mechanism work. The self-organisation capability presented in all ADACOR holons allows to balance the control between different control structures, reaching an adaptive control approach that combines the agile reaction to disturbances with the global optimisation.

3.2.1 Self-Organisation in Manufacturing Systems

In biological systems there are two different approaches to adaptation to the dynamic evolution of the environment [Vaario and Ueda, 1996]: evolutionary systems and self-organisation, that can be translate into the manufacturing control systems.

In the evolutionary approach, the nodes of the structure that represents the control system are encoded as genetic information, and are subject to the application of evolutionary techniques by selecting gradually a better system. A well known example of this approach is a neural network, where learning occurs evolutionary.

In the self-organising approach, the network of nodes that represents the control system is established by the nodes themselves. The driving forces drive the re-organisation process according to the environment conditions and to the control properties of the distributed entities.

The local self-organisation of each distributed entity can be defined as the capability to organise by itself into different structures, according to its perception of the environment. The global system self-organisation can be defined as the emergence of the global control or organisational structure as a result of the capability of local entities to change dynamically and autonomously

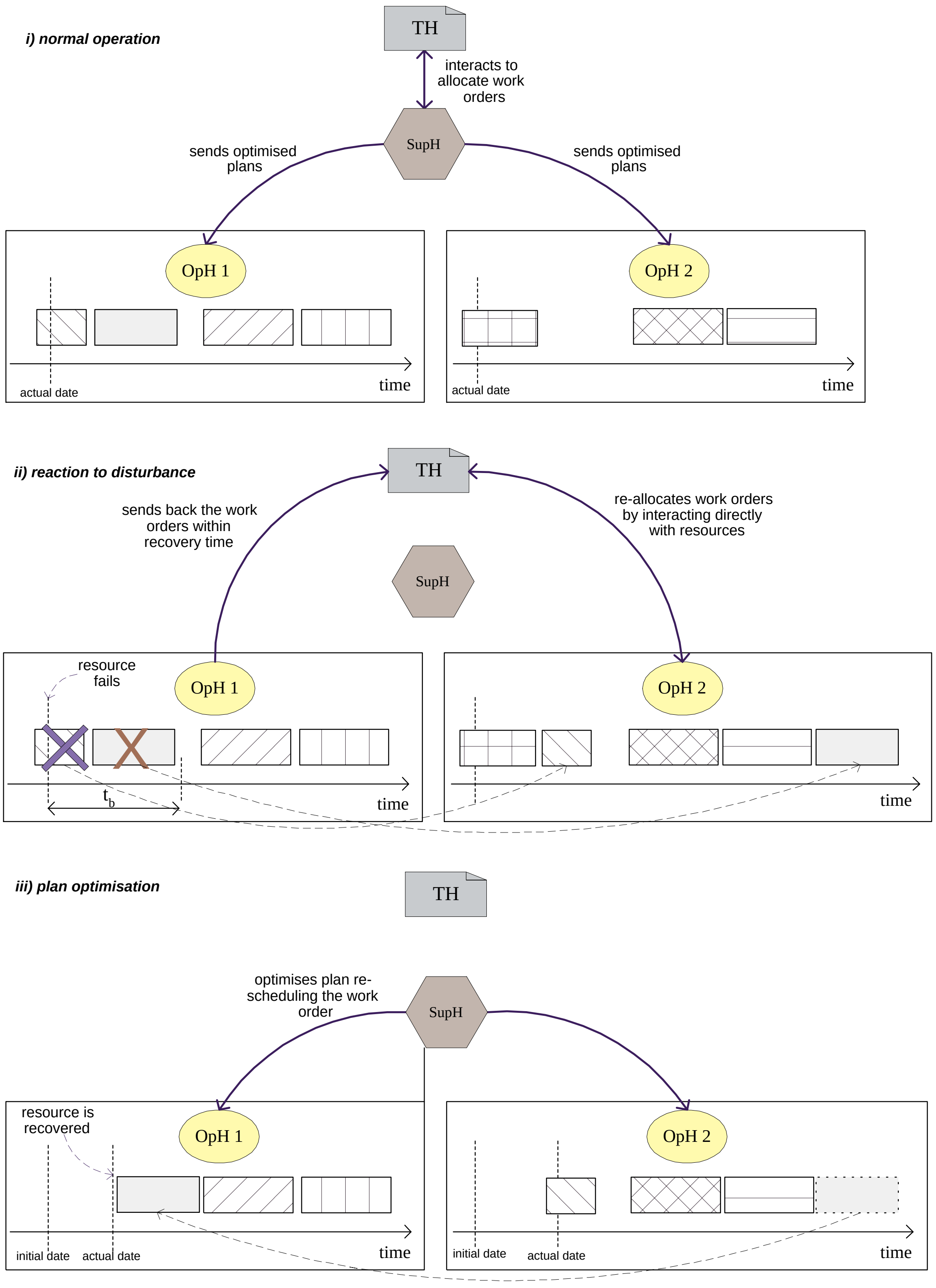


Figure 3.4: Adaptation in ADACOR Architecture

their properties. Self-organisation can contribute to adaptive manufacturing systems in the main following areas [Vaario and Ueda, 1997]:

- **Shop floor layout**, where the manufacturing entities present in the shop floor are movable, i.e. the producer and transporter resources moves physically in order to minimise the transportation distances.

- **Adaptive control**, where the goal is to find out an adaptive and dynamic production control strategy based in the dynamic and on-line schedule, adapted in case of occurrence of unexpected disturbances.

- **Product demand**, where the manufacturing system re-organises itself in order to adapt to the changes in the product demand, increasing or reducing the number of manufacturing resources, or modifying their capabilities.

In our work, the focus will be in the adaptive control, with self-organisation contributing for the re-organisation of the control system into different control architectures, more appropriated to the current situations.

As in the biological systems, where the evolution of the species or the groups results from the self-organisation of local entities, the adaptive ADACOR mechanism emerges in a bottom-up approach, built upon the individual self-organisation of manufacturing holons. Here, the dynamic adaptation of each holon to unexpected situations contributes to the adaptation of the system as a whole to the emergent contexts and to the quick reaction to the occurrence of unexpected disturbances.

As illustrated in Figure 3.5, in case of occurrence of an unexpected failure in one machine, that deviates the execution from the initial plan, each holon using its self-organisation capability, contributes to find out an alternative plan, by re-directing the flow of the product execution to alternatives machines.

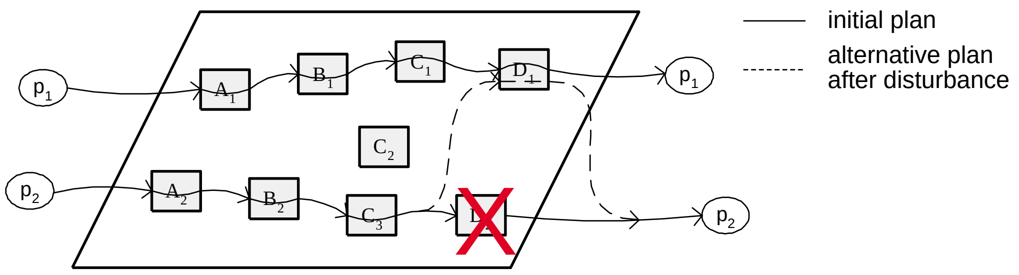


Figure 3.5: Self-Organisation to Support Adaptation to Disturbances

The self-organisation mechanisms require local driving forces to support the adaptation. In ADACOR architecture, the driving forces are the autonomy factor and the learning capability, which are inherent characteristics to each ADACOR holon. These mechanisms are complemented by global ones, namely the propagation mechanisms and the dynamic scheduling, which result from the interaction between distributed entities. In the following sections, the autonomy factor and the propagation mechanisms which are the basis for the self-organisation in ADACOR approach, will be described.

3.2.2 Autonomy Factor

The autonomy factor, α, associated to each operational holon, is a parameter that reflects the degree of autonomy of each holon, and evolves dynamically in order to adapt the holon behaviour according to its goals and constraints and with the environment where it is placed. The autonomy factor is a continous or discrete variable, regulated by a decision mechanism, representing the adaptation behaviour of each holon.

In this study, it was considered that the autonomy factor is a discrete binary variable comprising the states {Low, High}. The cardinality of the discrete set associated to the autonomy factor has strong impact in the dynamic adaptation mechanism: the higher is the number of discrete values, the more gradual will be the adaptation procedure. However, an high number of discrete values makes more complex the adaptation mechanism and requires the implementation of more complex decision mechanisms.

Normally, the operational holons have a {Low} autonomy factor, that allows to follow the supervisor holon coordination, accepting its schedule proposals [Leitão and Restivo, 2002c]. The supervisor coordination introduces the global optimisation in the system, since the supervisor holon has a wider view of the system than the operational holons under its coordination domain. The emergency, normally the occurrence of an unexpected disturbance, triggers the adaptation behaviour illustrated in Figure 3.6, and associated to each operational holon.

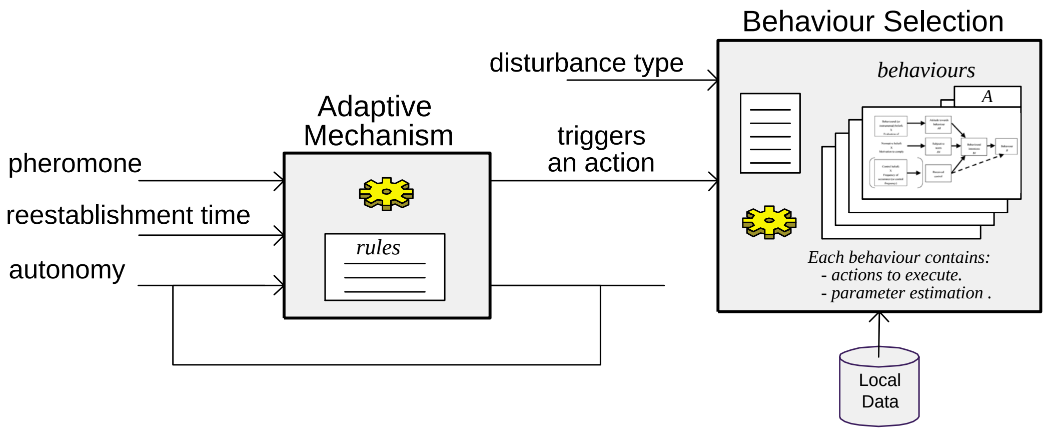


Figure 3.6: Adaptation Behaviour Model

The inputs of the adaptive mechanism are the autonomy factor, the reestablishment time (τ), that is the estimated time to recover from the disturbance, and the pheromone parameter (ρ), that is an indication of the occurrence of a disturbance (occurred in the local resource or propagated by other holons).

The adaptive mechanism determines the evolution of the autonomy factor and the action to trigger according to the Table 3.2, which represents the set of rules that regulates the adaptation behaviour of the holon, considering the discrete set of only two values for the autonomy and pheromone parameters.

Table 3.2: Behaviour of the Adaptive Mechanism

ρ	α	τ	New α	Action Triggered
High	Low	-	High	Trigger selection behaviour
High	High	Elapsed	High	Reload reestablishment time
Low	High	Elapsed	Low	Re-organise into default structure
-	High	Not Elapsed	-	-
Low	Low	-	-	-

In case of {Low} autonomy factor, the occurrence of an unexpected disturbance, represented by the {High} value associated to the pheromone parameter, triggers the change of the autonomy factor to {High} and the selection of one adequate behaviour to handle the disturbance. The behaviour selection, which will be described in chapter 5, is a function of the disturbance type, the actual state of the holon and the historic data. The behaviours that can be selected can include, for example, the re-organisation into a heterarchical control structure, the estimation of some required parameters, such as the reestablishment time, and the cancellation of the work order allocated to the resource.

When the reestablishment time has elapsed, if the autonomy factor is {High} and the pheromone is still active, which means that the disturbance is not completely recovered, the action triggered is the re-load of the reestablishment time. If the pheromone has already dissipated, which means that the disturbance is already solved, the holon can return to the original structure, changing the autonomy factor to {Low}. The other cases do not imply any action in the holon behaviour.

3.2.3 Propagation Mechanisms

The driving force associated to each individual holon to achieve adaptation was analysed in the previous section. However, the global self-organisation of the system is only achieved if the distributed entities have stimulus that drive their local self-organisation capabilities. The behaviour recalls the stimergy concept, which is often used in biology to describe the influence on behaviour of the persisting environmental effects of previous behaviours.

The global self-organisation requires global mechanisms that allow the interaction between local individual holons, supporting the propagation of the emergence and the need for re-organisation into different control structures.

The propagation mechanism defined in ADACOR uses the indirect communication principle and a pheromone-like spreading mechanism to distribute global information. The holons cooperating with this type of mechanisms should [Brussel et al., 2000]:

- Dissipate information to the other holons, similar to the ant that deposits pheromones.
- Sense the information dissipated by the other holons (like ants sensing the odours), in order to take their own actions, and sometimes reinforcing the odour for other holons.

In the ant-based mechanism the action taken by an holon can be the deposit of a pheromone or the reinforcement/adaptation of the odour, as illustrated in Figure 3.7. The flow field gradient is characterised by the reduction of the intensity of the odour and increase of the entropy. Some experiences have been undertaken using the ant-based interaction mechanism, such as described in [Parunak and Brueckner, 2001, Parunak et al., 2001, Brussel et al., 2000].

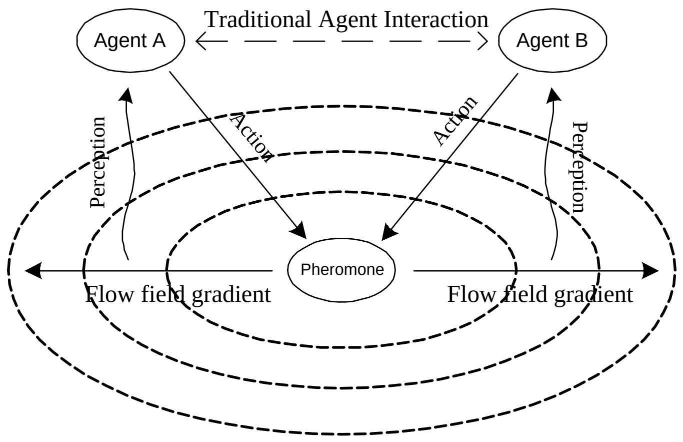


Figure 3.7: Ant-based Interaction (adapted from [Parunak and Brueckner, 2001])

In case of emergence or occurrence of an unexpected disturbance, the need for re-organisation is propagated through the deposit of a pheromone to the neighbour supervisor holons, as illustrated in Figure 3.8 [Leitão and Restivo, 2002a]. While spreading the need for re-organisation, the holon passes a parameter that reflects the estimated reestablishment time, similar to the odour from the pheromone-like techniques, which is forecasted according to the type of disturbance and to the historic data.

The holons associated to each supervisor holon receive the need for re-organisation by sensing the pheromone, propagating this need to neighbour holons. The intensity of the odour associated to the pheromone becomes smaller with the increase of the levels of supervisor holons (similar to distance in the original pheromone techniques), according to a defined flow field gradient.

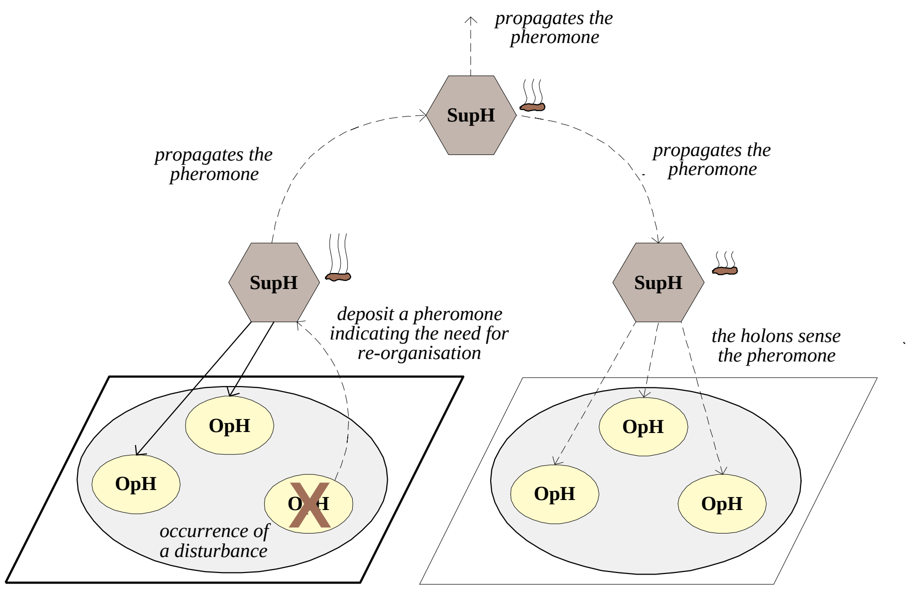

propagates the pheromone
SupH
propagates the pheromone
propagates the pheromone
SupH
SupH
deposit a pheromone indicating the need for re-organisation
the holons sense the pheromone
OpH
OpH
OpH
occurrence of a disturbance
OpH
OpH
OpH

Figure 3.8: Propagation of the Emergence using Pheromone-like Techniques

The propagation of the emergence and the need for re-organisation, using pheromone-like techniques, allows the dynamic and continuous adaptation of the system to disturbances, supporting the global self-organisation, reducing the communication overhead and improving the reaction to disturbances.

3.2.4 Adaptive Production Control

The ADACOR control approach is neither completely decentralised nor hierarchical, but balances between a more centralised approach to a more flat approach, passing through other intermediate forms of control [Leitão and Restivo, 2002c], thus implementing an agile and adaptive control approach, due to the adaptive and dynamic evolution of the autonomy of each ADACOR holon.

The proposed adaptive production control shares the control between supervisor and operational holons, and splits the control evolution into two alternative states: stationary state, where the system control uses coordination levels and the supervisor role to get global optimisation of the production process, and the transient state, triggered with the occurrence of disturbances and presenting a behaviour quite similar to the heterarchical approach in terms of agility and adaptability.

In **stationary state** the holons are organised in a federated architecture, with the supervisor holons representing cell controllers and/or shop floor controllers, and interacting directly with the task holons during the operation allocation process. The supervisor holon, as coordinator, elaborates optimised schedule plans that proposes to the task holons and to the operational holons within its coordination domain [Leitão and Restivo, 2002a]. The operational holons see these proposals as advices, having enough autonomy to accept or reject the proposed schedule. In this state, the autonomy factor of each operational holon is {Low}, allowing the operational holon to follow the proposals sent by the supervisor holon.

After the allocation of the manufacturing operations, the task holons interact directly with the operational holons during the execution of the operations, such as to ask for availability of space in the buffer. When an operational holon rejects one or more proposed operations, the supervisor holon must re-schedule the production plan, trying to find alternatives. The learning mechanisms allow that information related to the rejections will be used in future processes to elaborate optimised schedules.

If, for any reason, the system deviates from planned, due for example to a machine failure that provokes the destruction of the part that has been processed, the control system enters in the **transient state**. The transient state is characterised by the re-organisation of the holons required by the transition from the hierarchical control architecture to the heterarchical control architecture, allowing the agile reaction to disturbances of this control structure, as illustrated in Figure 3.9.

The operational holon which detects the disturbance tries to recover locally the failure, by analysing the symptoms and making a self-diagnosis, but if it cannot recover from the failure, it increases its the autonomy factor parameter to {High} and propagates the need for re-organisation to the other holons in the system, through the supervisor holon. According to the type of disturbance the operational holon also selects an appropriate behaviour to handle the disturbance, which includes the set of actions to execute and the estimation of parameters, such as the estimation of the disturbance recovery time (t_r). This learning mechanisms generate knowledge taht will help to forecast the impact of the disturbance in the actual plan, to maintain the system stable and to handle the reaction to disturbance.

The other holons that sense the pheromone also increase their autonomy factors according

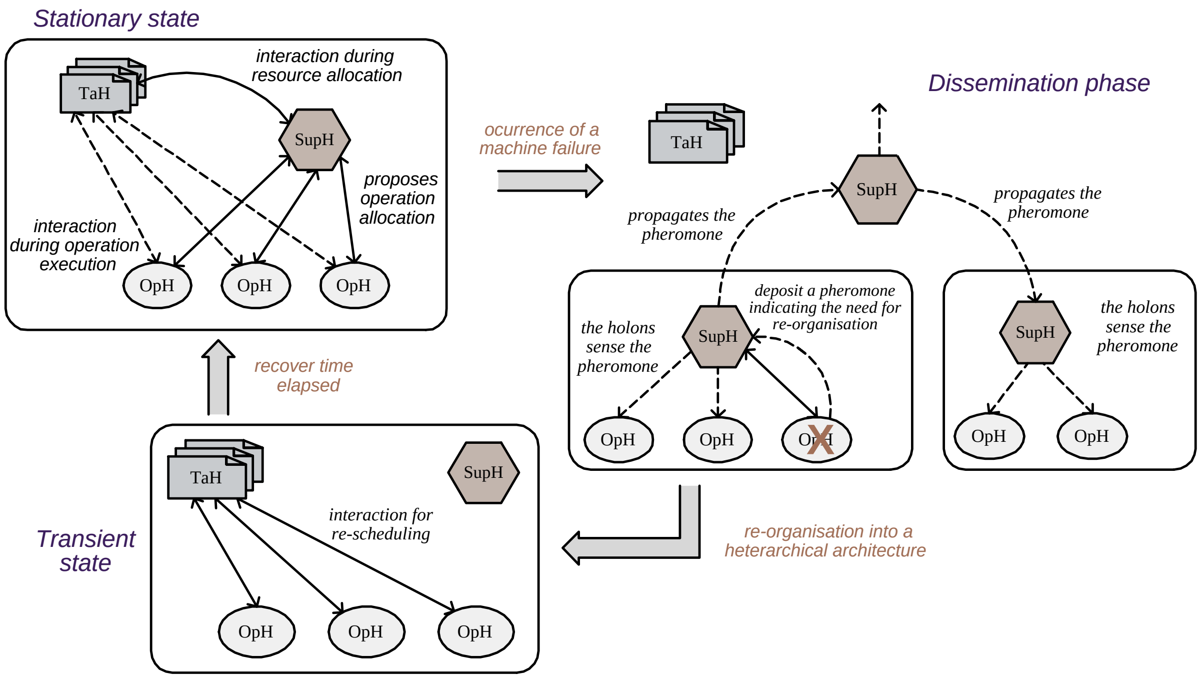

Stationary state
interaction during resource allocation
TaH
SupH
proposes operation allocation
interaction during operation execution
OpH
OpH
OpH
ocurrence of a machine failure
Dissemination phase
TaH
SupH
propagates the pheromone
propagates the pheromone
deposit a pheromone indicating the need for re-organisation
the holons sense the pheromone
SupH
OpH
OpH
OpH
SupH
the holons sense the pheromone
OpH
OpH
recover time elapsed
re-organisation into a heterarchical architecture
Transient state
TaH
SupH
interaction for re-scheduling
OpH
OpH
OpH

Figure 3.9: Dynamic Re-organisation in Reaction to a Machine Failure

to the intensity of the pheromone and their local knowledge, re-organising themselves into a heterarchical structure. In this transitory state, the task holons interact directly with the operational holons in order to achieve an alternative schedule plan. During this state, the supervisor holons can continue elaborating and proposing work orders to the operational holons, but since these now have {High} autonomy factors, in principle they will reject the proposals.

The holons remain in the transient state during the reestablishment time, τ, which typically is a short period of time estimated by the operational holon that detected the disturbance. When this time elapses, they verify if the pheromone odour is already dissipated or remains active. If the pheromone remains active, the operational holons stay in the transient phase during an additional reestablishment time, until the pheromone is dissipated.

When the pheromone is dissipated, each operational holon individually reduces again its autonomy factor and returns to the hierarchical control structure, entering again in the stationary state.

After the disturbance recovery, the operational holon ends the reinforcement of the pheromone, and the reestablishment and recovery times are adjusted and tuned using appropriated learning mechanisms. The other holons don't sense anymore the dissemination, reducing their autonomy factors, returning the system to the previous control structure. The supervisor holon returns to its coordination function, re-scheduling if necessary the work orders of the new local agendas to sinchronise the local and central schedules. The new re-schedule is sent to the operational holons, which have again {Low} autonomy factor, and accept the advised schedules.

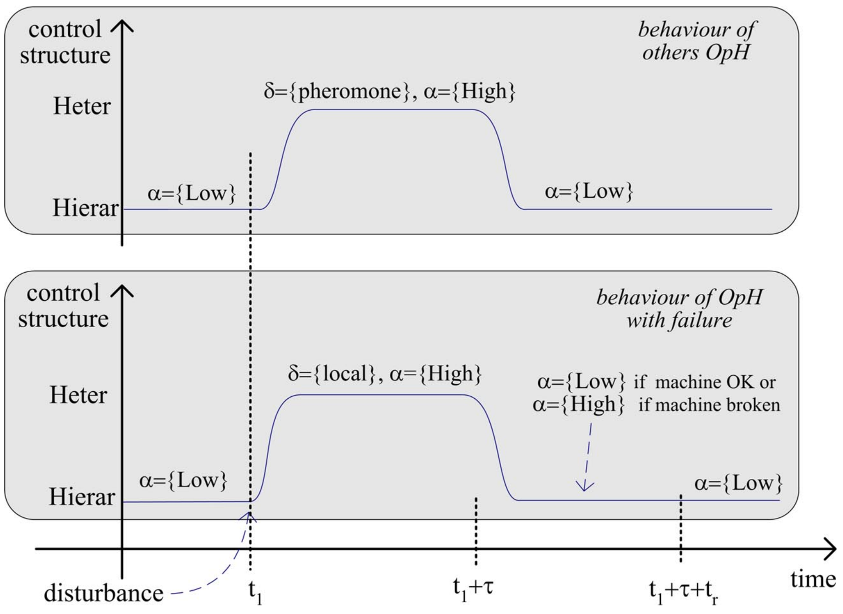


Figure 3.10: Temporal Adaptation to the Disturbance

If the operational holon where the disturbance occurred enters in the out-of-service state, it maintains the autonomy factor in {High}, decreasing the autonomy only when it returns to the operational state.

As illustrated in Figure 3.10, it is possible to verify that the δ, α and τ parameters, allow the self-organisation and dynamic behaviour in reaction to disturbances, giving operational holons an adaptive behaviour, accepting or rejecting the supervisor proposals according to their own autonomy.

3.2.5 Equilibrium in the Adaptation Mechanism

In dynamic and complex systems it is important to guarantee that the self-organisation of individual entities maintains the system in a stable and correct state. The stability concept is concerned to the condition in which a slight disturbance in a system does not produce a significant disrupting effect on that system.

In ADACOR approach, the stability in the adaptive production control structure can be affected by the number of holons present in the system, probability of failure of each individual resource or disturbance or reestablishment time.

The reestablishment time is the only parameter that it is possible to control, being necessary to adjust dynamically that value using learning mechanisms, to guarantee the stability of the system. A short reestablishment time is better in terms of stability but is worse in terms of guaranteing the completion of the disturbance recovery.

3.3 ADACOR Ontology

The ontologies are the instruments to define the vocabulary used by the holons during their interactions, to support the understanding of the message content. The actual meaning of the message content is captured in a message ontology.

ADACOR architecture defines its own manufacturing ontology, expressed in an object-oriented frame-based manner, as recommended by FIPA Ontology Service Recommendations [FIPA, 2003] to handle ontologies. This recommendation refers to the development of classes[3] describing concepts and predicates (relations), and their registration as a part of the application ontology. This allows a practical and fast way of creating an ontology with an immediate underlying implementation.

The manufacturing ontology used in ADACOR is based in the definition of a taxonomy of manufacturing components, which contributes to the formalisation and understanding of the manufacturing problem. These components are mapped in a set of objects, illustrated in the UML-like diagram of Figure 3.11, which defines the vocabulary used by distributed entities over the ADACOR platform, and indicates the concepts (objects or classes), the predicates (relation between the classes), the terms (attributes of each class), and the meaning of each term (type of each attribute).

Concepts are expressions that indicate entities with a complex structure that can be defined in terms of classes or objects. In the proposed manufacturing ontology, the main concepts are the following:

- **Product**: what the enterprise produces, including sub-products.
- **Raw-material**: the material that will be the basis of the product production.
- **Production order**: an order issued by the manufacturing planning, if possible aggregating several customer orders into a production order, to obtain volume and transport advantages or discounts.
- **Resource**: an entity that can execute a certain range of jobs, when it is available, and as long as its capacity is not exceeded. Producer, mover and transporter are specialised concepts from the resource concept, inheriting its characteristics.

[3]In some FIPA-compliant agent development platforms, such as JADE, these classes are translated to Java classes.

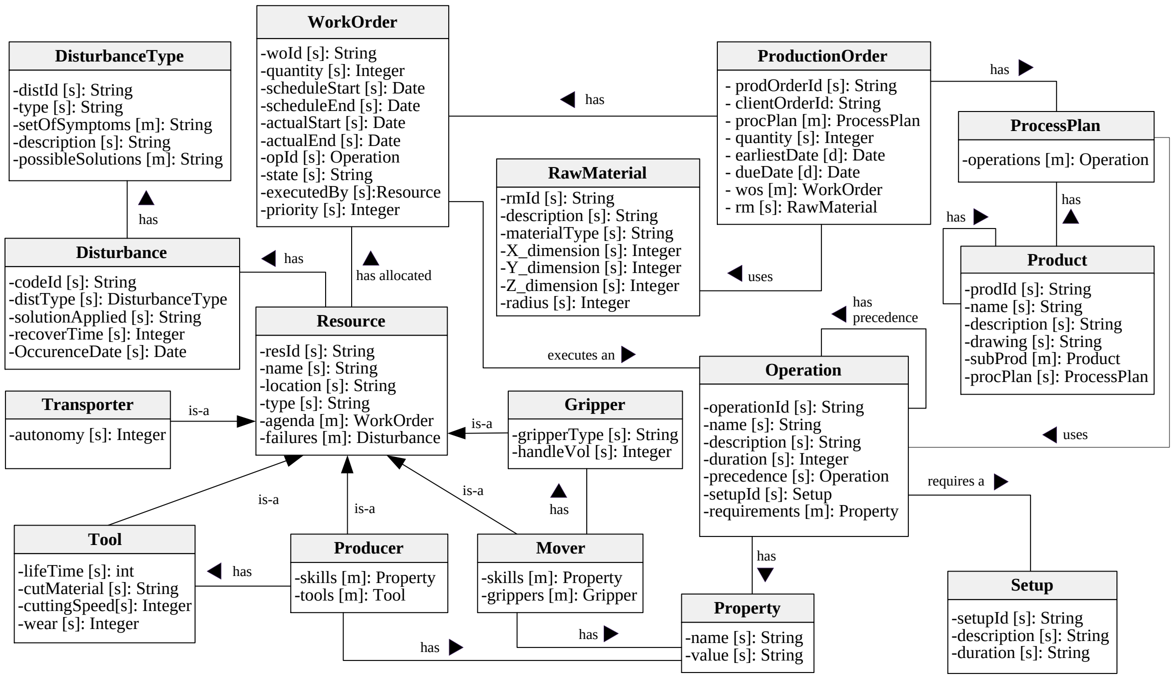


Figure 3.11: Manufacturing Ontology Developed in the ADACOR Architecture

Events or processes are actions that can be performed by some agents. The proposed manufacturing ontology comprises the following process entities:

- **Operation**: a job executed in one resource, aggregating a set of services to perform it.
- **Disturbance**: an unexpected event, such as machine failure or delay, that degrades the original production plan.

Descriptions are a special kind of concepts, used as if they form a separate class, avoiding possible confusions between concepts and their description. In the proposed ontology, there are the following descriptions:

- **Product Data Model**: description of (1) the parts of a product (the sub-products and raw material) needed to assemble the product, (2) the relationship (structure) between the parts, (3) the process plan to produce the product from its sub-products.
- **Work Order**: description of an operation including processing time, participants (e.g. name and number of machines involved in the execution of the operation), priority, scheduled dates, state and quantity.
- **Process Plan**: description of a sequence of operations. The process plan defines a list of operations and their sequence (temporal constraints like precedence of execution), in order to produce a product.

- **Agenda**: description of the actual set of operations allocated to a resource over the time (this set comprises the operations scheduled and in execution).

Predicates are expressions that allow the establishment of relationships between concepts. In the proposed manufacturing ontology, the following main predicates are defined:

- $SubproductOf(x, y)$: x is a product which is a sub-product (a component) of y.
- $Allocated(x, y, i)$: operation x is allocated to resource y during time interval i.
- $Available(x, y, t)$: resource x is available to perform operation y at time t.
- $HasTool(x, y, t)$: resource x has tool y available in its tool magazine at time t.
- $HasGripper(x, y, t)$: resource x has gripper y available at time t.
- $HasRequirement(x, y)$: execution of operation x requires the existence of the property y.
- $RequiresTool(x, y)$: tool y is used in operation x.
- $HasSkill(x, y)$: resource x has the property y.
- $HasPrecedence(x, y, z)$: operation x has precedence over operation y in process plan z.
- $OrderExecution(u, x, w, y)$: operation u is listed in the process plan w (which describes the production of product y) for production order x.
- $HasFailure(x, y, t)$: a disturbance x occurred in resource y at time t.
- $ProblemInExecution(x, y, t)$: a disturbance y causes a failure at time t and operation x will not be completed.
- $Proposal(x, y, z)$: the entity x proposes to the entity y the execution of the work order, charging the price z.

Terms are properties or attributes that are pertinent to each concept and may be pertinent to more than one concept. As an example, some properties associated to the skills of a resource and the requirements of an operation are listed in the following:

- **Axes**: a non-negative integer, that defines the number of axes of a machine.
- **Repeatability**: precision of the machine, expressed in mm.
- **Feed rate**: feed rate of a specific axis, expressed in mm/rot.

- **SpindleSpeed**: a range of non-negative integer values, i.e. [min, max], and expressed in rpm.
- **Tailstock**: the range of piece dimensions that can be processed in the machine, expressed in mm.
- **Payload**: maximum load of the robot that guarantees the repeatability, expressed in kg.
- **MaxReachability**: the work volume of the robot, expressed in mm.

3.4 Interactions Among ADACOR Holons

In distributed manufacturing environments, each holon is autonomous and has partial knowledge of the problem. The manufacturing control emerges, as a whole, from the interaction among the distributed holons, each one contributing with its local behaviour, as illustrated in the Figure 3.12.

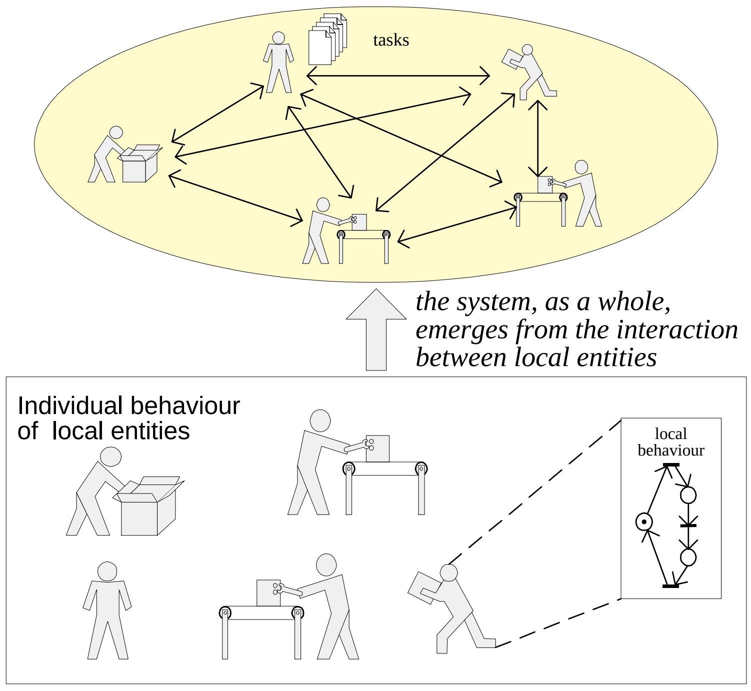

tasks
the system, as a whole, emerges from the interaction between local entities
Individual behaviour of local entities
local behaviour

Figure 3.12: Global System Emerged from the Behaviour of Local Holons

The **cooperation** concept is found in different domains such as social, politics and nature, and also in manufacturing. The cooperation process occurs when two or more entities, having partial knowledge about the global system or insufficient resources to achieve their individual goals, interact to share knowledge and skills, both expecting benefits. During the cooperation process, some conflicts can occur as result of incomplete local knowledge, different goals, priorities and evaluation criteria. [Ferber, 1999] defines cooperation as the conjunction of collaboration, coordination of actions and resolution of conflicts.

The **collaboration** process occurs when an entity helps another entity to perform a particular job, without expecting to have individual benefits.

The **coordination** appears with the need to synchronise a sequence of actions to be performed by autonomous entities, for example, to handle the interdependencies between tasks. The coordination does not imply the cooperation: an entity can coordinate its actions with an opponent entity to maximise its advantage against it.

The **negotiation** process consists in the effort made by two or more entities to achieve an agreement benefiting themselves, and occurs when it is necessary to solve a conflict, for example during the allocation of tasks to manufacturing resources.

At the shop floor level, the manufacturing components must cooperate to produce a product, and must co-ordinate their actions according to the product dependencies and the production plan. The complexity of the interaction process, in the manufacturing domain, is mainly dependent of the following factors:

- **Goals**: each entity has specific goals, which may be not compatible between themselves and may lead to conflicts.
- **Decision-making process**: an entity may have difficulties to reach the best decision within a required response time, due to the partial scope of its knowledge.
- **Heterogeneity**: the distributed entities are implemented using different architectures, technologies, communication languages, and ontologies.
- **Manufacturing type**: depending of the manufacturing type, such as make-to-stock, make-to-order and engineer-to-order, different actors are involved and different interaction complexities are presented.
- **Population**: the population of entities has a strong impact in the of the level of interactions, implying the increase of the communication volume.

In the ADACOR architecture, three different classes of interactions between ADACOR holons are identified: introduction of new orders, plan execution, and disturbance handling.

In this work, the interaction between distributed holons will be modelled using AUML interaction diagrams [Odell et al., 2000], which is an extension of the UML's sequence diagrams for the multi-agent systems.

3.4.1 Introduction of New Orders

When new production orders arrive to the shop floor, the ADACOR holons must interact to elaborate a plan for their execution.

Product and Process Information

Each product holon represents one available product, that can be produced by the factory plant and contains all the knowledge to produce the product, namely the product structure and the process plan.

When a product order is issued, the product holon triggers the physical execution of the product by launching a task holon that will be responsible to manage the physical production of the product. In case of multiple orders for the same product, the product holon handles the execution of several production orders, launching and handling the interaction with several task holons.

The coordination of the product and task holons activities requires the synchronisation of the individual product and task holon models, as illustrated in Figure 3.13, at two different moments: when the task holon is launched and when the task holon finishes the execution of the production order.

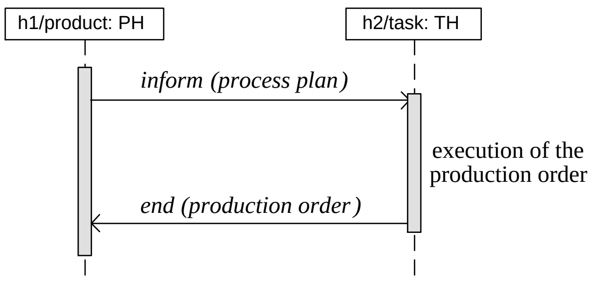


Figure 3.13: Interaction between Product Holon and Task Holon

In the first moment, the product holon launches the task holon, passing the product and the process information.

At the end of the production order execution, the task holon passes to the product holon the relevant information about the execution of the product, in order to support the learning mechanisms in the product holon.

Global Coordination

The presence of the coordination levels, i.e. the existence of supervisor holons guarantees stable scenarios when the system is running without the occurrence of unexpected disturbances.

The task holons interact with supervisor holons to announce new work orders. The supervisor holons elaborate, periodically, optimised schedules for the coordinated resources, decomposes them into individual work orders, and proposes them to the operational holons, as illustrated in Figure 3.14. The operational holon takes this proposed schedule as an advice, having enough autonomy to accept or reject, based in its local knowledge and actual behaviour.

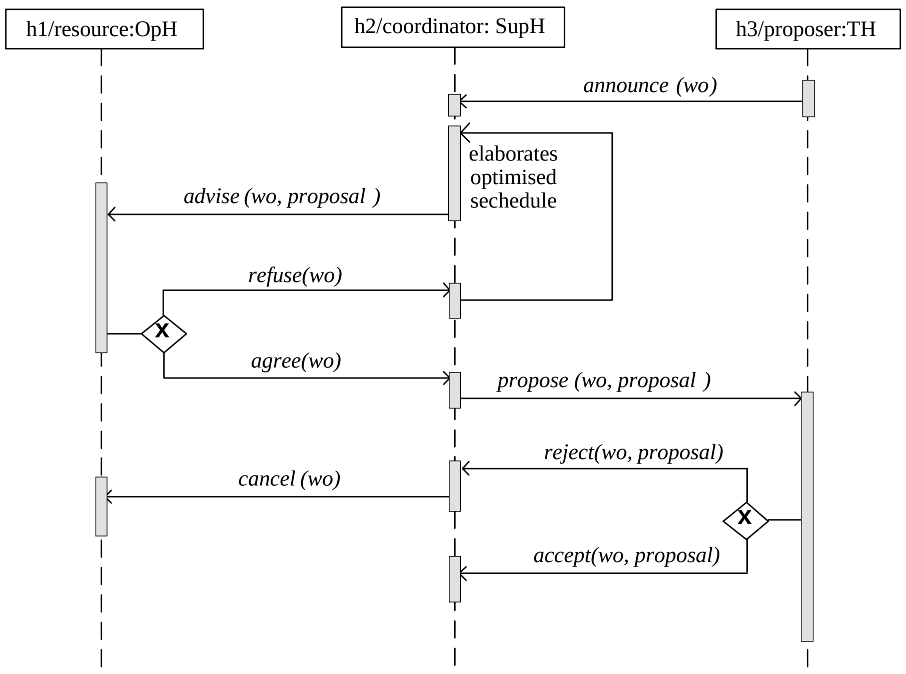


Figure 3.14: Global Coordination Interaction Diagram

If an operational holon rejects one or more proposed work orders, the supervisor holon reschedules the production plan, trying to find alternatives. If the proposed work orders are accepted by the operational holons, the supervisor holon proposes a calendar to the task holon.

The operational holon receives a cancellation of the previous allocation, if the task holon rejects the proposal elaborated by the supervisor holon.

3.4.2 Plan Execution

After the allocation of the work orders to the available plant resources, it is necessary to synchronise the activities leading to the physical execution of the production order. For this purpose the task and operational holons must interact.

Work Order Execution

The interaction diagram for the execution of a work order, illustrated in Figure 3.15, depicts the sequence of activities performed by the several holons in order to execute a work order from the resource allocation plan.

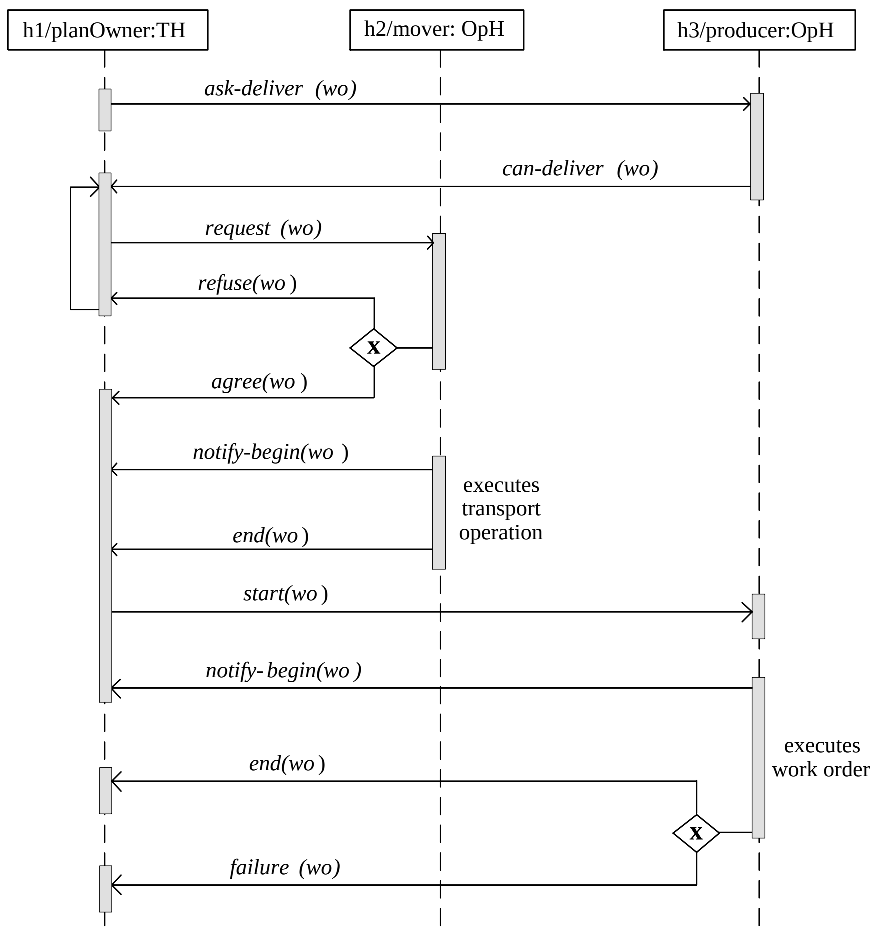


Figure 3.15: Sequence Diagram for the Execution of a Work Order

As after the allocation of one work order to a resource, it is not guaranteed that the operational holon can start immediately the execution of the work order, due to lack of material, delay in a precedent work order, or other constraint. The execution starts when the task holon asks the operational holon if it can deliver the part. When the operational holon is ready to receive the part, the task holon requests the transportation of the part to the machine buffer.

After receiving the notification that the part is in the machine buffer, the task holon uses the *start(work order)* message to indicate to the operational holon that it can start the execution of the work order.

When the operational holon starts the execution of the work order, it notifies the task holon and the supervisor holons, if exists. After finishing the execution of the work order, the operational holon notifies again both the task and supervisor holon.

In this interaction diagram, the primitive REQUEST refers to some quick response work orders, like certain transport operations, that do not need to be scheduled. The STARTS primitive refers to already allocated work orders.

Monitoring

In a manufacturing control system, the acquisition of information by monitoring the production process is required to coordinate activities and to determine if the plans are being executed as elaborated.

In the low level, the monitoring is related to the acquisition of information from the physical process. In this case, each operational holon accesses its controlled resource to know the status of the machine or the state of the sensors (for example, to know if the machine door is open or closed or if the water cooling system is activated or not).

In a higher control level, the need for monitoring comes from the need of holons to acquire new knowledge in order to expand their partial view of the system. In this case, each autonomous ADACOR holon interacts with other holons, requesting the desired information. The query of information between distributed holons can be performed in passive or active forms.

In the passive monitoring, the initiative belongs to the holon that needs some information, and the target holon could be a supervisor or an operational holon, depending of the organisational structure.

In case of an operational holon, as illustrated in Figure 3.16, it gets the required information from the local knowledge base and sends it back to the holon that requested the information. The operational holon may need to access the physical resource that it represents to obtain the requested information.

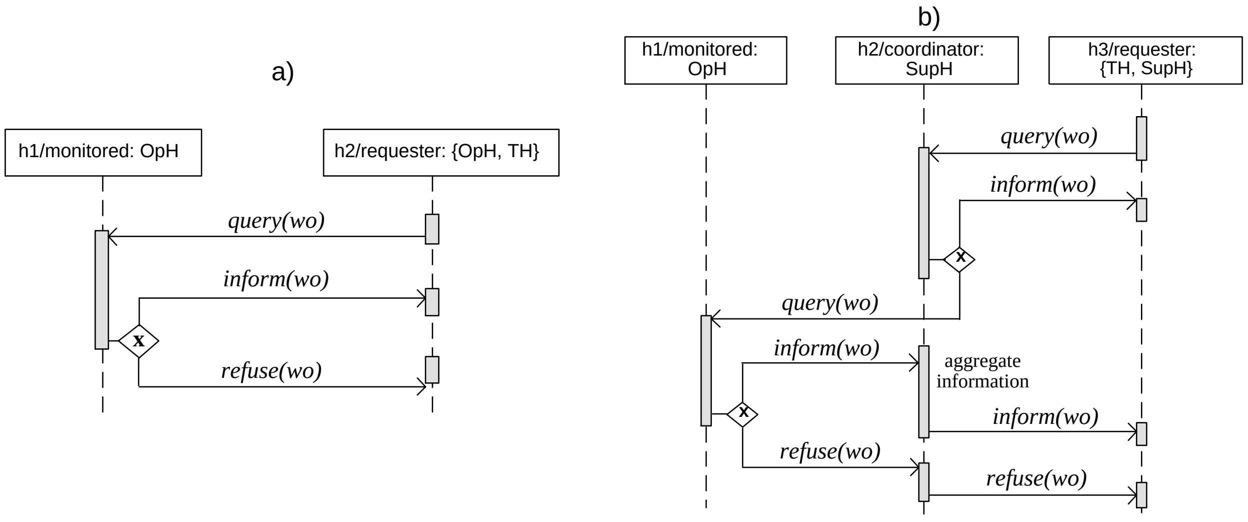


Figure 3.16: Passive Monitoring: a) simple and b) with coordination

In case of a query of information to the supervisor holon, this evaluates if it has enough knowledge to answer to the request. If not, the supervisor holon requests information to the lower-level operational holons, compiles the answers and sends back the requested information.

In active monitoring, the active notification interaction protocol, illustrated in Figure 3.17, involves the subscription of services for the future notification in case of occurrence of some events. Examples of services that can be subscribed for notification are the work order finished and the work order delay.

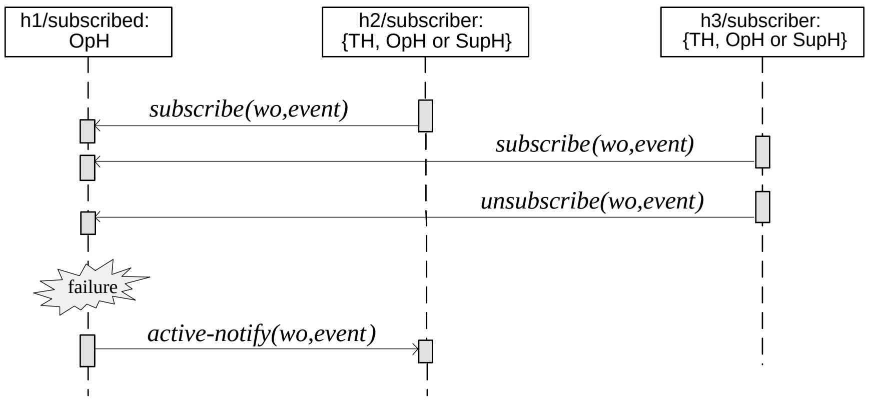


Figure 3.17: Active Monitoring

In the active notification process, the subscriber holon subscribes the service related to the event that wants to monitor in an active form, sending a *subscribe(wo,event)* message to the target holon. When an event occurs, the subscribed holon checks the subscription list of events and sends the notification about the occurrence of the event to all holons that subscribed the service. After the notification, the holon that subscribed the service decides how to handle the occurrence of the disturbance.

Physical Synchronisation

This type of interaction addresses the interaction between operational holons, for example to request the execution of maintenance or setup operations. In some manufacturing situations it is also required the physical coordination between different resources, as for instance the loading or unloading of a machine-tool by a handling robot, as illustrated in Figure 3.18, or the transfer of a pallet between an AGV and a machine buffer.

In this cooperation sub-process, the operational holons (mover and producer, specialised types of the operational holon) cooperate in order to make possible the synchronisation between the resources they represent, exchanging messages to synchronise their activities, as illustrated in Figure 3.19.

The holon that wants to request the execution of the service sends the *request(wo)* message

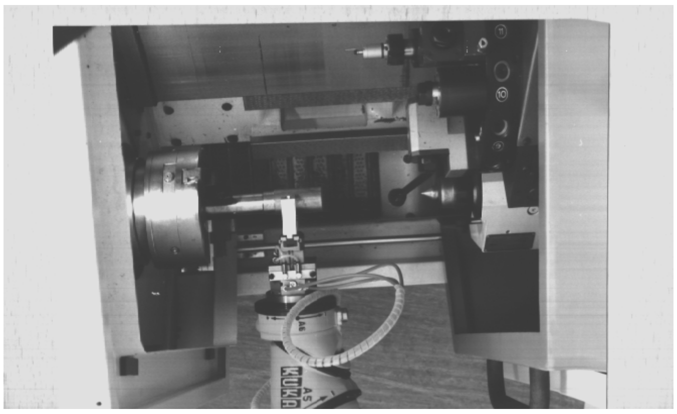


Figure 3.18: Synchronisation between Physical Resources

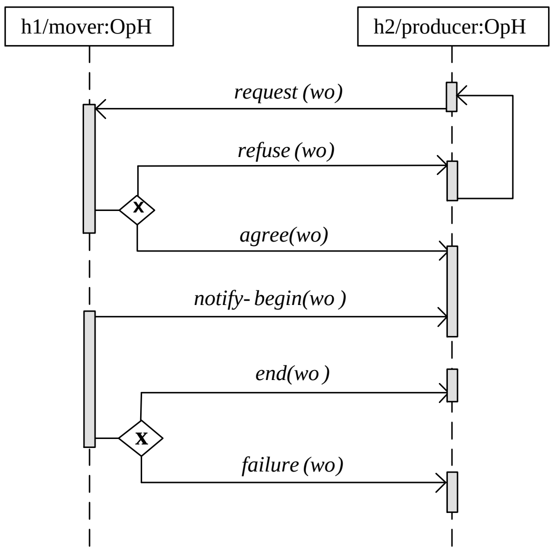


Figure 3.19: Sequence Diagram for the Synchronisation between Physical Resources

and waits for the agreement or refusal by the other holon. In case of agreement, the mover operational holon notifies the producer operational holon of the effective start of the service.

After the end of auxiliary service, the mover operational holon notifies the producer operational holon of the end of the service, and returns to the idle state. In case of failure during the execution of the auxiliary service, the producer operational holon receives a notification of failure.

3.4.3 Disturbance Handling

The occurrence of disturbances originates the re-organisation of the control structure, supported mainly by the propagation mechanism and the self-organisation of each ADACOR holon.

Propagation of the Re-organisation

The need for the re-organisation is propagated by the holon that detected the disturbance to the other holons, sending a *propagate(pheromone)* message to the neighbour supervisor holon.

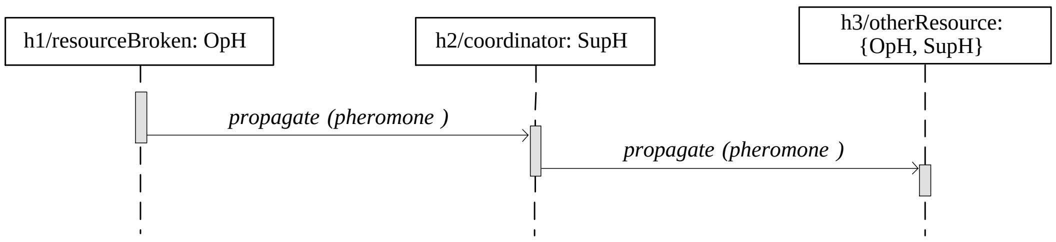


Figure 3.20: Interaction Diagram for the Failure Propagation

The supervisor holon propagates the pheromone received to the entities placed in its coordination domain and to the entities placed in a higher hierarchical level, according to the interaction diagram of Figure 3.20, allowing the evolution of the other operational holons into a heterarchical structure.

Dynamic Distributed Scheduling

In scenarios where the control application runs without the presence of coordination levels, the scheduling is distributed by the holons in the system, requiring high degree of interaction between the entities that have tasks to be executed and entities that have skills and resources to execute them. The dynamic distributed scheduling process occurs during the re-allocation of work orders after the occurrence of disturbances.

In these scenarios, the task holon receives a *failure(work order)* message indicating that the resource had a failure that destroyed the part, a *take-back(work order)* message indicating that the resource can not execute the work order or a *delay(work order, due date)* message indicating that the work order is delayed, and interacts with the operational holons to re-allocate the work orders.

As illustrated in Figure 3.21, each operational holon is responsible for its own schedule built dynamically from its local knowledge. The global scheduling is achieved through the interaction between the operational holons and the task holons, using this task allocation interaction mechanism.

ADACOR uses a distributed scheduling mechanism, based on a multi-round Contract Net Protocol (CNP), and extending the original functionalities of CNP schema with the following main features, useful in distributed manufacturing systems:

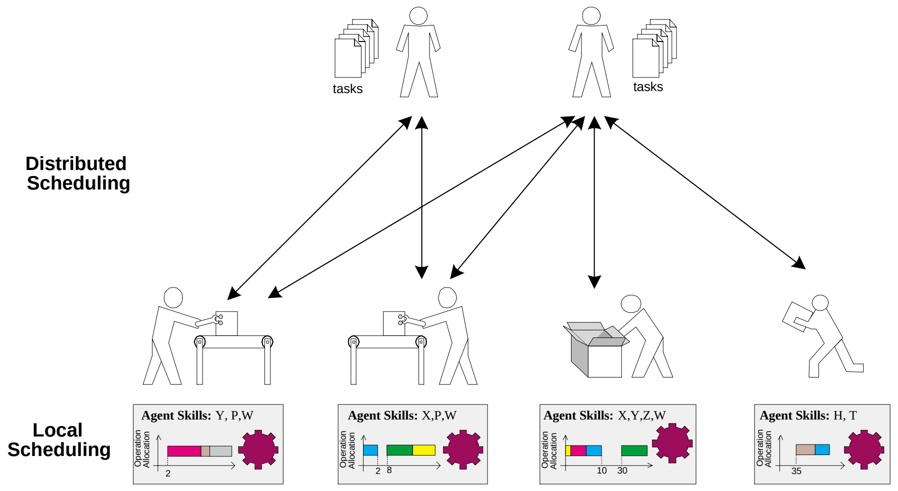


Figure 3.21: Distributed Scheduling

- Multiple iterations, applying the contract net schema several times, in order to support iterative optimisation.
- Capability to contract a partial quantity and to start another iterative negotiation process with the remaining quantity, in case of allocation of partial quantities.
- Introduction of penalty conditions in the contract, in case of delay or cancellation of operations.
- Support of the hierarchical control and the presence of third-party entities.

In the distributed scheduling interaction, illustrated in Figure 3.22, the task holon allocates the work orders that belongs to its production order to operational holons, by analysing the proposals sent by the operational holons and making a decision about to whom allocate each work order. Each proposal contains the proposed price, quantity, and information related to the resource.

In this negotiation process, the task holon tries to allocate the work orders at cheapest price, fulfilling the required due dates, and the operational holons try to get as many work orders as possible, at the highest price.

The task holon can make one of three different decisions as a result of the evaluation procedure: (1) accept one proposal and reject the other proposals, (2) award a partial quantity and start another iterative negotiation in order to allocate the remaining quantity, and (3) reject all proposal and start another iterative negotiation.

The new iterative negotiation requires the specification of new bid parameters. The start date, due date and announcement specifications of each work order must be dynamically changed

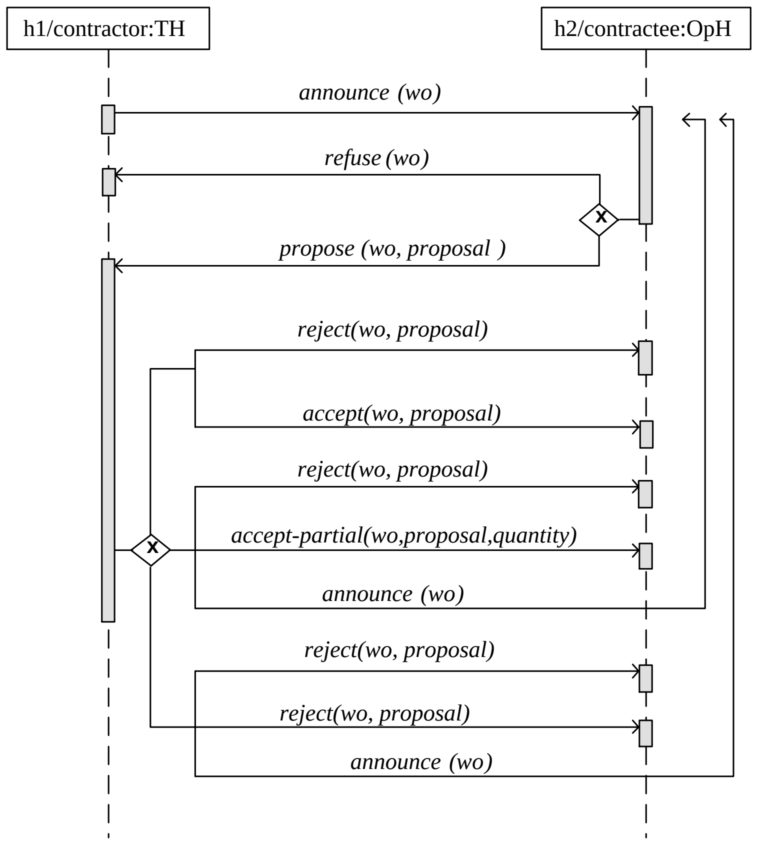


Figure 3.22: Dynamic Scheduling Interaction Diagram with Multiple Iterations

in order to allow the optimisation of the global allocation using multiple iterations.

When awarding a proposal for the execution of a work order, the task holon and the operational holon agree that the value paid by the task holon i to the operational holon k for the execution of the work order j will be φ (equals to the price p_{jik}), and in case of delay in the execution or cancellation, the operational holon will be penalised by the value ξ.

Aiming to reach better results in this scheduling mechanism, it is necessary that all the holons present in the interaction learn from previous experiences, adapting their decision parameters according to the behaviour of the other holons present in the system. For this purpose, after the execution of each work order, information related to its execution is stored in the task holon database. Also in case of failure or cancellation, it is necessary to store the information related to the failure.

3.5 Architecture of an ADACOR Holon

The conceptual model of a generic ADACOR holon, illustrated in the Figure 3.23, comprises the Logical Control Device (LCD) and the physical resource capable to perform the manufacturing tasks, if it exists. The LCD device acts as an agent and is the heart of the holon, being responsible for regulating all activities related to the holon.

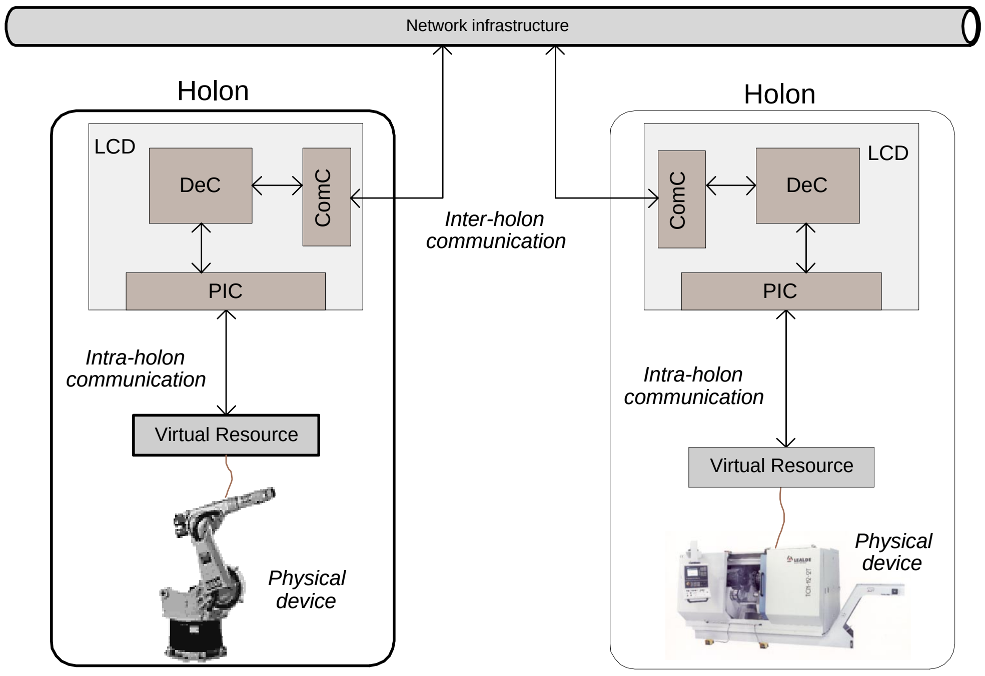


Figure 3.23: Conceptual Model for an ADACOR Holon

The internal architecture for a generic LCD inside an ADACOR holon, comprises three main components: decision (DeC), communication (ComC) and physical interface (PIC) components [Leitão and Restivo, 2002a].

The communication component is responsible for the inter-holon interaction, making transparent the data exchanged in order to support the cooperation with other holons in the system. The decision component is responsible mainly for the manufacturing control functions, regulating the holon's behaviour and activities. The physical interface component is responsible for the intra-holon interaction, providing mechanisms that make transparent the access to the manufacturing resources, supporting the monitoring of resource data and executing commands in the resource.

Each one of the components present in the internal architecture of the LCD device will be described in detail in the following sections.

3.5.1 Decision Component

The decision component regulates the holon behaviour, namely performing the manufacturing control functions, such as the process planning, scheduling, and plan execution (dispatching, monitoring and reaction to disturbances), and adapting to emergence (group formation, dynamic re-organisation, etc.).

Information and Knowledge

The decision-making and the ability to reason, of an ADACOR holon, is based in its local information and knowledge. The information and knowledge concepts are often confused.

Data represents facts or statements without relation to other things, having no significance beyond its existence. Information is data that has been given meaning through relational connections, providing answers to *who*, *what*, *where* and *when* questions. Knowledge is the appropriate application of data and information, which meaning is intended to be useful, answering to *how* questions. [Ackoff, 1989] defines also two additional related concepts: understanding and wisdom. Understanding is the cognitive process to answer to *why* questions and wisdom is the evaluation of that understanding.

According to [Ferber, 1999] the concept of knowledge *"can be defined as all the information needed by a human being (or a machine) organised in a such way that he, she or it can carry out a task considered as being complex"*. As an example, we need knowledge to repair a damage machine or solving a specific problem.

In decision systems the formal representation of the knowledge is a crucial factor, affecting the development, efficiency, speed and maintenance of the system. In the research community, it is recognised that there is no single formalism to represent properly the knowledge for all problems. However it is possible to find two main classes of knowledge representation [Torsun, 1995]:

- declarative or descriptive knowledge, where the information is stored explicitly, such as tables and graphs.

- procedural or imperative knowledge, which is an implicit type of knowledge manipulated in the execution of the process. An example is a computer program.

In this way, several different formalisms are commonly used, such as production systems, semantic networks, frames and conceptual graphs [Giarratano and Riley, 1998, Torsun, 1995].

Decision-Making Model

As each individual holon is placed in a society of holons, according to an organisational structure, the decisions are result from the interactions with other holons. Each holon possesses a decision-making mechanism which is independent of the level of hierarchy where the holon is placed, if any.

The decision-making model, illustrated in Figure 3.24, comprises basically the following steps: sensing, deciding, acting and learning.

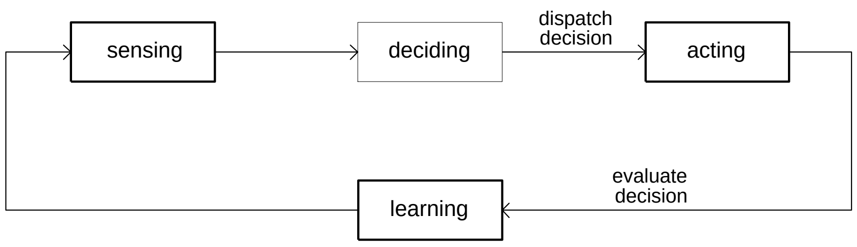


Figure 3.24: Decision-Making Process Model

The holon is continuously and cyclicly available to perform a decision by aggregating the available information, to support the decision-making, stored in the local database. This information is acquired by sensing the environment using sensors and through the arrival of messages from other holons.

When the available information is not enough, the holon may decide to start a co-operation process with other holons or with the manufacturing resource, trying to find out the necessary information. Multiple iterations in the cooperation process may be required before a final decision is made.

The decision related to a manufacturing control function is made according to the available knowledge and to appropriate decision-making techniques. In the literature, several different approaches have been proposed to implement the decision-making components, mainly derived from the Artificial Intelligence (AI) area, such as expert systems, neural networks, genetic algorithms or fuzzy logic [Pham and Pham, 2001, Torsun, 1995].

Expert systems are computer programs designed to simulate the problem-solving behaviour of human experts within very narrow domains. An expert system usually comprises two main elements, a knowledge base and an inference mechanism [Pham and Pham, 2001]. The knowledge base contains domain knowledge which may be expressed as a set of rules, and factual statements. The inference mechanism is the part of an expert system which manipulates the stored knowledge to produce solutions to problems.

A neural network mimics the brain's learning and decision-making process, usually assuming that computation is distributed over several simple units called neurons that are interconnected and operate in parallel [Pham and Pham, 2001]. Neural networks learn through training and experience. The training phase intends to calibrate the behaviour of the neural network and to improve its response.

Genetic algorithms are inspired by Darwin's theory of evolution, where problems are solved by an evolutionary process which leads to the best solution [Obitko, 2003]. The algorithm begins with a set of solutions, called population. Solutions from one population are taken and used to form a new population, implying that the new population should have better solutions than the previous one. The new solutions are selected according to their fitness - the more suitable they are, more are the chances they have to reproduce. This process is repeated until some condition (for example number of populations or relative improvement) is satisfied.

Fuzzy logic [Zadeh, 1965] allows to support uncertain or vague knowledge, by describing the human reasoning as qualitative or fuzzy statements, using whole interval between 0 (false) and 1 (true) to represent intermediate values. In fuzzy logic, the precise value of a variable is given by a linguistic description, translated by a fuzzy set, and inferencing is carried out based on this representation [Pham and Pham, 2001].

After the decision is taken, the selected actions are dispatched and executed. These actions can be in the form of commands for actuators, messages for other holons, or execution of procedures.

The increase of the holon's knowledge and the improvement of its behaviour performance, supporting the adaptation of the holon to the environment emergency and evolution, requires the implementation of learning capabilities. The results from the executed actions may be evaluated and according to the results, new knowledge is generated, improving the holon efficiency in the decision-making process in future situations.

The learning capability is dependent of the decision mechanisms and of the learning algorithms. As an example, in case of neural networks, the learning is associated to the adjustment of the nodes coefficients, but in case of expert systems, the learning is associated to the addition of new facts or to the generation of new rules. The elaboration of new rules is more complex than the simple acquisition of new factual knowledge, and requires a special attention to verify dynamically the possible contradiction of the new knowledge rules in relation to the initial behaviour knowledge.

3.5.2 Communication Component

The cooperation and coordination processes used in ADACOR architecture to support the manufacturing control functions use as infrastructure the mechanisms provided by the communication component. The *communication* component deals with the inter-holon interaction to support the sharing of local knowledge by distributed holons.

Definition of Communication Mechanisms

As the interaction between distributed entities requires the capability to communicate with other holons, the communication mechanisms that support the interaction assume a crucial factor. The interaction can be classified into two different forms [Sousa et al., 1999]:

- **Direct interaction**, where the entities exchange information directly between each other, and they must individually understand some communication mechanism.
- **Indirect interaction**, where entities have no direct communication with each other, but can exchange information through the use of their environment. An example can be found in the ant colonies that communicate indirectly by means of pheromones, which are chemical substances that an ant can drop in the environment and that other ants can sense when they pass by.

According to [Ferber, 1999] the communication infrastructure can be defined through the definition of the type of connection, the medium and the intention to communicate.

The type of connection during the communication can be point-to-point, in which there is a direct connection between the sender and the receiver entities, multicast, in which the sender announces the message to a group of entities, and the broadcast, in which the sender announces the message to all entities belonging to the community.

The medium used to communicate can be direct routing and public notice routing (when there is a common place, visible to all agents, where the messages are stored, called noticeboard or blackboard).

The intention to communicate can be intentional communication (occurs when an entity wants to obtain something from other entity and takes the initiative to ask it to receiver) and the incidental communication (occurs without the sender entity taking any active part in them).

The communication component in ADACOR architecture provides mechanisms to support the interaction between the holons through the use of messages over a communication infrastructure. The interaction between the holons is asynchronous, i.e. a holon that sends a message continues its execution without the need to wait for the response, the type of connection can be point-to-point, broadcast or multicast, and the medium used to communicate is the direct routing.

In order to make easier the management of the conversations, ADACOR holons use the previously described interaction protocols that defines the sequence of messages exchanged between the holons.

Mechanisms for Communication Standardisation

In order to standardise the communication process, to make transparent the data exchange and to support the inter-operability of the holons or between different holonic control platforms, the communication process is divided into three main phases, Figure 3.25: content standardisation, format of the content and standardisation of the message.

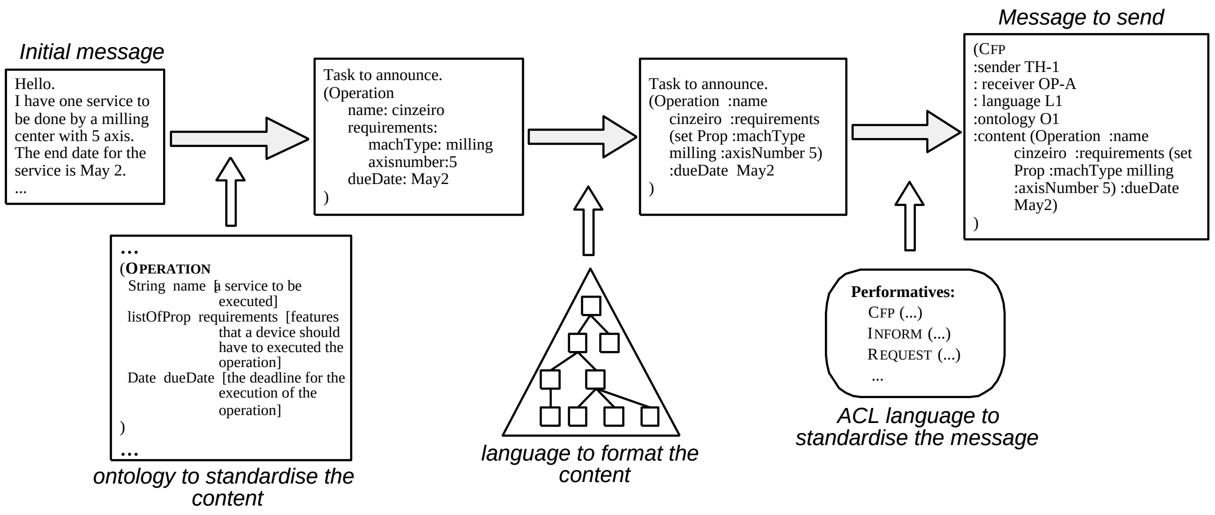

Initial message
Hello. I have one service to be done by a milling center with 5 axis. The end date for the service is May 2. ...
Task to announce. (Operation name: cinzeiro requirements: machType: milling axisnumber:5 dueDate: May2)
Task to announce. (Operation :name cinzeiro :requirements (set Prop :machType milling :axisNumber 5) :dueDate May2)
Message to send
(CFP :sender TH-1 : receiver OP-A : language L1 :ontology O1 :content (Operation :name cinzeiro :requirements (set Prop :machType milling :axisNumber 5) :dueDate May2))
... (OPERATION String name [a service to be executed] listOfProp requirements [features that a device should have to executed the operation] Date dueDate [the deadline for the execution of the operation]) ...
ontology to standardise the content
language to format the content
Performatives: CFP (...) INFORM (...) REQUEST (...) ...
ACL language to standardise the message

Figure 3.25: Application of Communication Mechanisms

In distributed environments, characterised by autonomous and cooperative entities, the inter-operability and the contents interpretation during the cooperation is an important aspect. When two or more holons intend to communicate, it is necessary for them to agree on the vocabulary used during the communication act. Thus, the first phase of the communication process is concerned to the standardisation of the content of the message, by using an appropriate ontology.

The second phase of the mechanism is concerned to the formation of the content using a semantic language. This language does not define a meaning for the symbols, which is the responsibility of the domain language representation and the relative ontology. The SL0 language, that is the minimal subset of the FIPA SL content language, is an example of a semantic language for the formation of the message content.

The third and last phase of the communication mechanism is concerned to the standardisation of the message using a proper agent communication language. Examples of agent communication languages are the KQML and FIPA-ACL. In spite of being more recent, FIPA-ACL seems to be the more promising ACL language in multi-agent domain, since it has been proposed as part of FIPA specifications by the FIPA organisation [FIPA, 2003], that tends to be the standard in the agent domain.

3.5.3 Physical Interface Component

The *physical interface component* provides the mechanisms to integrate the manufacturing resources, such as robots and machine-tools, i.e. the connection between the software part of the holon and the physical manufacturing resource. As the local resource controllers have mostly closed architectures it is necessary to develop wrappers to hide the details of each resource controller and to supply primitives that represent the functionality of the physical manufacturing resource [Barata et al., 2001]. Thus, the integration of manufacturing resources into holonic and agent-based applications assumes a crucial aspect, requiring mechanisms that make transparent and independent the control application from the details of local resource controllers. The PIC component aims to fulfil this objective, by comprising mechanisms for the interaction with the physical devices that makes transparent the access to manufacturing resources from the holonic control application.

Manufacturing Resource Integration Technologies

Several approaches for the integration of physical resources within holonic and agent-based systems have been proposed and implemented. FIPA organisation aims to produce standards for the interoperation of heterogeneous software agents [FIPA, 2003]. At, the moment, there aren't specifications to support the integration of physical resources, in spite of the effort to introduce new specifications that support manufacturing requirements, through the FIPA Product Design and Manufacturing working group.

MMS, which defines the application layer of the ancient MAP (Manufacturing Automation Protocol) protocol, provides a platform capable to interconnect various industrial devices supplied by different suppliers. MMS brought together many IT and manufacturing specialists to define a common framework for developing communication support between industrial computerised equipment, under the ISO 9506 international standard [ISO/IEC9506-1, 1992].

The basic concepts of the MMS protocol are a client-server mechanism and the VMD (Virtual Manufacturing Device) model. The VMD, associated with every real manufacturing device, is an abstract model of the server application, which maps the functionalities of the real device, and offers all services concerning itself and its related abstractions, mainly: *domains*, *variables*, *program invocations* and *events*. This set of objects and generic services can be applied to a large set of manufacturing devices, such as robots and numerical control machines [Nussbaumer, 1991].

However, the technology vendors do not closely follow the MMS standard, since certain functionalities have different implementations depending of the machine vendor and the underlying network. This missed adhesion by the vendors to the standardisation associated to the high price of this technology retracted the expansion of the MMS technology in the market.

Some research teams introduced the idea to use the MMS concepts combined with a distributed object platform technology to integrate the manufacturing resources. [Ariza et al., 2001] and [Boissier et al., 2001] introduce approaches that uses the MMS concepts over the CORBA (Common Object Request Broker Architecture) distributed object technology, with successful results. The real-time constraints in the industrial manufacturing world requires real-time response, being the CORBA technology and the classical TCP/IP (Transmission Control Protocol/Internet Protocol) not adequate. The ReTINA model has been used within the *Jonathan* distributed environment [Dumant et al., 1998], in order to support applications subject to real-time functioning [Boissier et al., 1998].

Another available technology is the OPC (OLE for Process Control Automation), which is based on Microsoft's OLE/DCOM technology. It allows software in the form of software components to interoperate regardless of where they are located [Barata et al., 2001]. The OPC servers offer an automation interface, which allows to design PC-based clients that import real time automation data using standard Windows applications. The Windows proprietary scope remains an important limitation of this approach for the heterogeneous environments; however, some available tools, such as Bridge2Java[4] and JIntegra[5], allow to overcome this problem.

IEC 61499 is an approach for the easy and fast integration of large re-configurable systems, defining a way to model the control and execution of algorithms in distributed control systems, through encapsulated and reusable software modules [Balasubramanian et al., 2001, Christensen, 2002]. However, at moment the implementation scope of this approach is limited, because the low level programmable controllers do not support yet completely the IEC 61499 specifications. As it is planned to replace the existing IEC 1131-3 programming languages in the future, this approach should be taken in consideration for the integration of processes and devices by the direct communication between the entities.

Virtual Resource Approach

From the preceding, it is clear that is missing an approach that supports cheaper, independent and transparent interfaces for physical manufacturing resources, allowing to integrate them into holonic control applications. The use of light MMS concepts combined with distributed object paradigms seems a suitable approach to make transparent the resource integration from the agent-based or holonic control system.

[4]http://www.alphaworks.ibm.com/java

[5]http://j-integra.intrinsyc.com/

The ADACOR approach to transparent resource integration within holons is taking advantage of the OO-MMS mechanism for communication between any client and a VMD server [Leitão et al., 2003a]. The resource integration scheme, displayed in Figure 3.26, comprises the virtual resource and the client-server model.

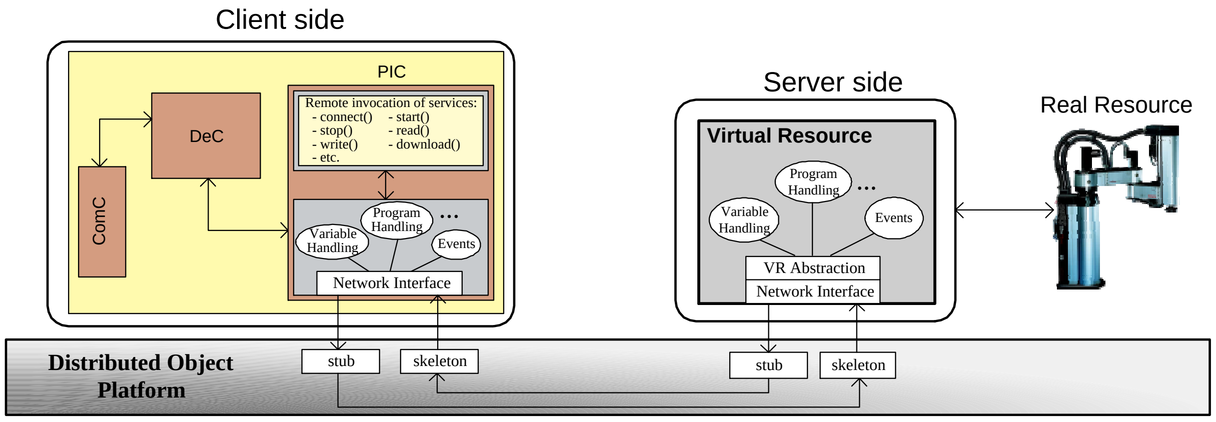


Figure 3.26: Invocation of Remote Services using the Virtual Resource concept

The server part in the proposed mechanism is the virtual resource, inspired in the VMD concept from the MMS protocol [ISO/IEC9506-1, 1992]. It acts as an abstract machine that represents the functionalities of the real manufacturing device and its local controller, and supplies primitives to be invoked remotely by the client part. The virtual resource components can be re-used for additional and new applications, since the manufacturing resources are independent from the control application. The virtual resource is developed for each physical device according to the specifications of the machine vendors and comprises a set of objects that maps the services of manufacturing resources.

The use of MMS specifications in the definition of services is important in order to standardise the approach, but due to the complexity of these specifications, a sub-group of services were defined, as closest as possible to the MMS specifications, in order to make things easier and lighter. These services are grouped in re-use libraries, such as the *VR Support*, *Variable Handling*, *Program Handling* and *Events*, as represented in Figure 3.27, which shows some services that provide the interaction with the physical manufacturing devices.

The input parameters and return values of each available service are always the same, making transparent the development of the holonic manufacturing applications from the particular details of each resource, improving the ability to support the heterogeneity.

The interaction between the physical manufacturing resource and the virtual resource is also dependent of the communication platform, such as serial link, fieldbus networks and TCP/IP protocol or different connectivity applications developed under OPC, ActiveX components or

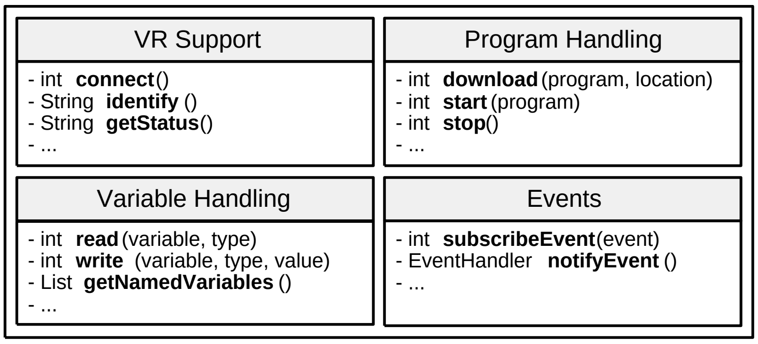

VR Support
- int connect()
- String identify ()
- String getStatus()
- ...
Program Handling
- int download (program, location)
- int start (program)
- int stop()
- ...
Variable Handling
- int read (variable, type)
- int write (variable, type, value)
- List getNamedVariables ()
- ...
Events
- int subscribeEvent(event)
- EventHandler notifyEvent ()
- ...

Figure 3.27: Re-use Libraries of Services Provided by the Virtual Resource

other technology.

The second main concept in the proposed mechanism is the client-server model. The PIC component acts as a client part accessing to the real manufacturing resource by invoking remotely the primitives that represent services in physical resource and are supplied by the virtual resource.

The industrial manufacturing environments are characterised by its heterogeneity, with the distributed processing resources, i.e. computers, industrial controllers and automation devices, running in distinct platforms, such as Windows, Linux and AS400. This heterogeneity requires the use of distributed object platforms to support the interoperability between the clients (PIC component of operational holons) and virtual resource components. The available technologies to support the distributed object platform are mainly the CORBA, Microsoft's .NET[6] and RMI (Remote Method Invocation) [Barata et al., 2001].

CORBA is based essentially in the Object Request Broker (ORB) concept[7], which allows a local client to invoke methods on a remote platform as if they were local. In order to mask remoteness and networking details, the middleware platform installs end points (stub and skeleton) on the client side and on the server side. The behaviour is very close to the Remote Procedure Calls (RPC) mechanism; however, a RPC is offered by a dedicated server, while ORB methods are attached to a client and a server can handle many clients.

Like CORBA, the Java Remote Method Invocation (JRMI) is conceptually similar to the RPC, providing a means of communicating between Java applications using normal method calls, and offering the capability for applications to run on different computers. The RMI uses also skeletons to connect the server to the RMI framework, stubs that act as a proxy server in the client's environment, and a registering mechanism to store the location and name of the

[6]The XML based middleware that overcomes DCOM (Distributed Common Object Model).

[7]Also designated as software bus or middleware.

server object. The major advantages of RMI are its better performance and it its easy interface (allows to pass any Java object as arguments).

IBM and Sun, with the cooperation of the OMG (Object Management Group), jointly developed RMI over IIOP (Internet Inter-ORB Protocol), so called RMI-IIOP, which joined together the interoperability of CORBA and the easy development of RMI. The implementation of the interface platform using RMI-IIOP was the easiest when compared with the CORBA and JRMI approaches.

The choice of the distributed object platform should take in consideration the easy mapping of MMS-based services, the easy integration with programming environment, and the ability to support heterogeneity and the real-time constraints. A comparison between CORBA, JRMI and RMI-IIOP, to support the client-server model of ADACOR resource integration approach, can be found in [Leitão et al., 2003a].

3.6 Summary

In this chapter, the ADACOR holonic architecture for the manufacturing control was introduced, addressing the improvement of the flexibility and agility, mainly in industrial scenarios with frequent occurrence of unexpected disturbances.

ADACOR is based upon a set of autonomous and cooperative holons, each one possessing a partial and imprecise perspective of the system, needing to cooperate to have a global view of the system or to reach its objectives. ADACOR architecture defines the product, task, operational and supervisor classes of holons to represent the manufacturing components, and supports the generalisation and aggregation concepts.

An adaptive production control approach was introduced, that distributes the control between the supervisor and operational holons, being supported by self-organisation. The self-organisation, based in the autonomy factor and propagation mechanism, allows balancing the production control between the stationary (presenting a hierarchical-like control structure) and transient (presenting a heterarchical-like control structure) control states, combining the agile reaction to disturbances with the global optimisation.

The manufacturing control system, as a whole, emerges from the interaction of autonomous and cooperative holons, each one contributing with its individual behaviour. Each identified type of interaction between the ADACOR holon classes are described using the AUML sequence diagrams.

The architecture of a generic ADACOR holon comprises a communication component, a decision component and a physical interface component. The decision component regulates the behaviour of the holon, performing the manufacturing control functions and providing self-organisation capability. The communication component uses an agent communication language and an appropriate ontology to standardise the communication process. The resource integration uses the virtual resource concept to make transparent the integration of physical manufacturing devices within holonic control applications. The ADACOR holons have also the capability to learn, allowing the acquisition of new knowledge to improve the holon performance.

In the next chapter, the dynamic behaviour of the ADACOR holons will be modelled, using the Petri Net formalism. The validation of the developed models will be also performed, to verify the correctness of these models.

Chapter 4

Modelling the Dynamic Behaviour of ADACOR Holons

"The one who is looking for the goal will be left empty when reaching it;
the one who has found the way will always carry the goal in his soul."
Nejc Zaplotnik

ADACOR architecture is based on a set of autonomous, intelligent and co-operative holons, which represent the factory components. Each individual holon has an autonomous behaviour, reaching individual manufacturing control objectives, and the cooperation of those holons is required to achieve global objectives.

Distributed manufacturing control systems, as ADACOR, are complex systems that are difficult to apprehend and to design, due to the presence of concurrent and asynchronous activities.

The formal modelling of the structural and behavioural specifications of the ADACOR control system is required to simplify the understanding and to get a comprehensive view of the system functionality, playing a key role in the design of manufacturing control systems.

In this chapter, the specifications and functionalities of each ADACOR holon class will be formally modelled, with special attention devoted to the behaviour of each individual ADACOR holon class, using the Petri Net (PN) formalism. The Petri net formalism ensures a rigorous specification and validation due to its powerful mathematical foundation.

The formal validation of the behavioural models, elaborated for the ADACOR holon classes, allows to verify the correctness of the model and the system specifications, and to elaborate a performance analysis.

4.1 Petri Nets to Model the Dynamic Behaviour

A model is an abstract representation of a portion of reality, intended to promote its understanding. The advantage to model the reality or a system, is that it can be made more explicit, simpler and easier to manipulate than the reality it is supposed to represent [Ferber, 1999]. Modelling is applied in different fields to represent the real world, from the use of mockups, developed in wood, plastic, or other substances, by architects to reproduce physical buildings, to the Newton's laws of motion, which establish a relation between the forces acting on a body and the motion of the body, that are a mathematical model that regulates the real movement of the bodies.

In order to model the structure and the behaviour of the holonic manufacturing control systems, and to validate the correctness of those models, to understand and to synthesise the system specifications, it is important to have the support of a formal modelling methodology.

The modelling of the dynamic behaviour of the system requires a formal tool that captures characteristics like concurrency, asynchronous operations, deadlocks, conflicts or resource sharing, which are inherent to FMS [Silva and Valette, 1989]. Additionally, the formal modelling tool must have the capability to validate the behavioural characteristics of these event-driven systems, and also to analyse other important aspects, such as the deadlock detection and the performance analysis.

UML (Unified Modelling Language) [Rumbaugh et al., 1998] is a modelling tool adequate to model object-oriented systems, but doesn't support efficiently the modelling of their dynamic behaviour and the formal validation of these specifications [Leitão et al., 2003c]. Additionally, UML fails to capture the autonomous and proactive behaviour of the agents, as well as the richness of their interactions [Wooldridge and Ciancarini, 2000].

On the other hand, Petri net is a formal modelling tool, both graphical and mathematical, that seems adequate to model and to analyse the behaviour of complex event-driven systems characterised as being concurrent, asynchronous, stochastic and with high distribution degree, such as flexible manufacturing systems. In comparision with UML, the Petri net formalism allows to design the control system behaviour, but also to validate and to verify the behaviour of the system, based in the mathematical foundation embedded in the Petri net formalism.

In this sense, the proposed methodology uses a kind of Petri net tailored for production control modelling purposes, proposed in [Colombo and Carelli, 1997], to model the behaviour of the ADACOR manufacturing control system.

A Petri net is directed graph defined by a four tuple, $PN = \{P, T, I, O\}$, where

- $P = \{p_1, ..., p_m\}$ is a finite set of places.

- $T = \{t_1, ..., t_n\}$ is a finite set of transitions.
- $I : (P \times T) \rightarrow \mathbb{N}$ is an input function that defines directed arcs from places to transitions. Each element of I represents the weight of the input arc from the place p_i to the transition t_j.
- $O : (P \times T) \rightarrow \mathbb{N}$ is an output function that defines directed arcs from transitions to places. Each element of O represents the weight of the output arc from the transition t_j to the place p_i.

Petri nets can be represented graphically, using circles to represent places and bars to represent transitions. Places and transitions are connected by directed arcs, pictured by arcs with arrows. It is not possible to connect places to places nor transitions to transitions. An integer value, adjacent to an arc, represents the weight of the arc; if there is no value associated to an arc, a unit weight is assumed.

A marked Petri net contains tokens, pictured by black dots, in addition to the previous elements. Tokens reside in places, travel along arcs and their flow through the net is regulated by transitions [Desrochers and Al-Jaar, 1995].

The marking of a Petri net is a function $m : P \rightarrow \mathbb{N}$ that gives the number of tokens in each place $p \in P$. The presence or absence of tokens indicates the status of a place. Formally, a marked Petri net is defined by the five tuple $PN = \{P, T, I, O, m\}$. The initial marking, m_0, represents the initial state of the system.

The dynamic evolution of the Petri net can be described through the following enabling and firing rules:

- A transition is enabled when each one of its input places contains at least the number of tokens equal to the weight of its arc, i.e. considering $\cdot t$ as the set of input places for a transition t, this transition is enabled if and only if $\forall p_i \in \cdot t, m(p_i) \geq I(p_i, t)$.
- An enabled transition t fires reaching a new marking m' by following the rule: $m'(p) = m(p) + O(p,t) - I(p,t)$. On other words, when an enabled transition fires the number of tokens specified in the input arcs are removed from each input place, and the number of tokens specified in the output arcs are placed in each output place.

The set of all markings reachable from an initial marking m_0 is designated the reachability set of m_0 and represented by $R(m_0)$. A Petri net is reversible if for any $m \in R(m_0)$ there exists a firing sequence of transitions such that $m_0 \in R(m)$.

The tokens, places and transitions have assigned a meaning for the proper interpretation of the model. In a manufacturing environment, and also in this work, they are interpreted as following [Desrochers and Al-Jaar, 1995]:

- Places represent the states of the system. The existence of tokens in a place indicates the status of a place (active or not), or the availability of the resources (for example the number of tokens can mean the number of empty spaces in a buffer).
- A transition firing represents an activity.

A modelled system can comprise activities that take place at a much faster (or slower) pace than others. Additionally, it may be required the introduction of transitions that corresponds to purely logical aspects of the system behaviour, which has no associated time [Colombo et al., 1997]. In these circumstances, the temporised Petri net used in this work considers two distinct types of transitions:

- **Immediate transitions**, drawn by a thin bar, which fire in time zero, i.e. the time between the event that notifies the begin of the activity and the event that indicates its end is zero. This type of transition can be used to model atomic activities, such as sending a message or downloading a program.
- **Timed transitions**, drawn by a thick bar, which have associated a delay time that specifies the amount of time that must elapse before the transition fires. This type of transition is used to represent temporal activities, i.e. time consuming activities. As an example, a machine repair is a temporal activity that begins with the event *repair initiated* and some time later finishes with the event *repair completed*.

The timed transitions have a three-phases firing process [Colombo et al., 1997]. In the first phase, the transition is marked enabled and captures the marks from the precedent places (instantaneous). In a second phase, the transition is evolving, i.e. the modelled activity is being performed. This phase has associated a certain number of time-units as parameter, corresponding to the modelled activity. As soon as this time elapses, the transition starts the third phase and puts the corresponding marks in the subsequent places (instantaneous).

In order to achieve a formal specification of the logic control structure, a top-down methodology is used, by refining step by step some timed transitions to include enough system operation details for purpose of hardware implementation, i.e. replacing a timed transition by a more detailed and refined sub-net so that a large Petri net can be obtained, as illustrated in the Figure 4.1.

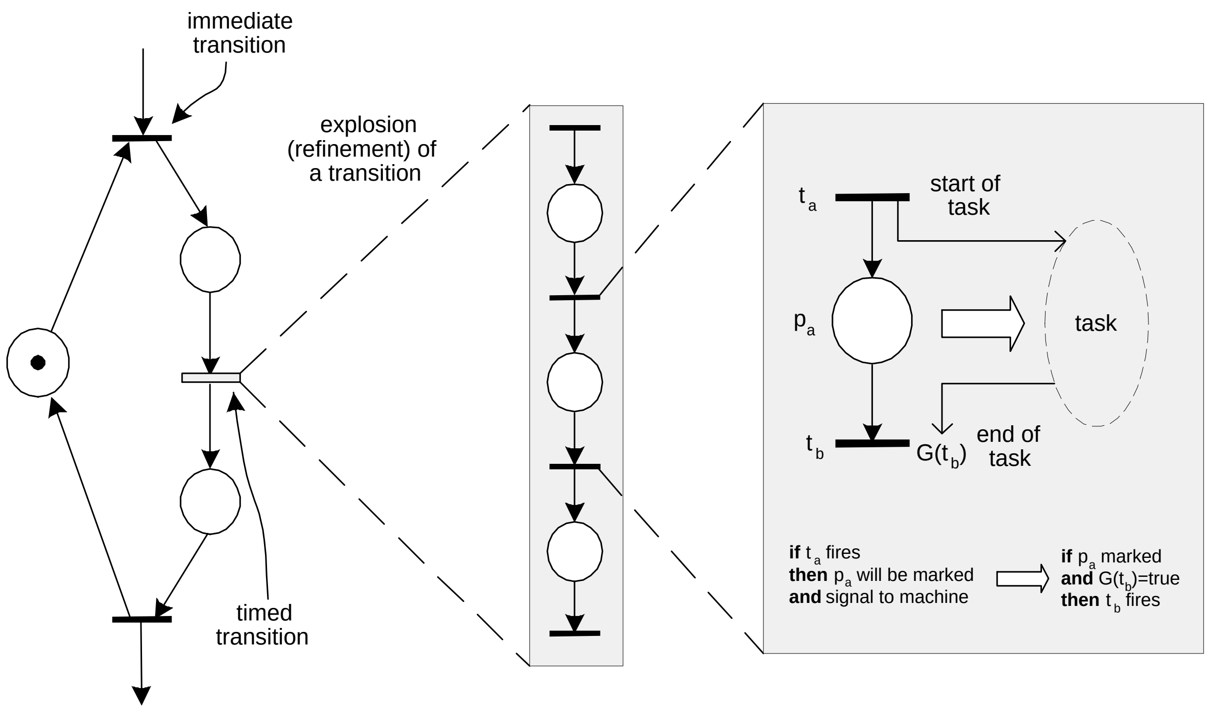

immediate transition
explosion (refinement) of a transition
timed transition
t_a
start of task
p_a
task
t_b
$G(t_b)$
end of task
if t_a fires **then** p_a will be marked **and** signal to machine
if p_a marked **and** $G(t_b)$=true **then** t_b fires

Figure 4.1: Refinement of Timed Transitions in the Petri net Formalism

An important aspect when replacing a timed transition by a more refined sub-net is to guarantee that the large Petri net preserve its live and bounded properties. A number of theorems have been established by [Vallete, 1979] and latter generalised by [Suzuki and Murata, 1983] about the preservation of boundedness and liveness in Petri nets obtained using the step by step refinement. Vallete has proven that [Vallete, 1979]:

- Considering a transition t_i of a Petri net Z, a well-formed[1] sub Petri net Z' and a Petri net Z'' as the result of the substitution of t_i by Z',
- The Petri net Z'' is: bounded if Z is bounded, safe if Z and the sub Petri net Z' are safe, and live if Z is live.

This theorem points out that using stepwise refinement, it is not necessary to do the analysis of the detailed and large Petri net, because all of its properties can be deduced from the analysis of the initial Petri net and each one of the sub Petri nets.

[1]A block is a Petri net with one and only one initial transition, t_i, and one final transition t_f. The associated Petri net of the block Petri net, $\widehat{PN}$, is the net obtained by adding a place, p_o, to the block, where t_i is the only output transition of p_o, t_f is the only input transition of p_o and the initial marking, $\widehat{m_o}$, is $m(p_o) = 1$. A block Petri net is said to be well-formed if $\widehat{PN}$ is live, $\widehat{m_o}$ is the only initial marking of $\widehat{PN}$ and the only transition enabled by $\widehat{m_o}$ is t_i.

More details about Petri net theory and mathematical fundamentals can be found in [Murata, 1989, Desrochers and Al-Jaar, 1995, Colombo and Carelli, 1997, Colombo et al., 1997].

4.2 Product Holon Behavioural Model

Each available product to be produced by the factory plant is represented by a **product holon** that contains all knowledge related to the product and is responsible for the short-term process planning. The behavioural model of the product holon is illustrated in the Petri net model of the Figure 4.2 [Leitão et al., 2003c].

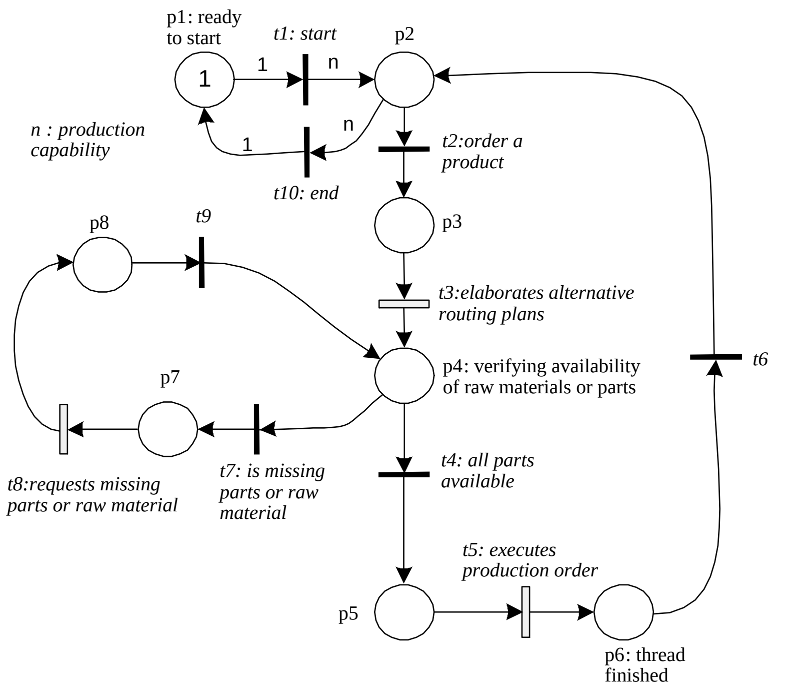

p1: ready to start
t1: start
p2
1
n
1
n
1
n : production capability
t2:order a product
t10: end
p8
t9
p3
t3:elaborates alternative routing plans
p4: verifying availability of raw materials or parts
t6
p7
t8:requests missing parts or raw material
t7: is missing parts or raw material
t4: all parts available
t5: executes production order
p5
p6: thread finished

Figure 4.2: Behaviour Model of the Product Holon

The product holon starts its execution entering in a state waiting for new product orders. Each new order will generate a new thread to handle its execution, comprising the following main activities: short-term process planning, management of the sub-parts and management of the production order execution. The product holon continues waiting for new orders, being able to process simultaneously several product orders, limited by the production capability n.

4.2.1 Short-term Process Planning

The process planning represents the link between the engineering design and the shop floor, and typically involves the material selection, the process selection, the process sequencing and the machine parameter selection. In the context of this work, the process planning is the short-term one, that is the elaboration of alternative routing plans according to the available resources.

The *elaborates alternative routing plans* activity defines several alternative plans for the execution of the product, based in the knowledge related to the sequence of operations and in the available resources in the system. The transition t_3 from the model of Figure 4.2, that represents this activity, can be exploded in a sub-Petri net model, illustrated in the Figure 4.3.

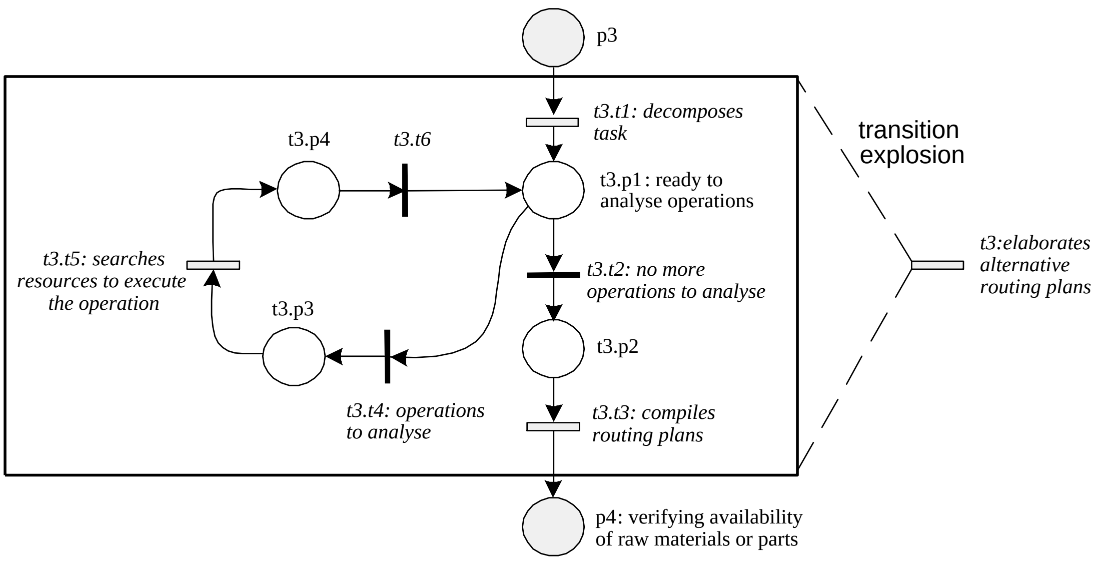

p3
t3.t1: decomposes task
t3.p4
t3.t6
t3.p1 : ready to analyse operations
t3.t5: searches resources to execute the operation
t3.t2: no more operations to analyse
t3.p3
t3.p2
t3.t4: operations to analyse
t3.t3: compiles routing plans
p4: verifying availability of raw materials or parts
transition explosion
t3:elaborates alternative routing plans

Figure 4.3: Model for the *Elaborates Alternative Routing Plans* Activity

In this activity, the product holon reads the process plan and decomposes the task into a set of operations ($\Theta_i = \{o_{i1}, o_{i2}, o_{i3}, ..., o_{in}\}$). It also gets the list of precedences between the operations ($\Omega_i = \{< o_{ij}, o_{ik} > | o_{ij} \prec o_{ik}\}$) and the list of requirements for each operation ($\mathcal{B}_{ik} = \{B_{ikz} | z \in I\}$) that a machine should satisfy to be able to execute the operation.

The *searches resources to execute the operation* activity searches in the actual plant resources, those that fulfil the requirements imposed by each operation o_{ik}. The set of resources $\mathcal{R}_{ik}$ contains all the resources whose skills match the requirements of the operation o_{ik}. A resource R_j possessing the set of skills $\mathcal{S}_j$, has abilities to execute an operation o_{ik} if

$$\mathcal{B}_{ik} \subseteq \mathcal{S}_j \Leftrightarrow \forall x \in \mathcal{B}_{ik} \Rightarrow x \in \mathcal{S}_j$$

that is, a resource has abilities to execute an operation if it fulfils all the requirements presented by the operation.

The decision about which is the best resource to execute the operation, is done according to some confidence indicator, using the information learned from the previous experiences, in terms of percentage of executed, delayed and cancelled work orders.

When all operations are analysed, the elaborated alternative routing plans are compiled in the form of $\Gamma = \{\Theta_i, \Omega_i, \mathcal{R}_{ik}\}$, which can also be represented as a graph.

The detailed study of process planning is out of scope of this thesis. Detailed work related to process planning can be found in [Kruth et al., 1996, Marapoulos, 1995, Cho and Wysk, 1995].

4.2.2 Management of the Sub-Parts

The product holon, according to the product structure, verifies if the required parts or raw materials are available on the storage system, interacting with the operational holon that is responsible for the storage management. When the raw materials or parts are not available, the product holon interacts with other product holons to request the execution of sub-products, invoking recursively the product holon model.

When all sub-products or raw materials are available, the net evolves to the *executes the production order* activity.

4.2.3 Management of the Production Order Execution

The transition t_5 of the product holon model, that represents the *executes the production order* activity, can be exploded in the sub-Petri net model illustrated in the Figure 4.4. In this activity, the product holon launches a task holon that will be responsible for the management of the production order execution[2], passing the information related to the alternative routing plans.

At this moment, the control is transferred to the task holon, and the product holon waits for the conclusion of the execution of the production order. The notification is logically represented by marking the place p_a which output transition is a transition of the task holon model[3]. This place allows the shynchronisation of both Petri net models.

When the execution finishes, the task holon notifies the product holon, providing the relevant information related to the execution of the production order, by marking the place p_b.

The product holon gather new knowledge by analysing these data, giving rewards to the resources that executed with success the work orders and penalties to the resources that had failures, delays or did not fulfil the quality requirements, in the execution of the allocated work

[2]A production order is created based on the demand, i.e. from the customer or forecast orders. However, the production order can also aggregate several customer and/or forecast orders, aiming to take advantage of batch production.

[3]In order to make easier the description of each Petri net model, the places that synchronises the Petri net models are not represented or discussed during the stepwise refinement.

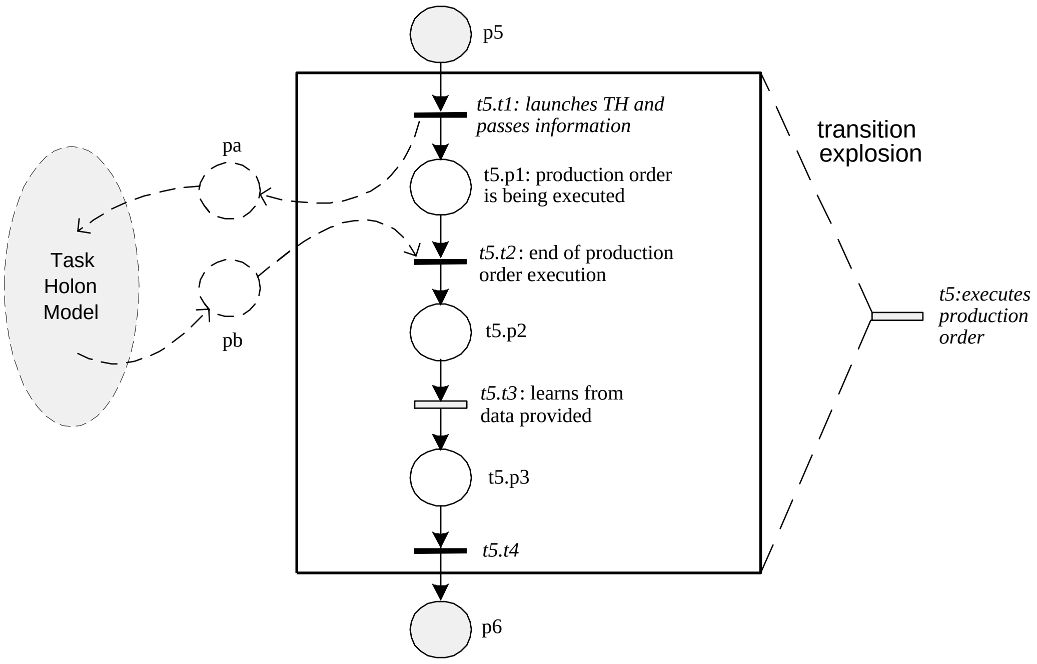

p5
t5.t1: launches TH and passes information
t5.p1: production order is being executed
t5.t2: end of production order execution
t5.p2
t5.t3: learns from data provided
t5.p3
t5.t4
p6
pa
pb
Task Holon Model
transition explosion
t5:executes production order

Figure 4.4: Model for the *Executes Production Order* Activity

orders. The new knowledge will support the elaboration of more efficient and accurate routing plans in the future, by a more precise information about the performance of each resource.

4.3 Task Holon Behavioural Model

Each available production order, launched to produce a part, is represented by a **task holon** that is responsible to manage its execution. The dynamic behaviour of a task holon comprises mainly the order decomposition, resource allocation planning and plan execution activities, as illustrated in the Petri net-model of Figure 4.5 [Leitão et al., 2003c].

4.3.1 Task Decomposition

Initially, the task holon requests a pallet and material to the storage system. According to the tyep of production, the pallet can contain several parts of the same material or contain all necessary parts to execute the final part. The number of pallets in the factory plant is regulated by the value m in the monitor place p_3[4].

The task decomposition involves the decomposition of a production order into a set of individual work orders and the analysis of the process plan provided by the product holon. The

[4]A monitor place, introduced by [Murata, 1989], possesses an initial marking that represents the limited number of real components that will move throughout the net model, such as pallets or orders.

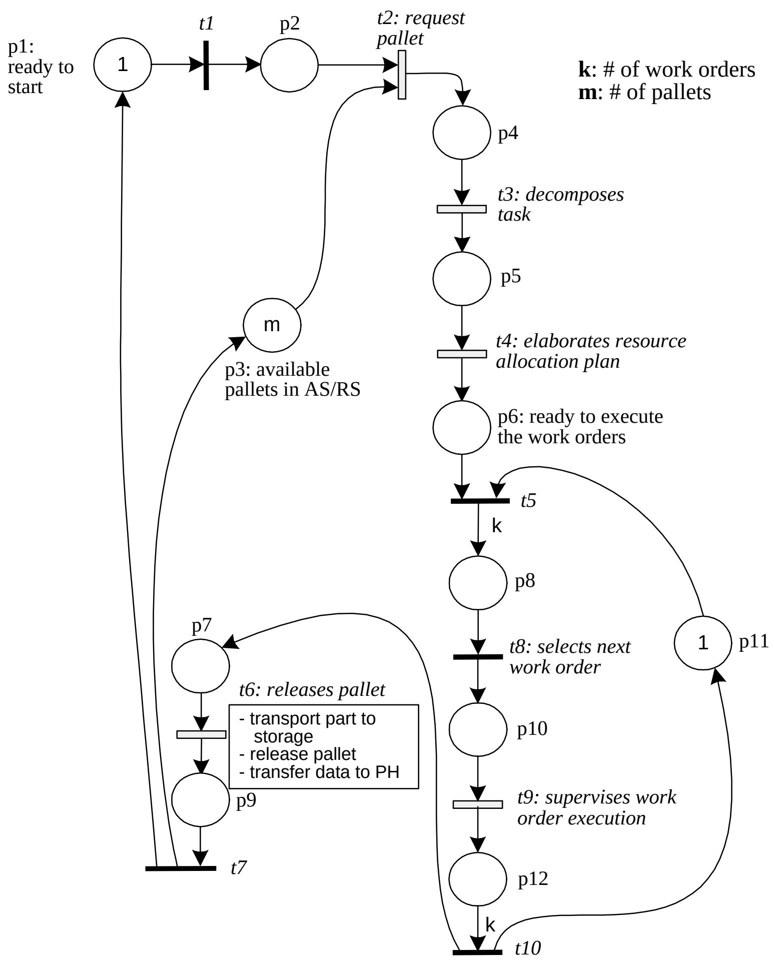

p1: ready to start
1
t1
p2
t2: request pallet
k: # of work orders
m: # of pallets
p4
t3: decomposes task
p5
m
p3: available pallets in AS/RS
t4: elaborates resource allocation plan
p6: ready to execute the work orders
t5
k
p8
p7
t8: selects next work order
1
p11
t6: releases pallet
- transport part to storage
- release pallet
- transfer data to PH
p10
p9
t9: supervises work order execution
t7
p12
k
t10

Figure 4.5: Task Holon Behaviour Model

plan resulted from this activity is not completely defined, since is missing the allocation of each work order to a plant resource.

4.3.2 Resource Allocation Planning

The *elaborates resource allocation planning* activity, represented by the special temporised transition t_4 and exploded in the sub-Petri net model illustrated in Figure 4.6, executes the allocation of each work order to an available resource.

The task holon interacts directly with the supervisor or operational holons, according to the current organisational control structure, applying a mechanism to allocate each work order in the production order. The allocation of an individual work order starts with its announcement, in a multicast manner, to the resources that possibly can match the requirements imposed by the operation. In the announcement, it is indicated the start date and due date, the list of requirements and the precedences that each work order involves.

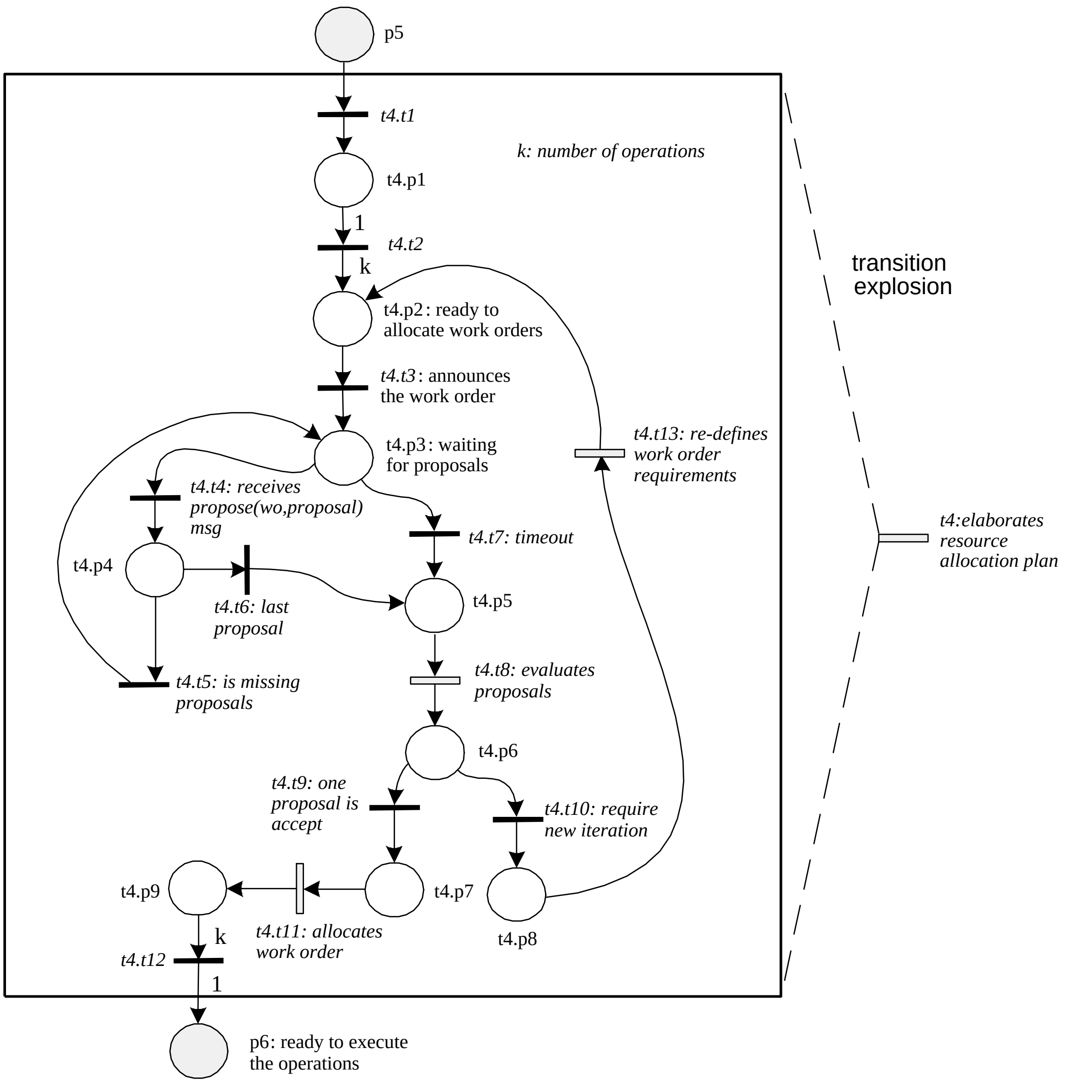

p5
t4.t1
k: number of operations
t4.p1
1
t4.t2
k
transition explosion
t4.p2: ready to allocate work orders
t4.t3: announces the work order
t4.p3: waiting for proposals
t4.t13: re-defines work order requirements
t4.t4: receives propose(wo,proposal) msg
t4.t7: timeout
t4:elaborates resource allocation plan
t4.p4
t4.t6: last proposal
t4.p5
t4.t5: is missing proposals
t4.t8: evaluates proposals
t4.p6
t4.t9: one proposal is accept
t4.t10: require new iteration
t4.p9
t4.p7
t4.p8
k
t4.t11: allocates work order
t4.t12
1
p6: ready to execute the operations

Figure 4.6: Model for the *Elaborates Resource Allocation Plan* Activity

The task holon collects the proposals sent by the holons interested to execute the work order. When all proposals have arrived or the time defined to wait for proposals was elapsed, the task holon initiates an evaluation procedure to allocate the work order to the best proposal.

The evaluation procedure may combine the analysis of the proposed price, the location of the resource, the proposed due date and the degree of confidence about the holon. The task holon adjusts dynamically the evaluation parameters, based in the knowledge learned from previous interactions.

In case of allocation, the task holon sends an *accept(work order)* message to the selected operational holon, and *reject(work order)* messages to the other holons. If no proposals fulfill the expectatives of the task holon, it starts a new iteration, re-defining the specifications of the work order.

4.3.3 Plan Execution

After the allocation of all work orders, the task holon starts the execution of the achieved plan, interacting with the operational holons, even if the allocation is provided by the supervisor holon. The task holon, according to the schedule plan, selects the next work order to be executed, and triggers the *supervises work order execution* activity, which is exploded in the sub-Petri net of the Figure 4.7.

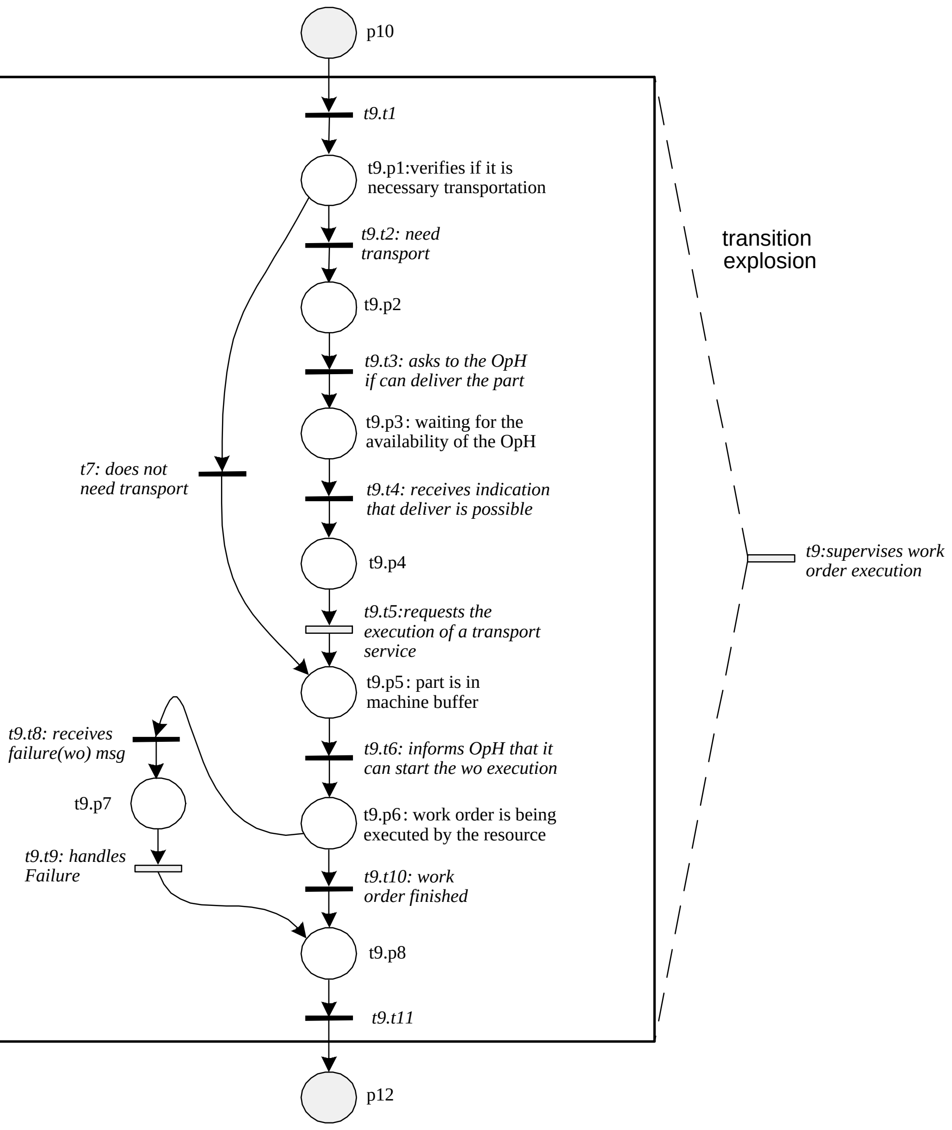

p10
t9.t1
t9.p1:verifies if it is necessary transportation
t9.t2: need transport
transition explosion
t9.p2
t9.t3: asks to the OpH if can deliver the part
t9.p3: waiting for the availability of the OpH
t7: does not need transport
t9.t4: receives indication that deliver is possible
t9.p4
t9:supervises work order execution
t9.t5:requests the execution of a transport service
t9.p5: part is in machine buffer
t9.t8: receives failure(wo) msg
t9.t6: informs OpH that it can start the wo execution
t9.p7
t9.p6: work order is being executed by the resource
t9.t9: handles Failure
t9.t10: work order finished
t9.p8
t9.t11
p12

Figure 4.7: Model for the *Supervises Work Order Execution* Activity

This activity involves a set of preparation actions, before the execution phase. Initially, the task holon asks if the machine can receive the part, for example to avoid deadlocks due to missing space in the machine buffer. When the operational holon notifies the availability to receive the part, the task holon requests the transportation of the part to the machine.

The *requests the execution of a transport service* activity is the request of a transport service to the operational holon that has ability to execute this type of service, being the transport operation executed according to the availability of the transport system.

When the transporter device finishes the transportation of the part to the machine buffer, the task holon notifies the operational holon, in charge of executing the work order, that it can start the processing of the part. This synchronisation is required in order to guarantee that the producer resource can perform a set-up operation, if necessary. Once the work order is started, the control is given to the operational holon, and the task holon waits for the end of the work order execution.

The described procedure for one work order is repeated for all the work orders that belong to the production order.

4.3.4 Pallet Release

After the execution of all the work orders, belonging to the production order, the task holon executes a set of actions, illustrated in the model of Figure 4.8, related to the release the pallet, and to finalise the task holon life-cycle.

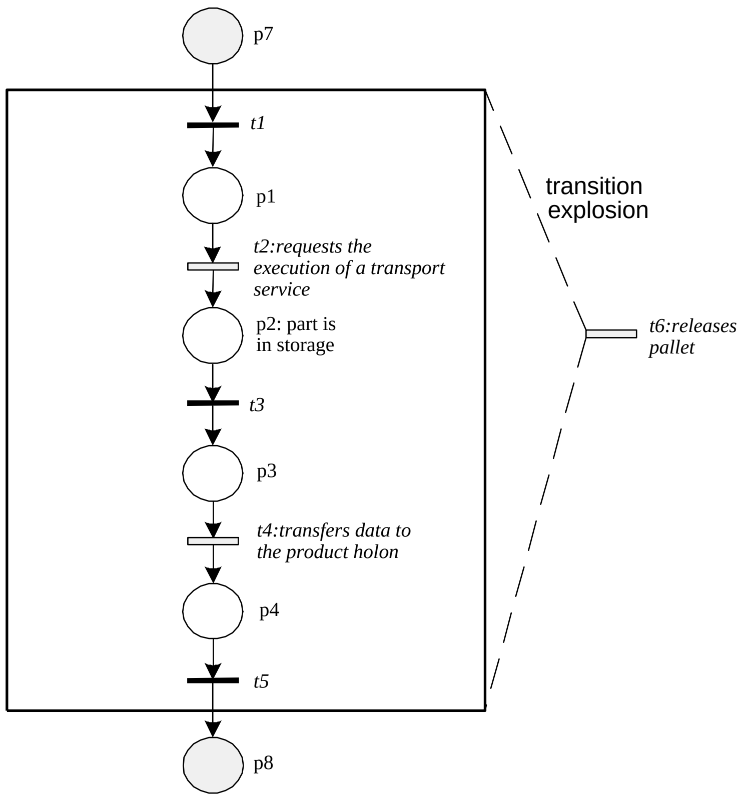

p7
t1
p1
t2:requests the execution of a transport service
p2: part is in storage
t3
p3
t4:transfers data to the product holon
p4
t5
p8
transition explosion
t6:releases pallet

Figure 4.8: Model for the *Releases Pallet* Activity

In this activity, the task holon requests to the transport system the transportation of the part to the storage facility. When the part is already stored in the storage facility, the task holon releases the pallet.

The task holon finishes its execution by transferring to the product holon the relevant information about the production order execution.

In case of a production order that only requires the assembly of several other sub-products, the global assembly is decomposed in several work orders, which requires for each one a different part that was been previously executed. Each work order is allocated to a specific assembly station and the task holon manages the assembly task execution, by requesting the parts, according to the execution evolution.

4.4 Operational Holon Behavioural Model

The **operational holons** represent the physical manufacturing resources, such as operators, robots and numerical control machines, managing their behaviour according to the resource goals, constraints and skills, and optimising their agenda. Each operational holon represents one physical resource. The dynamic behavioural model of the operational holon is represented in Figure 4.9 [Leitão et al., 2003c].

Initially, the operational holon initialises its components (i.e. the communication, decision, and physical interface) and registers its skills and capacity in the appropriated supervisor holon, according to the organisational control structure.

The operational holon model contains several sub-behaviours that are handled asynchronously in parallel, so that the execution of one process doesn't block the execution of another process; for example, when executing a work order, the operational holon can monitorise its execution or can handle the announcement of new work orders.

4.4.1 Work Order Announcement

In presence of a work order announcement, i.e. the arrival of an *announce(work order)* message, the operational holon, based on its local schedule, analyses its availability to execute the work order, as illustrated in the sub-Petri net model of Figure 4.10.

The analysis of the availability to execute the work order comprises the verification if the holon has the required skills to execute the work order, and the verification if it has capacity to execute the work order fulfilling the specified due date and quantity.

The capacity is verified by re-scheduling the work orders, trying to find an empty space to allocate the work order. Each operational holon has a scheduling mechanism to schedule the

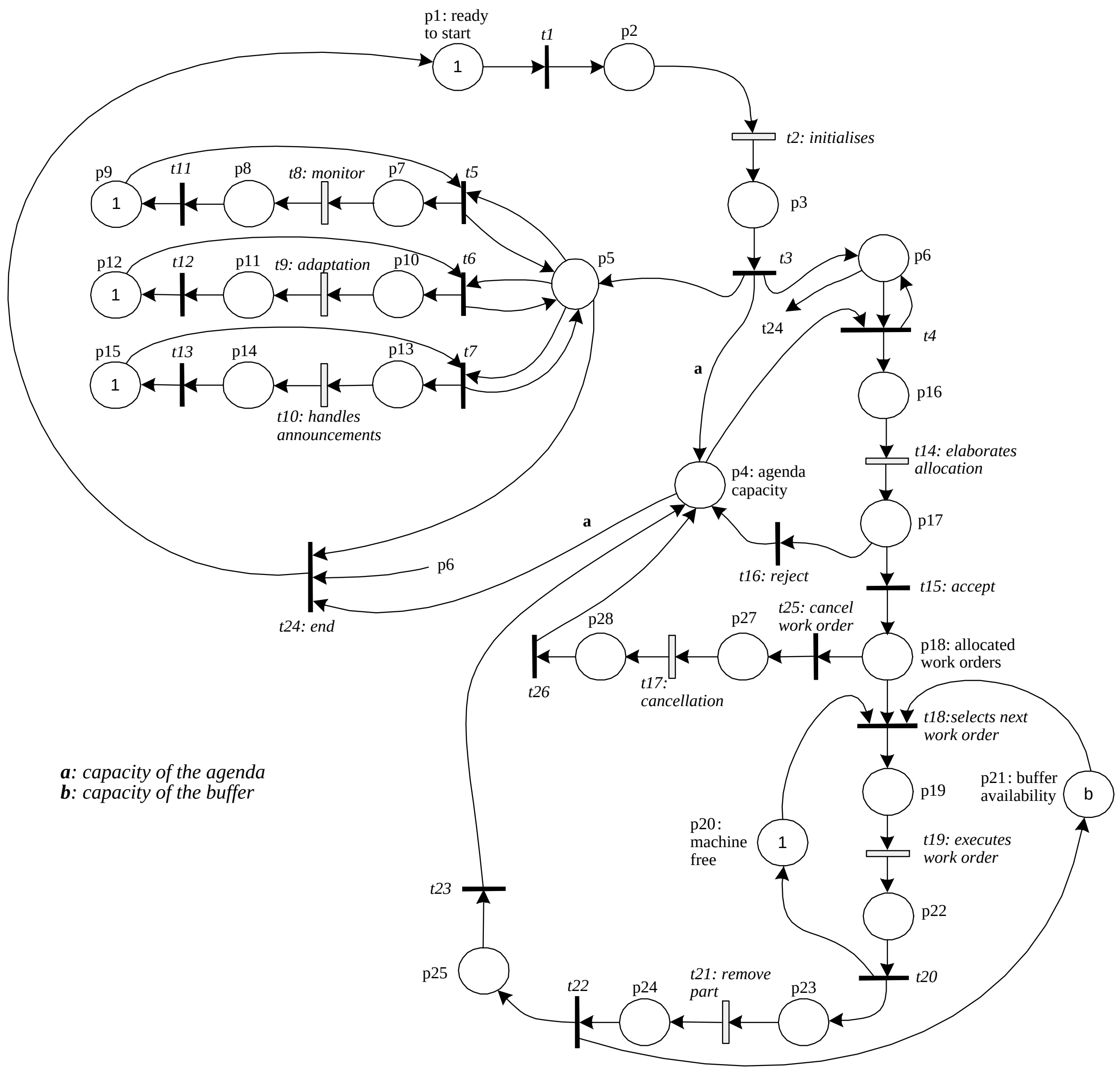


Figure 4.9: Operational Holon Behaviour Model

work orders over the time, addressing the problem of multiple jobs for a single machine.

In case that the requirements are fulfilled and exists capacity to execute the work order, the operational holon has the ability to respond to the sender holon biding the announcement; otherwise, the holon refuses the announcement, sending a *refuse(work order)* message.

In case of availability, the operational holon calculates the price to execute the work order, includes it in the proposal and sends a *propose(work order,proposal)* message to the task holon.

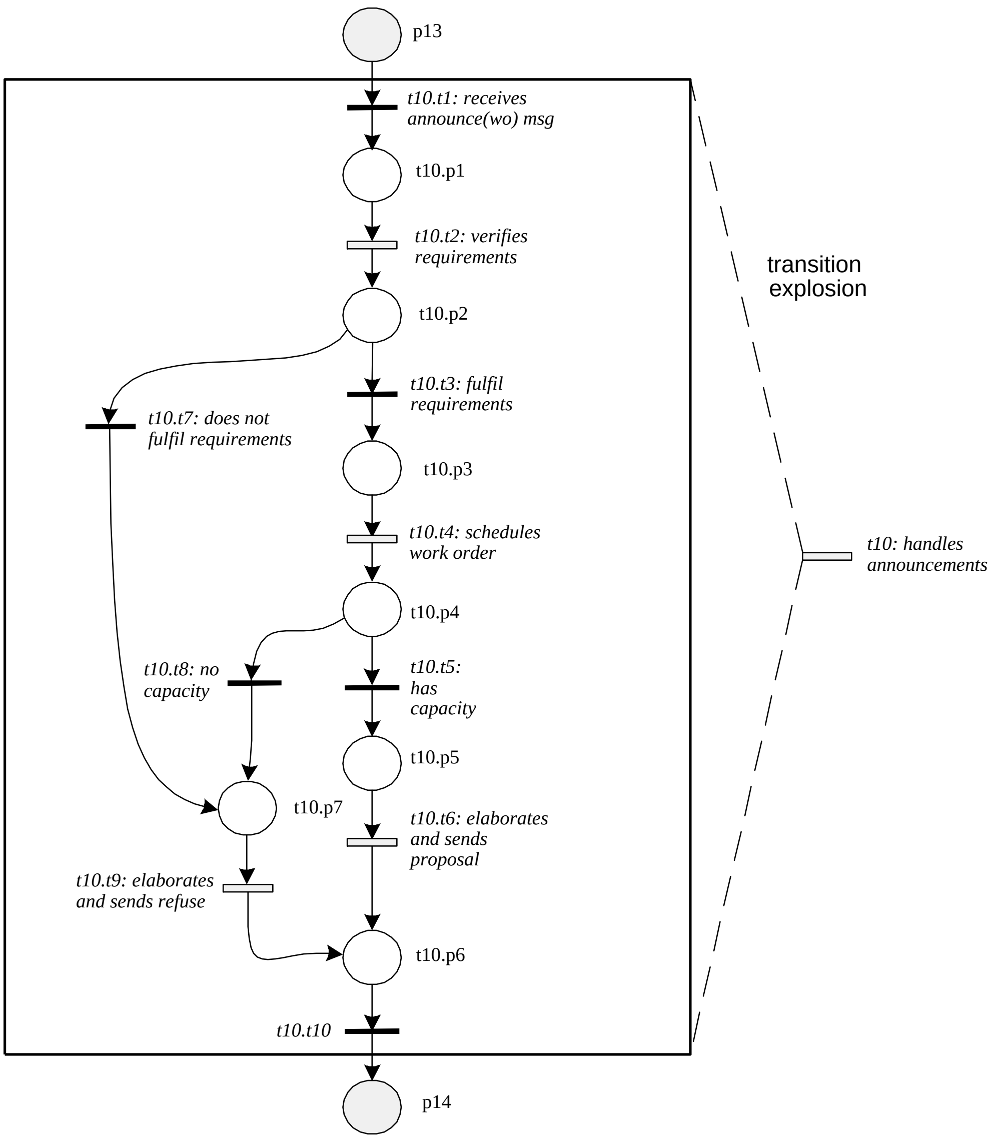

p13
t10.t1: receives announce(wo) msg
t10.p1
t10.t2: verifies requirements
t10.p2
t10.t3: fulfil requirements
t10.t7: does not fulfil requirements
t10.p3
t10.t4: schedules work order
t10.p4
t10.t8: no capacity
t10.t5: has capacity
t10.p5
t10.p7
t10.t6: elaborates and sends proposal
t10.t9: elaborates and sends refuse
t10.p6
t10.t10
p14
transition explosion
t10: handles announcements

Figure 4.10: Model for the *Handles Announcements* Activity

4.4.2 Work Order Allocation

When an operational holon receives a proposal for the execution of a work order, it decides the acceptance according to its autonomy degree and actual agenda capacity, modelled by the *allocates work order* activity, illustrated in Figure 4.11.

The proposal for the execution of a work order can be of one of three different types: (1) a request for an auxiliar operation (for example, mover or transport operations that does not need to be previously scheduled) that requires the verification if the resource skills matches the work order requirements and the verification of the actual capacity, (2) an award decision related to a previous work order proposal sent by the operational holon, which requires the confirmation

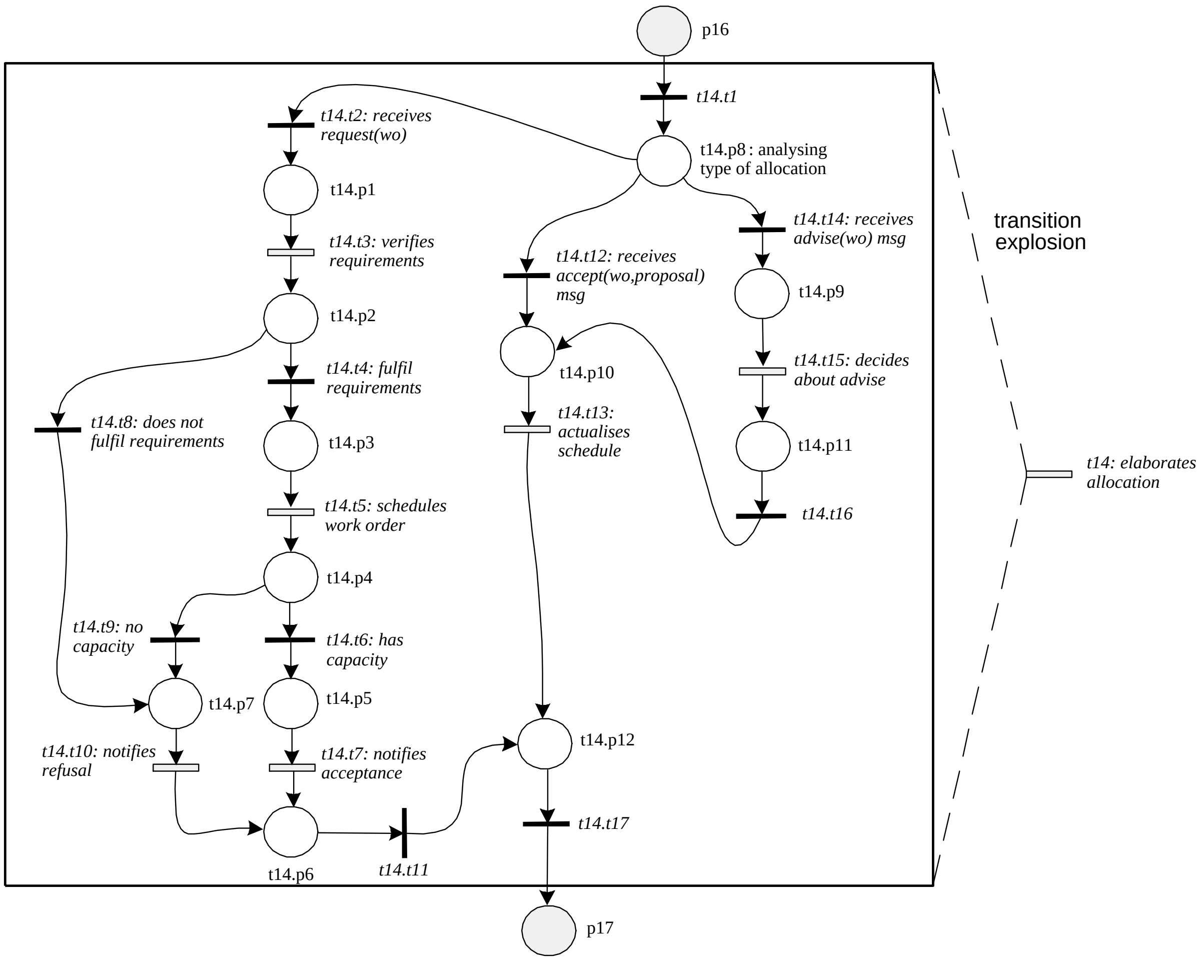

p16
t14.t1
t14.t2: receives request(wo)
t14.p8 : analysing type of allocation
t14.p1
t14.t3: verifies requirements
t14.t14: receives advise(wo) msg
transition explosion
t14.t12: receives accept(wo,proposal) msg
t14.p9
t14.p2
t14.t15: decides about advise
t14.p10
t14.t4: fulfil requirements
t14.t8: does not fulfil requirements
t14.t13: actualises schedule
t14.p3
t14.p11
t14: elaborates allocation
t14.t5: schedules work order
t14.t16
t14.p4
t14.t9: no capacity
t14.t6: has capacity
t14.p7
t14.p5
t14.p12
t14.t10: notifies refusal
t14.t7: notifies acceptance
t14.t17
t14.p6
t14.t11
p17

Figure 4.11: Model for the *Elaborates Allocation* Activity

of the current schedule, (3) a work order proposal sent by the supervisor holon that requires the analysis of the autonomy degree and the actual capacity of the holon.

The *decides about the advise* activity decides if the proposals provided by the supervisor holons are accepted or not, according to the adaptation mechanism described in chapter 3.

In case of rejection, due to lack of capacity or capabilities to execute the work order or because the execution of the work order is not interesting for the operational holon, the monitor place (p_4) that represents the agenda capacity is marked again with the previously removed token. In case of acceptance, the work order is stored in the local agenda, being added a token to the place p_{18} (*allocated work orders*), waiting for the appropriate moment to start its execution.

The cancellation activity handles the cancellation of a work order, notified by the *cancel(work order)* message sent by the task or supervisor holon. This activity involves mainly the remotion of the work order from the agenda and the execution of a re-scheduling to optimise the agenda.

4.4.3 Work Order Execution

The next work order to be executed by the manufacturing resource is selected according to the local scheduling and the availability of the buffer and the state of the machine. After the selection, the execution of the work order is prepared and then physically executed, represented by the timed transition t_{19} and illustrated in the model of Figure 4.12.

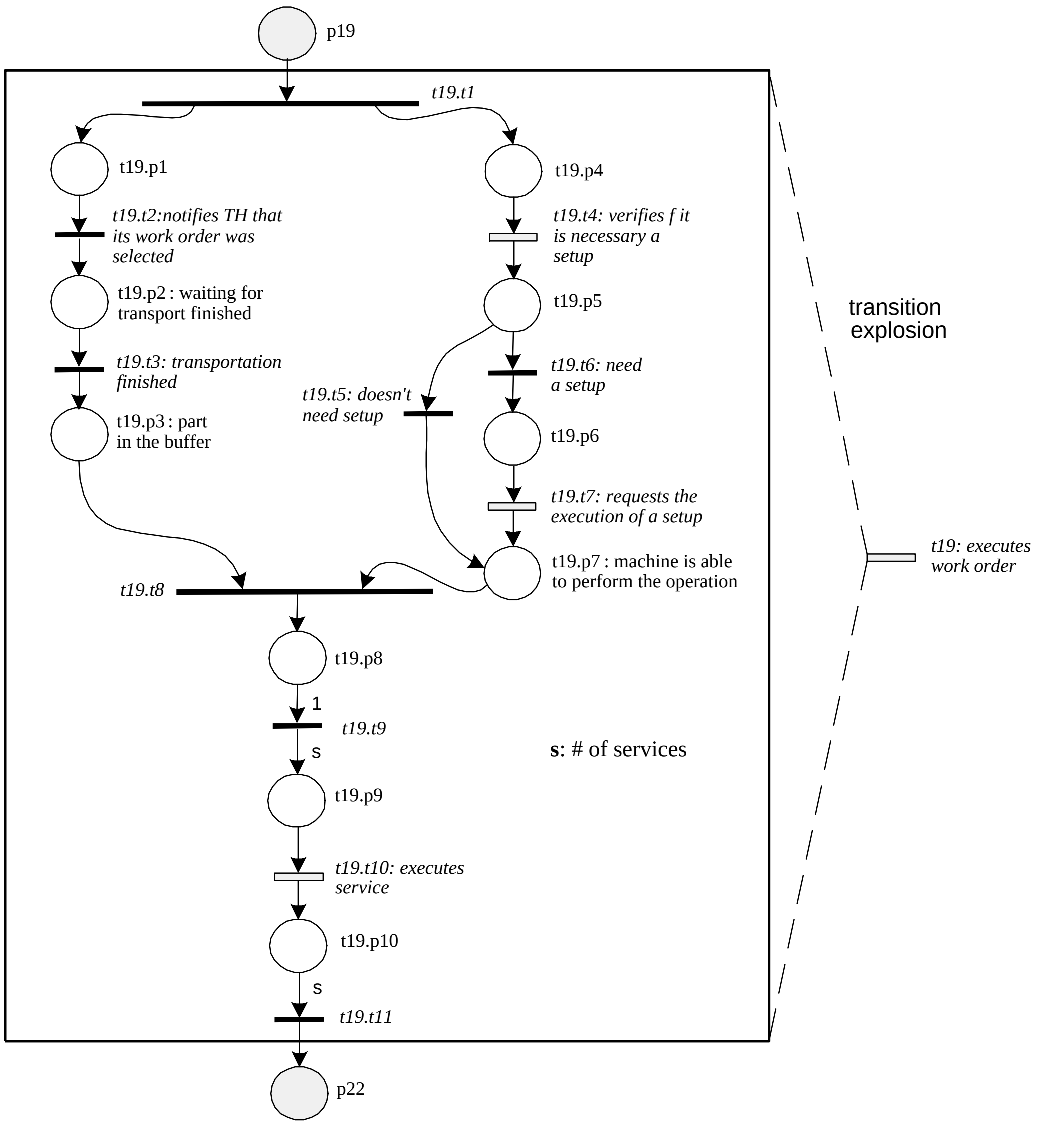

p19
t19.t1
t19.p1
t19.p4
t19.t2:notifies TH that its work order was selected
t19.t4: verifies f it is necessary a setup
t19.p2 : waiting for transport finished
t19.p5
transition explosion
t19.t3: transportation finished
t19.t6: need a setup
t19.t5: doesn't need setup
t19.p3 : part in the buffer
t19.p6
t19.t7: requests the execution of a setup
t19.p7 : machine is able to perform the operation
t19: executes work order
t19.t8
t19.p8
1
t19.t9
s
s: # of services
t19.p9
t19.t10: executes service
t19.p10
s
t19.t11
p22

Figure 4.12: Model for the *Executes Work Order* Activity

The preparation of the execution involves the transportation of the part to the machine and the execution of a set-up if necessary. The set-up aims to endow the machine with the required tools and fixtures to execute the work order. In case that the execution of a set-up is required, the operational holon deals directly with the operational holon that represents the team that executes the set-ups.

The execution of the work order is started after the preparation phase, involving normally the execution of several services. As an example, the execution of a machining program in a NC machine comprises the following main services: to load the program to the machine, to select the program in the machine memory and to execute the program.

The explosion of the *executes service* activity is illustrated by the sub-Petri net model of Figure 4.13, considering both the local service and the request for physical synchronisation. The transition *starts the execution of the service* represents the command for the physical execution.

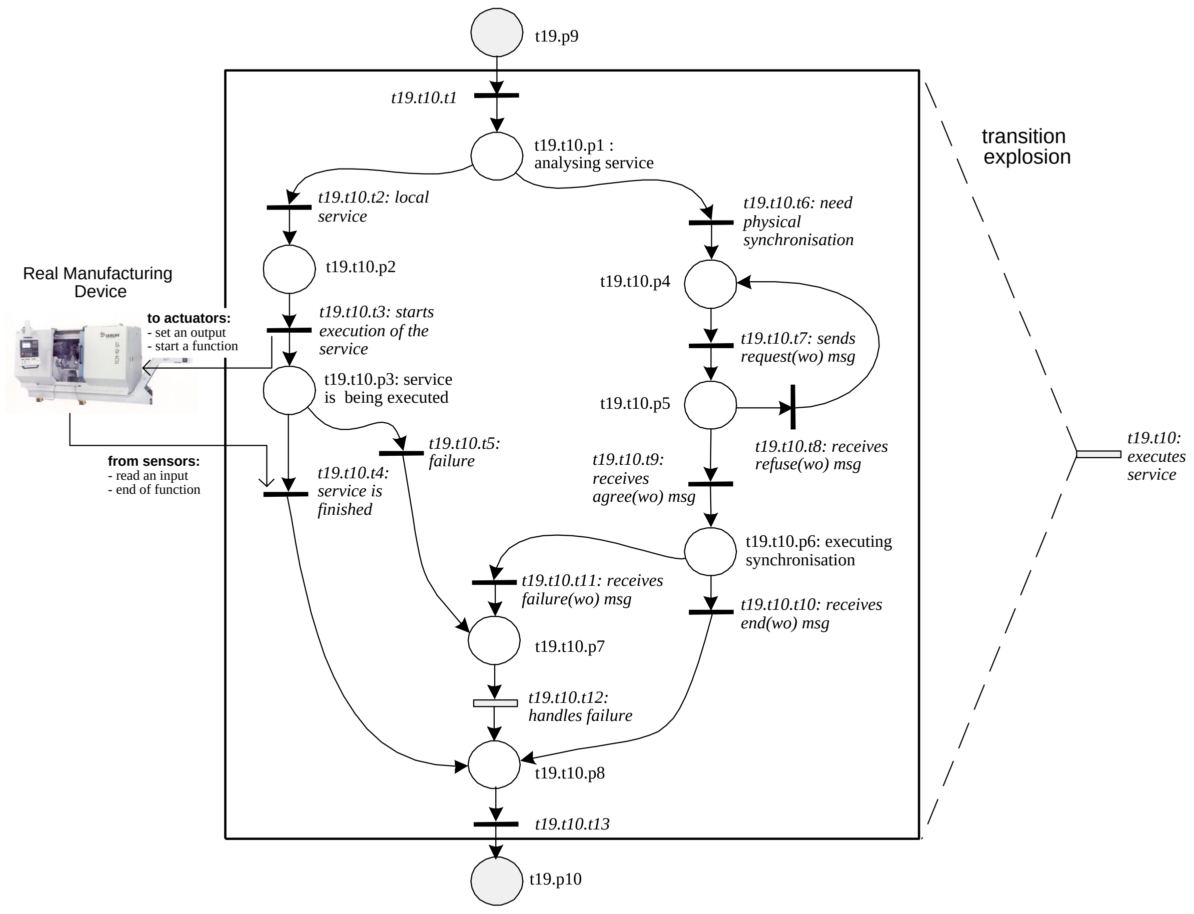

t19.p9
t19.t10.t1
t19.t10.p1 : analysing service
transition explosion
t19.t10.t2: local service
t19.t10.t6: need physical synchronisation
t19.t10.p2
t19.t10.p4
Real Manufacturing Device
to actuators:
- set an output
- start a function
t19.t10.t3: starts execution of the service
t19.t10.t7: sends request(wo) msg
t19.t10.p3: service is being executed
t19.t10.p5
from sensors:
- read an input
- end of function
t19.t10.t5: failure
t19.t10.t4: service is finished
t19.t10.t9: receives agree(wo) msg
t19.t10.t8: receives refuse(wo) msg
t19.t10: executes service
t19.t10.p6: executing synchronisation
t19.t10.t11: receives failure(wo) msg
t19.t10.t10: receives end(wo) msg
t19.t10.p7
t19.t10.t12: handles failure
t19.t10.p8
t19.t10.t13
t19.p10

Figure 4.13: Model for the *Executes Service* Activity

The transition *requests physical synchronisation* represents the request to another operational holon, for the synchronisation of their actions. For example, when a robot loads a part to the machine, it is necessary that the machine close its claws, in order to keep the part.

The work order is finished when all services are executed. When the work order finishes, the resource returns to the idle state, being able to initiate the execution of another work order, and the part is removed from the machine to the next machine, according to the resource allocation plan.

4.4.4 Monitoring and Adaptation

The monitoring and adaptation activities are related to manufacturing control functions that regulate the behaviour of the holon. These activities are performed concurrently, that are one thread for each activity.

In the operational holon behaviour it is necessary to have a detailed and up-to-minute knowledge of the status of its physical resource, using the passive and active monitoring forms.

The adaptation activity is related to the adaptation mechanism described in chapter 3, where the holon senses the propagation of the re-organisation need or the occurrence of an internal failure. This activity consists in the management of the holon's control behaviour during the transient state, as illustrated in Figure 4.14

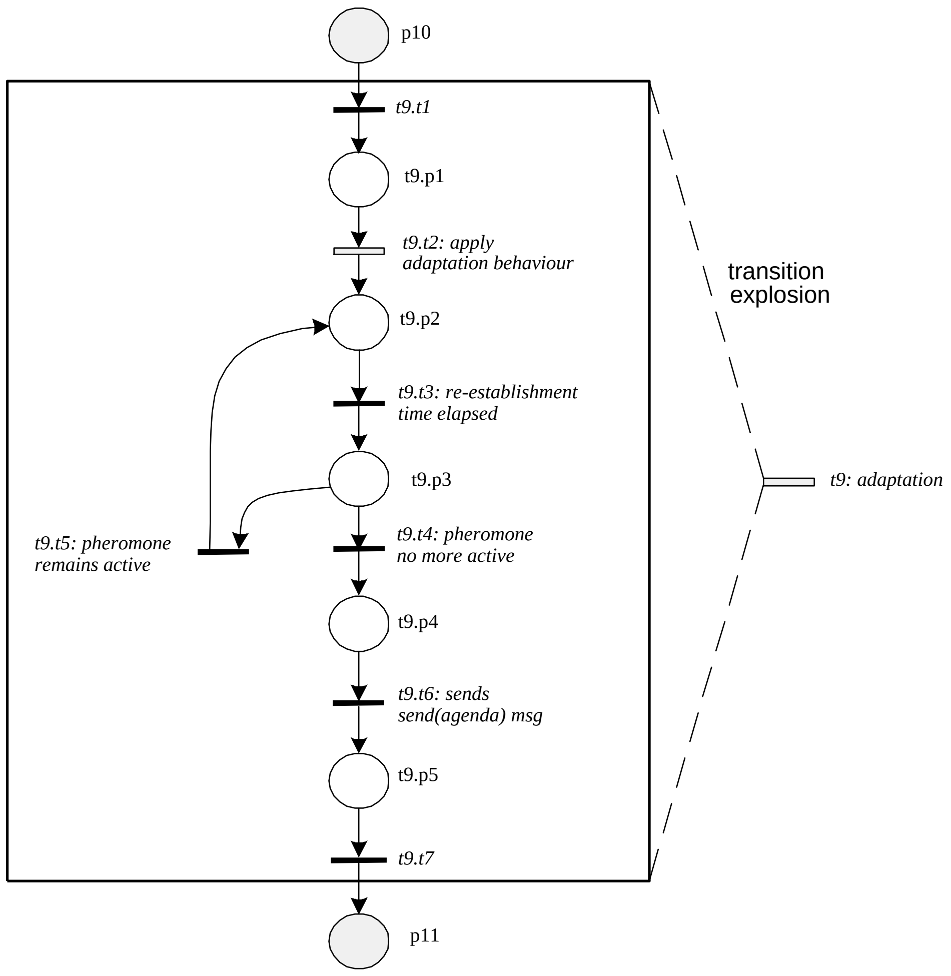

p10
t9.t1
t9.p1
t9.t2: apply adaptation behaviour
t9.p2
t9.t3: re-establishment time elapsed
t9.p3
t9.t4: pheromone no more active
t9.t5: pheromone remains active
t9.p4
t9.t6: sends send(agenda) msg
t9.p5
t9.t7
p11
transition explosion
t9: adaptation

Figure 4.14: Model for the *Adaptation* Activity

After increasing the autonomy factor and self-organising into a heterarchical control structure, the operational holon waits for the elapsing of the reestablishment time, being however able to execute work orders if available. When the reestablishment time is elapsed, the operational holon verifies if the pheromone is already dissipated or remains active. If it remains active, the operational holon stays in the transient state during an additional reestablishment time, until the pheromone be dissipated. When the pheromone is dissipated, the operational holon can return to the previous control structure, reducing its autonomy factor.

During the transient state, the operational holon may have allocated work orders by interacting directly with the task holons. In this way, the operational holon should notify the supervisor holon about its actual schedule, allowing the supervisor to actualise its global schedule and start an schedule optimisation. For this purpose, the operational holon sends the *send(agenda)* message to the supervisor holon, passing to the operational holon a list of work orders allocated.

4.5 Supervisor Holon Behavioural Model

The **supervisor holon**, introduces coordination and global optimisation in decentralised control approaches, coordinating several operational and supervisor holons. In normal operation, the supervisor holon coordinates the activity of the holons under its domain, while when a disturbance occurs, these holons may have to find their way without the help of the supervisor holon.

The supervisor holon shows simultaneously a reactive and pro-active behaviour: reactive, as a server waiting for requests from other holons and pro-active generating optimised schedules that are sent to the lower-level holons, as illustrated in Figure 4.15 [Leitão et al., 2003c]. The supervisor holon functions comprise mainly the elaboration of optimised schedules, the execution of coordination decisions and the aggregation of skills.

4.5.1 Elaboration of Optimised Scheduling

The elaboration of optimised schedules is performed periodically, regulated by the internal clock of the supervisor holon, or to optimse the schedule achieved after the occurrence of a disturbance. The *elaborates optimised schedule* activity executes an optimised schedule of all the work orders, to all the holons under its coordination domain.

The new scheduled work orders and the allocated work orders that were modified due to the new schedule, are proposed by the supervisor holon to the appropriate holons under its coordination domain, as an advice. In case of acceptance, the supervisor holon proposes the calendar to the task holons; otherwise, it re-schedules taking into consideration the rejection.

The *performes a decision* activity, illustrated in the model of Figure 4.16, is related to the

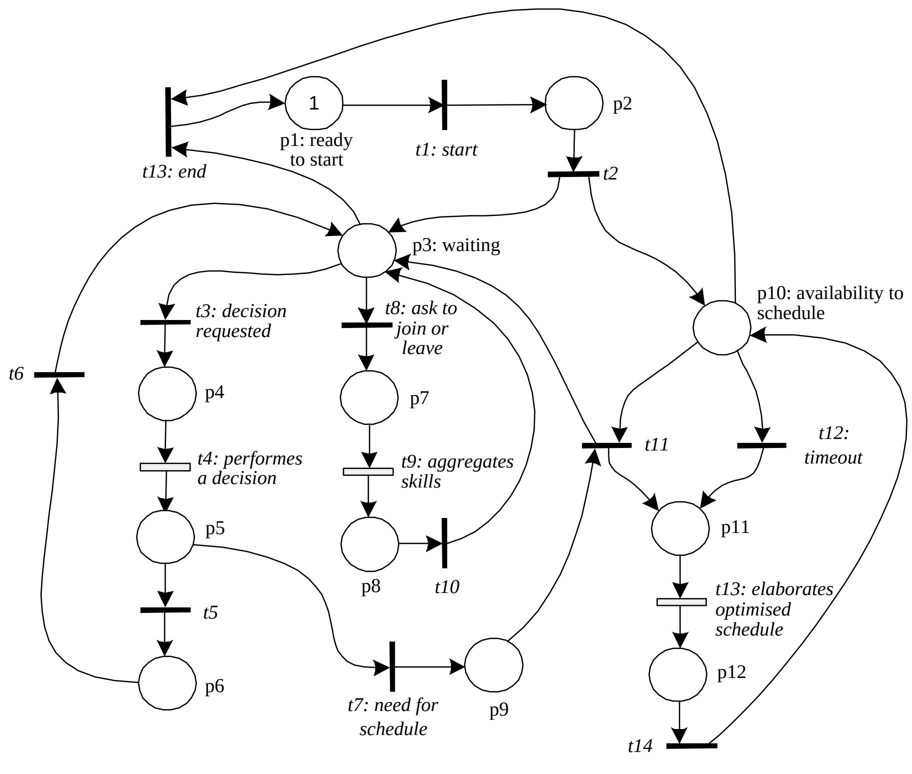


Figure 4.15: Supervisor Holon Behaviour Model

actions associated to monitoring, agenda synchronisation, disturbance handling and work order announcements. Some actions are similar to those presented in the operational holon model, but now presenting more complexity due to the need to handle the aggregated lower-levels holons knowledge and skills.

After a transient state, the supervisor holon receives the agenda of lower-level holons, in order to actualise its agenda. After actualising the agenda, the supervisor holon executes a re-scheduling, optimising the global schedule.

When a supervisor holon receives a *reject(work order)* message from the task holon, it removes the work order from its agenda and sends a *cancel(work order)* message to the appropriate operational holon.

4.5.2 Aggregation of Skills

As the supervisor holon co-ordinates several holons, it manages the group of holons under its coordination domain, aggregating the skills and capacity of the operational holons, when they want to join the group by sending an *ask-join(skills)* message.

At the coordination level, the supervisor holon aggregates the skills by making the union of the several individual member skills. As illustrated in the Figure 4.17, the aggregation of

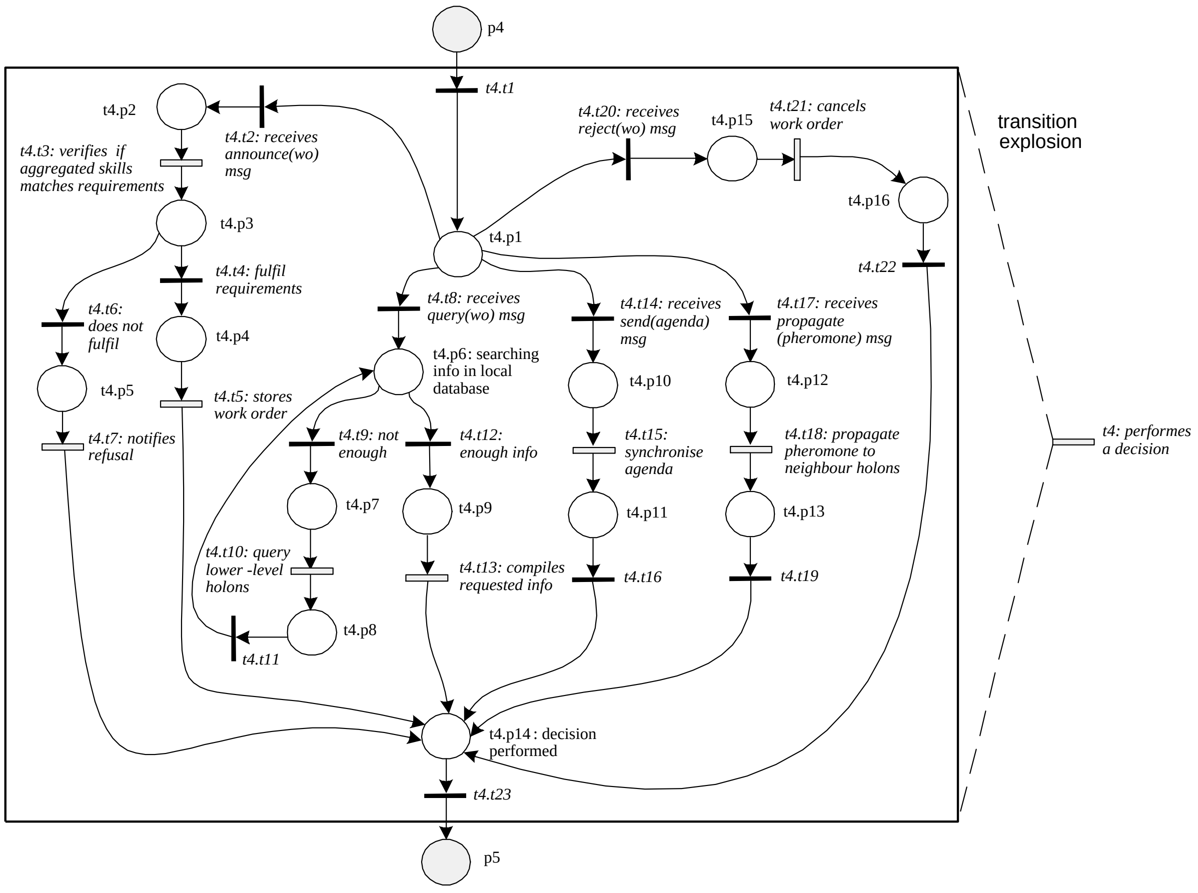

p4
t4.t1
t4.p2
t4.t2: receives announce(wo) msg
t4.t3: verifies if aggregated skills matches requirements
t4.p3
t4.t4: fulfil requirements
t4.p4
t4.t5: stores work order
t4.t6: does not fulfil
t4.p5
t4.t7: notifies refusal
t4.p1
t4.t8: receives query(wo) msg
t4.p6: searching info in local database
t4.t9: not enough
t4.t12: enough info
t4.p7
t4.p9
t4.t10: query lower -level holons
t4.p8
t4.t11
t4.t13: compiles requested info
t4.t14: receives send(agenda) msg
t4.p10
t4.t15: synchronise agenda
t4.p11
t4.t16
t4.t17: receives propagate (pheromone) msg
t4.p12
t4.t18: propagate pheromone to neighbour holons
t4.p13
t4.t19
t4.t20: receives reject(wo) msg
t4.p15
t4.t21: cancels work order
t4.p16
t4.t22
t4.p14 : decision performed
t4.t23
p5
transition explosion
t4: performes a decision

Figure 4.16: Model for the *Performes a Decision* Activity

skills allows to build a new entity that has new skills and synergies. In this case, grouping a robot device that has movement facilities with grippers that can handle pieces of types A and B, transforms the three individual entities in one larger entity that can handle the pieces of types A and B.

In the same way, grouping the previous group with a machine-tool that can execute turning operations, builds a new entity capable to execute turning operations with automatic handling.

When an operational holon wants to exit the group, it sends an *ask-leave()* message.

4.6 Synchronisation of Petri net Models

The interaction between the ADACOR holons is performed by synchronising the individual Petri net models, using additional places that synchronise the evolution of each individual model. The synchronisation of the Petri net models can be extracted from the sequence of interactions during each type of conversation, described in the chapter 3 and modelled using AUML interaction diagrams.

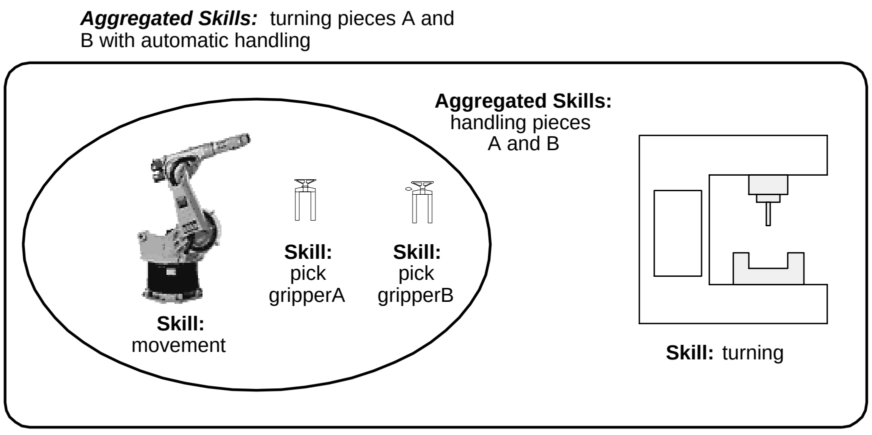

Aggregated Skills: turning pieces A and B with automatic handling
Aggregated Skills: handling pieces A and B
Skill: pick gripperA
Skill: pick gripperB
Skill: movement
Skill: turning

Figure 4.17: Skills Aggregation due to a Group Formation

The synchronisation of Petri net models is illustrated in two distinct cases: interaction between two different ADACOR holon classes and interaction between two holons using the same holon model.

The first example is related to the synchronisation between the product holon and the task holon, Figure 4.18. In the first moment of the interaction, which corresponds to the launching of the task holon, the synchronisation is done by the shared place pa_1, which is marked when the product holon launches the task holon to manage the execution of the production order. The existence of a token in this shared place represents the indication that the task holon can start its execution.

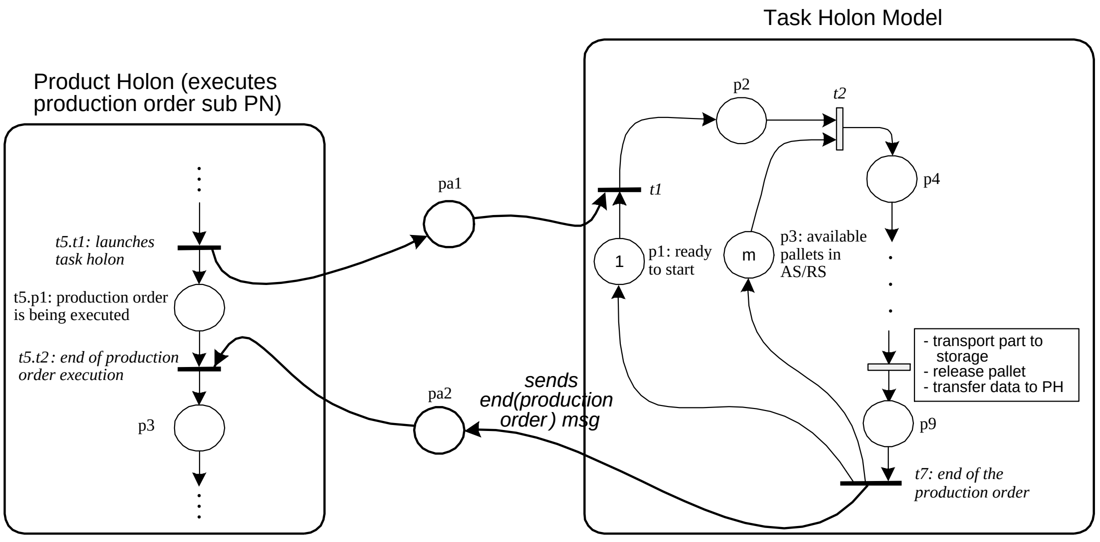

Task Holon Model
Product Holon (executes production order sub PN)
t5.t1: launches task holon
t5.p1: production order is being executed
t5.t2: end of production order execution
p3
pa1
pa2
sends end(production order) msg
t1
p1: ready to start
1
p2
t2
p3: available pallets in AS/RS
m
p4
- transport part to storage
- release pallet
- transfer data to PH
p9
t7: end of the production order

Figure 4.18: Synchronisation of Individual Models of Product and Task Holons

The second moment of the interaction, which corresponds to the end of execution of the production order, the synchronisation is performed by the shared pa_2, which is marked when the task holon finishes the execution of the production order. A token in this place corresponds to the arrival of an *end(production order)* message from the task holon.

The simulation of this coordination model allows to verify some important production parameters, such as the WIP and productivity. As an example, the analysis of the evolution of the number of tokens in the place $t_5.p_1$ of the *executes production order* sub-net, allows to determine the number of instances of the same product present simultaneously in the factory plant.

Another example is the synchronisation between a producer and a mover operational holons that interact during the loading/unloading of the machine.

In terms of synchronisation of individual models, the mover and producer holons use the same operational holon Petri net model, as illustrated in Figure 4.19. The producer holon requests physical coordination by marking the place pb_1 (which corresponds to send a *request(wo)* message), that will enable the transition t_{16} in the operational holon model, entering in a state waiting for the response of the mover operational holon (agree or refuse of the work order execution).

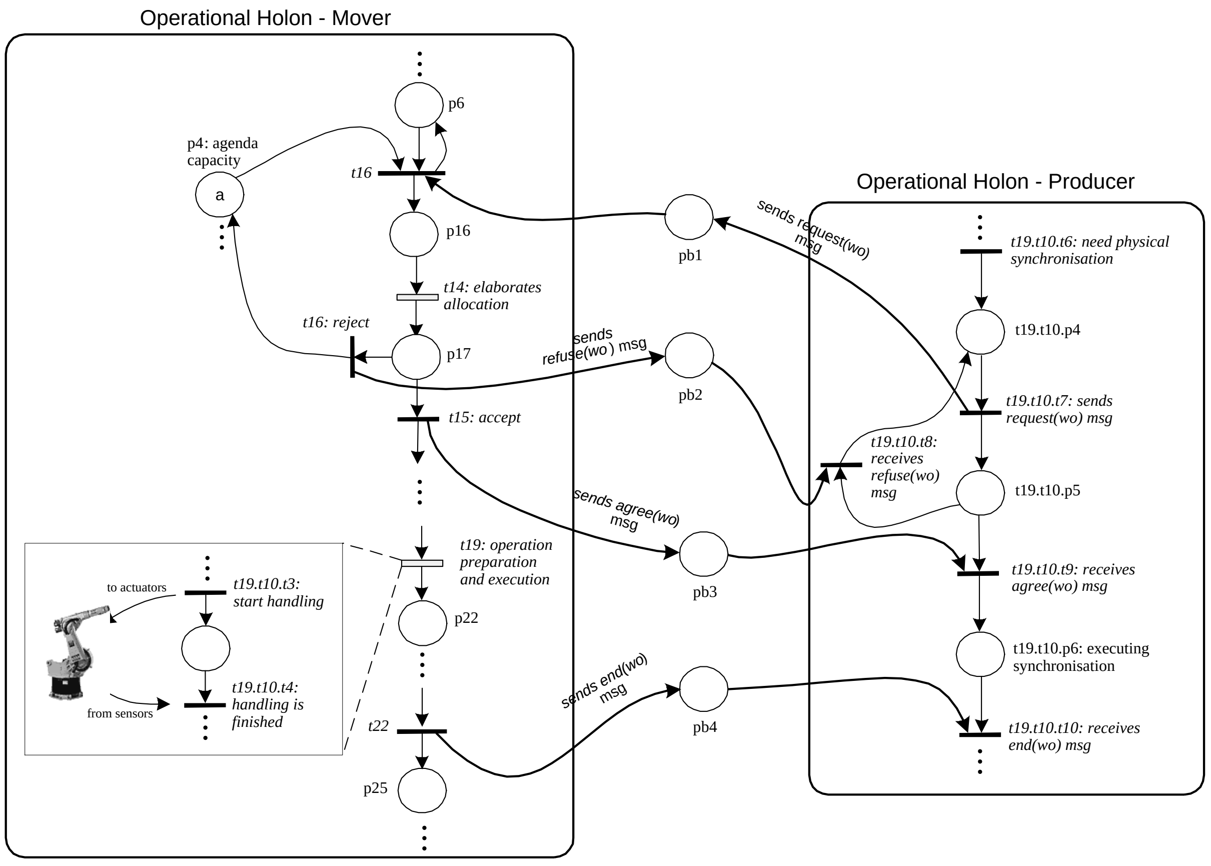

Operational Holon - Mover
p6
p4: agenda capacity
a
t16
p16
t14: elaborates allocation
t16: reject
p17
t15: accept
t19: operation preparation and execution
p22
t22
p25
to actuators
t19.t10.t3: start handling
from sensors
t19.t10.t4: handling is finished
sends request(wo) msg
pb1
sends refuse(wo) msg
pb2
sends agree(wo) msg
pb3
sends end(wo) msg
pb4
Operational Holon - Producer
t19.t10.t6: need physical synchronisation
t19.t10.p4
t19.t10.t7: sends request(wo) msg
t19.t10.t8: receives refuse(wo) msg
t19.t10.p5
t19.t10.t9: receives agree(wo) msg
t19.t10.p6: executing synchronisation
t19.t10.t10: receives end(wo) msg

Figure 4.19: Synchronisation of Individual Models of Producer and Mover Holons

At the end of the auxiliary operation execution, the mover holon notifies the fact by marking the place pb_4 (which corresponds to send an *end(wo)* message) that will enable the transition $t_{19}.t_{10}.t_{10}$ in the producer holon model.

The same methodology is applied to all the other interactions.

4.7 Formal Validation of ADACOR Models

The use of Petri net models with their strong mathematical foundation (functional analysis and linear algebra) allows the verification of important qualitative parameters, such as boundedness of resources, reversibility, deadlock-freeness, conservativeness of resources and mutual exclusion relationships. The use of temporised Petri net, with the association of the time parameter to the transitions, allows additionally to obtain quantitative performance indicators related to the time evaluation of the network.

Together, these analysis help to validate the behavioural models of ADACOR holons, verifying the correctness of the models and verifying if the models fulfils the specifications of the control system [Leitão et al., 2003b].

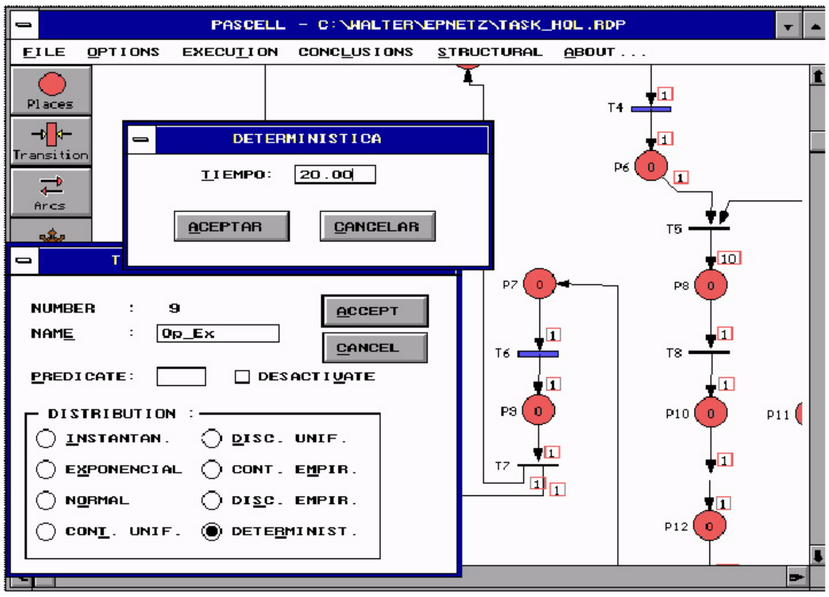
4.7.1 Edition of ADACOR Holon Petri net-Models

The edition, simulation and validation of ADACOR-holons Petri net models will be done using the PASCELL software tool [Colombo, 1998, Colombo et al., 1997]. This software tool, developed in C++ programming language, allows the edition, simulation, and analysis of generalised and temporised Petri Net models. The qualitative analysis is based on discrete-event simulation and structural analysis of the matrix representation of the net. The quantitative analysis is performed by means of the simulation of the temporised Petri net models.

The arcs in the Petri net-models have associate weights: some have the unitary weight and others have the weight associated to the number of instances presented in the system, such as the number of products, resources, pallets, and operations.

The time parameter associated to each transition can be programmed to a fixed value or to have a specific probabilistic law. A screen shot with the time assignment to the transitions is illustrated in the Figure 4.20. In this case, the transition t_9 has associated a deterministic time parameter with value 20 time-units.

Figure 4.21 illustrates the edition of the Petri net-models for the four ADACOR holon classes described in the previous sections, to support the qualitative and quantitative analysis, and the subsequent formal validation of the models, described in this chapter.


PASCELL - C:\WALTER\EPNETZ\TASK_HOL.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
ACEPTAR
CANCELAR
ACCEPT
CANCEL
T4
P6
T5
P7
P8
T6
T8
P9
P10
P11
T7
P12

DETERMINISTICA

TIEMPO: 20.00

NUMBER : 9

NAME : Op_Ex

PREDICATE: ☐ DESACTIVATE

DISTRIBUTION :

○ INSTANTAN. ○ DISC. UNIF.

○ EXPONENCIAL ○ CONT. EMPIR.

○ NORMAL ○ DISC. EMPIR.

○ CONT. UNIF. [✓ ● DETERMINIST.]

Figure 4.20: Assignment of Time Parameter to the Transitions

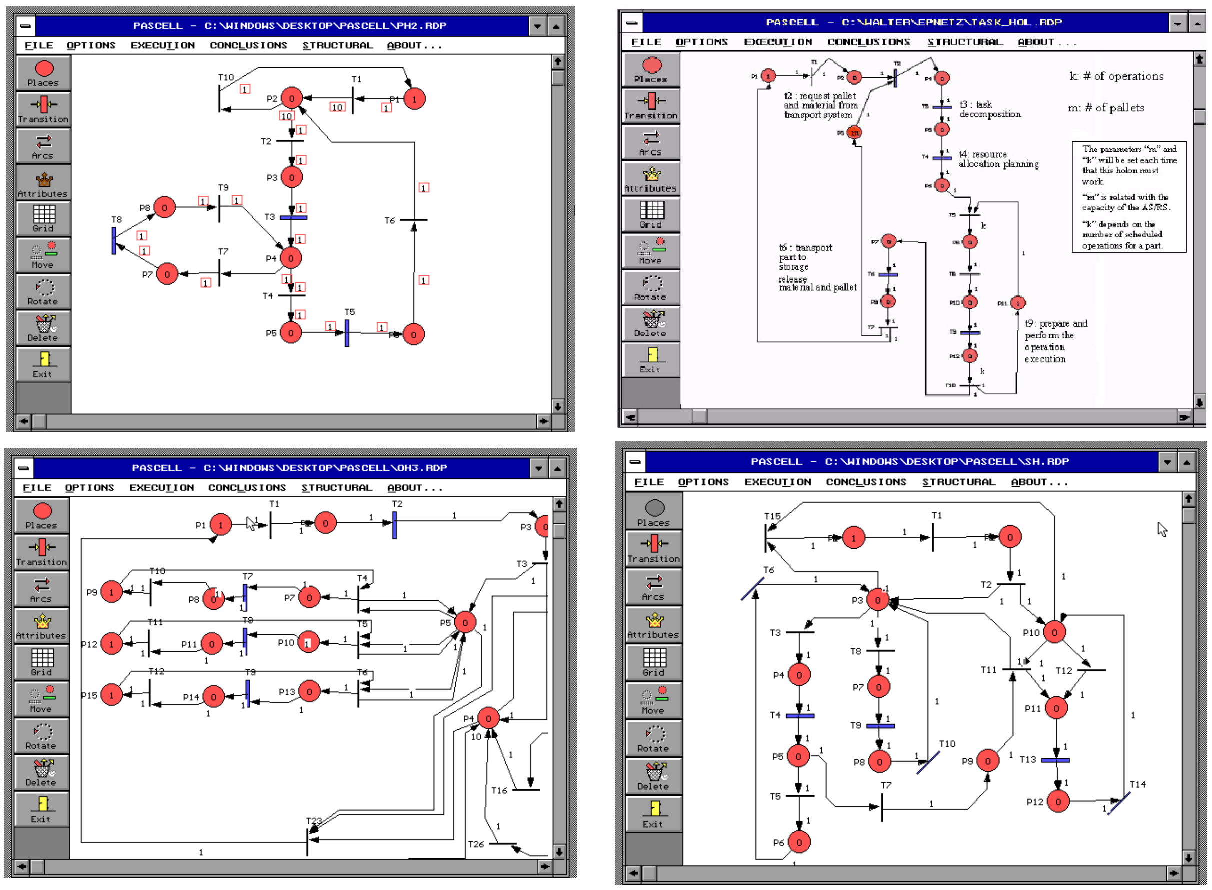

PASCELL - C:\WINDOWS\DESKTOP\PASCELL\PH2.RDP
PASCELL - C:\WALTER\EPNETZ\TASK_HOL.RDP
PASCELL - C:\WINDOWS\DESKTOP\PASCELL\OH3.RDP
PASCELL - C:\WINDOWS\DESKTOP\PASCELL\SH.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
k: # of operations
m: # of pallets
t2 : request pallet and material from transport system
t3 : task decomposition
t4: resource allocation planning
t6 : transport part to storage
release material and pallet
t9: prepare and perform the operation execution
The parameters "m" and "k" will be set each time that this holon must work.
"m" is related with the capacity of the AS/RS.
"k" depends on the number of scheduled operations for a part.

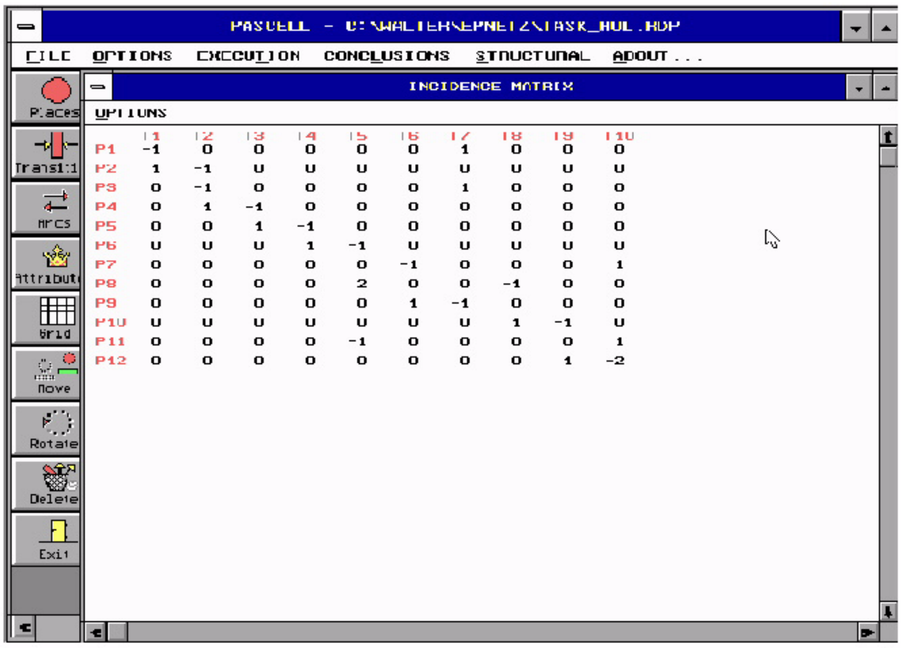
Figure 4.21: Edition of the ADACOR Holon Models

4.7.2 Qualitative Analysis

After the edition of the Petri net models, the next phase is related to the structural analysis of the Petri net models.

The qualitative analysis, based on the structural analysis of the models, allows the verification of the structural and behavioural properties of the model, extracting conclusions about the operation of the system, such as the existence of deadlocks, the bounded capacity of resources, and the existence of structural and behavioural conflicts in the system [Feldmann et al., 1996]. For the purposes of this work, the following properties will be examined: boundedness, reversibility, liveness and conservativeness. More details about the basic concepts of qualitative analysis can be found in Appendix A.

In this work, the behavioural properties are analysed using linear algebra methods, provided by the PASCELL software tool. The results of the qualitative analysis will be exemplified and illustrated by means of only one holon, namely the task holon, due to the easy understanding that it provides. The validation of the other ADACOR holons are shown in Appendix A.

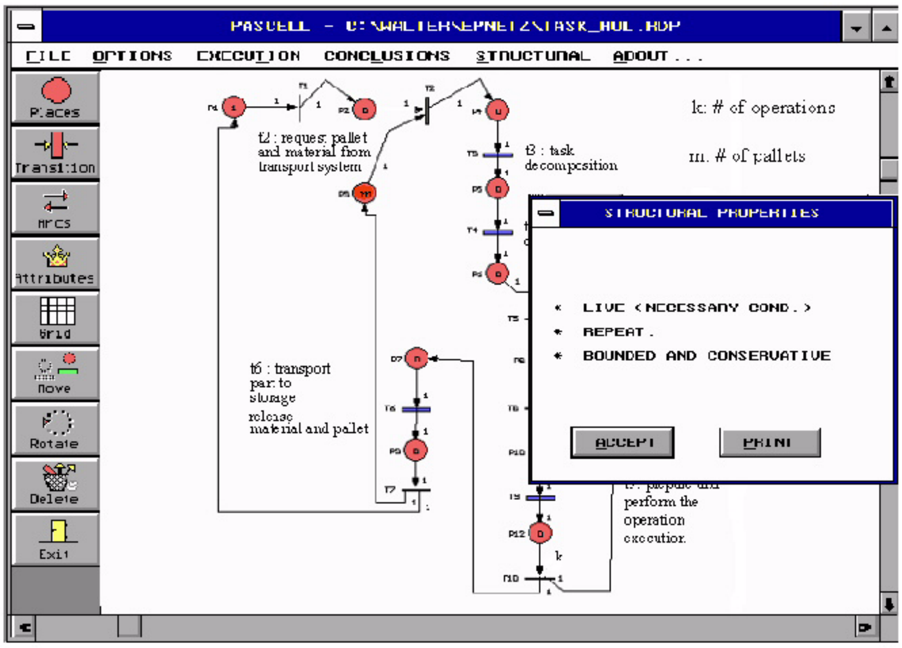


	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
P1	-1	0	0	0	0	0	1	0	0	0
P2	1	-1	0	0	0	0	0	0	0	0
P3	0	-1	0	0	0	0	1	0	0	0
P4	0	1	-1	0	0	0	0	0	0	0
P5	0	0	1	-1	0	0	0	0	0	0
P6	0	0	0	1	-1	0	0	0	0	0
P7	0	0	0	0	0	-1	0	0	0	1
P8	0	0	0	0	2	0	0	-1	0	0
P9	0	0	0	0	0	1	-1	0	0	0
P10	0	0	0	0	0	0	0	1	-1	0
P11	0	0	0	0	-1	0	0	0	0	1
P12	0	0	0	0	0	0	0	0	1	-2

Figure 4.22: Incidence Matrix for the Task Holon

In the analysis of the task holon model, it was considered that a production order includes two work orders, which will be executed by the transition t_9 of the model illustrated in Figure 4.5. Figure 4.22 illustrates the task holon Petri net, by means of an algebraic form. This matrix, the incidence matrix W, represents the holon dynamic behaviour in a non graphic form. As an example, the firing of transition t_1 will remove a token from place p_1 and put a token in the place p_2.

The results of the structural analysis of the incidence matrix for the task holon model are illustrated in Figure 4.23. This structural analysis shows that the Petri net model is live (satisfying the necessary condition), which guarantees the deadlock freeness, i.e. the execution of a production order is done without stopping in an undetermined intermediate state.


PASCELL - C:\WALTER\EPNET2\TASK_HOL.HDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
t2 : request pallet and material from transport system
t3 : task decomposition
t6 : transport part to storage
release material and pallet
k: # of operations
m: # of pallets
perform the operation execution
STRUCTURAL PROPERTIES
ACCEPT
PRINT

* LIVE (NECESSARY COND.)
* REPEAT.
* BOUNDED AND CONSERVATIVE

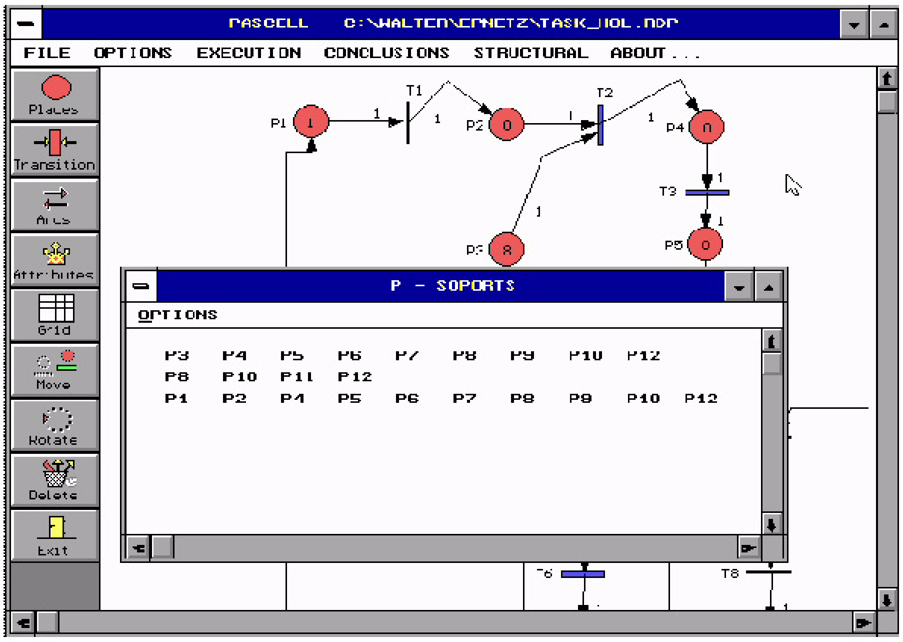
Figure 4.23: Structural Analysis of the Task Holon Model

From the structural analysis, it is also possible to verify that the Petri net model is repetitive, which means that the Petri net returns to the initial state (marking), through well defined work cycles in the execution of production orders.

The Petri net-model is conservative, which means that the pallet tokens do not disappear neither new pallet tokens are created, during the execution of the production order. Additionally, the model is bounded, which means that the number of production orders in the system is limited to the maximum value of m, due to the existence of the monitor place that regulates the available pallets, and the number of work orders in each production order is also limited to the value k (in this case equal to two).

Other characteristics of the Petri net model for the task holon can be found analysing the incidence matrix. Figure 4.24 presents the set of P-supports of the incidence matrix. This set is related to the P-invariants referenced in Appendix A.

The analysis of P-invariants constitution allows confirming mutual exclusion relationships among places and "of course", functions and resources involved in the holon-structure and


PASCELL C:\WALTER\CRNETZ\TASK_HOL.NDR
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
T1
T2
T3
P1
P2
P4
P5
1
T8
P - SOPORTS
OPTIONS

```
P3   P4   P5   P6   P7   P8   P9   P10  P12
P8   P10  P11  P12
P1   P2   P4   P5   P6   P7   P8   P9   P10  P12
```

Figure 4.24: P-invariants for the Incidence Matrix of the Task-Holon Model

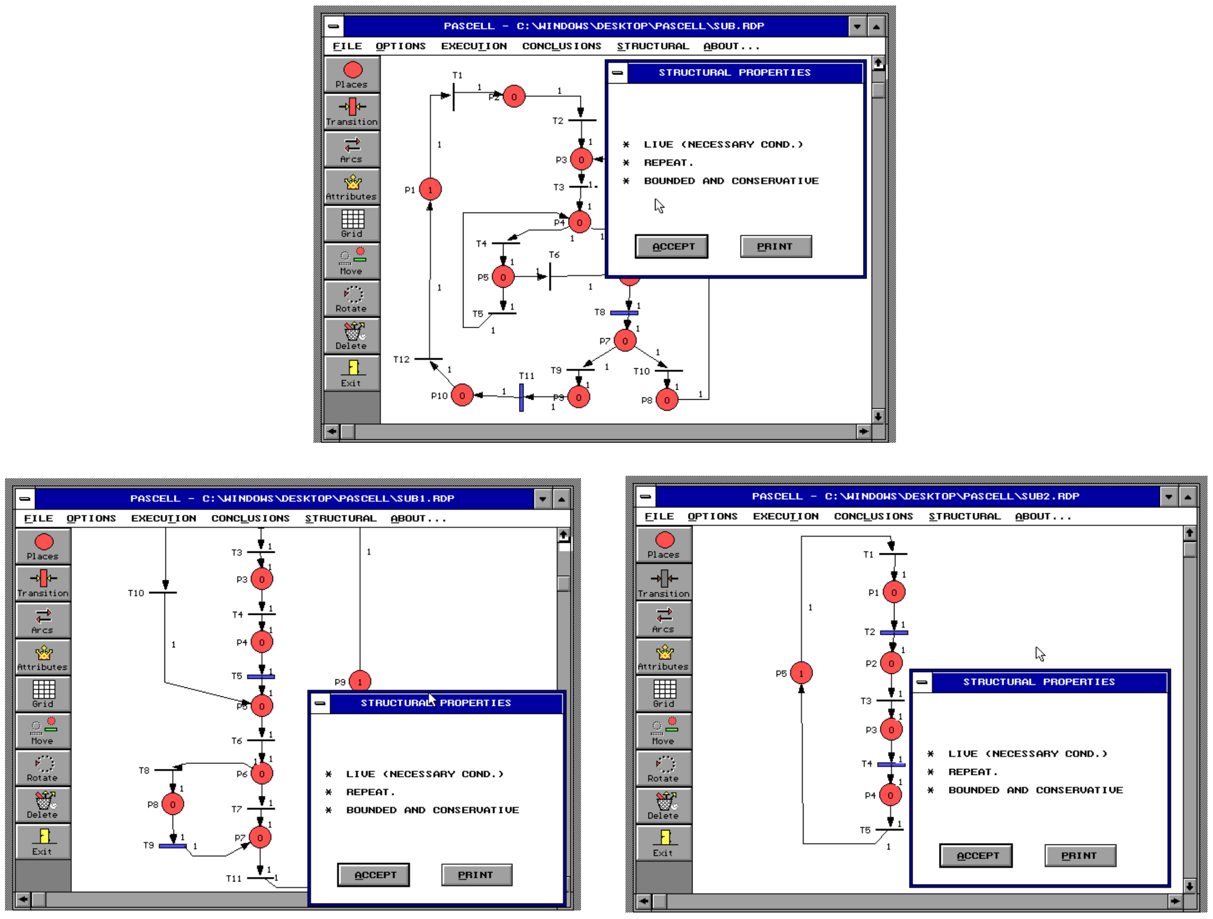
behaviour. For example, analysing the P-invariant $\{p_8, p_{10}, p_{11}, p_{12}\}$, it is possible to confirm that for the same product instance, only one of the places referred in the P-invariant can be marked at any time, i.e. there is a mutual exclusion relationship among those places.

Since each ADACOR holon class was modelled using a top-down approach by the refinement of timed transitions, it is necessary to analyse also the properties of each sub-net block. Analysing the *Elaborates Resource Allocation Plan*, *Supervises Work Order Execution* and *Releases Pallet* sub-nets, it is possible to conclude that each refined sub-net is well-formed since, Figure4.25

- the associated Petri Net $\widehat{PN}$ is live,
- $\widehat{m_o}$ is the only initial marking of $\widehat{PN}$,
- the only transition enabled by $\widehat{m_o}$ is the initial transition (see section 4.1)

In these circumstances, the large task holon Petri net preserve its properties, according to the theorems introduced by [Vallete, 1979].

The qualitative analysis performed by PASCELL, without simulation and only by means of linear algebra and functional analysis, guarantees that the model is correct from the functional analysis point of view, and it can be seen as a virtual representation of the holon. In this case, the next step is to use the same models, but now with the time parameter associated to the transitions, to perform a quantitative analysis of the holon's behaviour.


PASCELL - C:\WINDOWS\DESKTOP\PASCELL\SUB.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
ACCEPT
PRINT
PASCELL - C:\WINDOWS\DESKTOP\PASCELL\SUB1.RDP
PASCELL - C:\WINDOWS\DESKTOP\PASCELL\SUB2.RDP

STRUCTURAL PROPERTIES

* LIVE (NECESSARY COND.)
* REPEAT.
* BOUNDED AND CONSERVATIVE

STRUCTURAL PROPERTIES

* LIVE (NECESSARY COND.)
* REPEAT.
* BOUNDED AND CONSERVATIVE

STRUCTURAL PROPERTIES

* LIVE (NECESSARY COND.)
* REPEAT.
* BOUNDED AND CONSERVATIVE

Figure 4.25: Structural Analysis of the sub-Petri nets of the Task Holon Model

4.7.3 Quantitative Analysis

As in the qualitative analysis, only the quantitative analysis for the task holon will be described. The quantitative analysis[5] requires the introduction of the time parameter associated to the transitions. For this purpose, deterministic distribution times will be used, according to the Table 4.1.

The transitions t_2 to t_4 are computational activities, namely, the pallet request, the task decomposition, and the resource allocation planning, being estimated at 1 second per transition.

The transition t_6 includes the transportation of the part to the storage system and release of the pallet, after the execution of all the work orders. The transfer of relevant information about the product execution to the product holon is also included, and the time associated to this transition is dependent of the actual capacity of the transport system, but for the study is was estimated in 10 seconds.

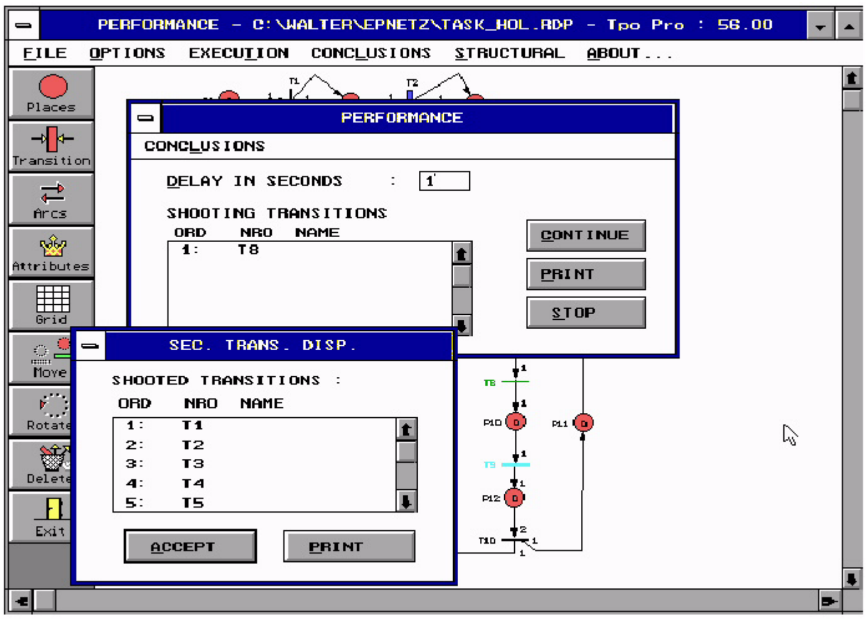
[5]Also designated by performance analysis.

Table 4.1: Time Associated to the Transitions

Transition	**Time Units**
t_2	1
t_3	1
t_4	1
t_6	10
t_9	20

The transition t_9 represents the activity related to the preparation of the work order execution and the physical execution of the work order. The preparation phase involves the transportation of the part to the machine where the part will be processed, and the execution of set-ups in the machine, if necessary. For this study it was considered that this activity takes 20 seconds.

The other transitions are instantaneous (atomic-firing).


PERFORMANCE - C:\WALTER\EPNETZ\TASK_HOL.RDP - Tpo Pro : 56.00

PERFORMANCE

CONCLUSIONS

DELAY IN SECONDS : 1

SHOOTING TRANSITIONS

```
ORD   NRO   NAME
 1:    T8
```

SEC. TRANS. DISP.

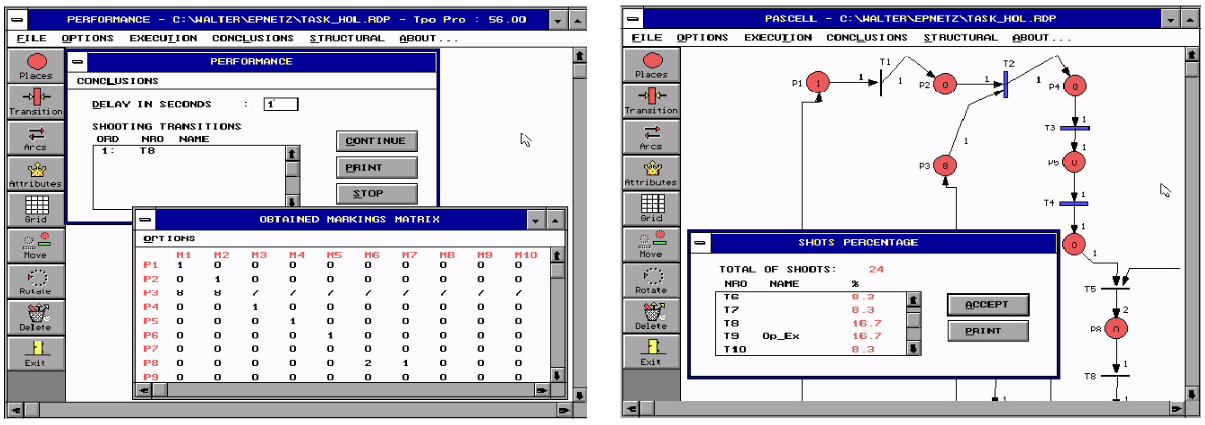
SHOOTED TRANSITIONS :

```
ORD   NRO   NAME
 1:    T1
 2:    T2
 3:    T3
 4:    T4
 5:    T5
```

Figure 4.26: Evolution of the Firing Process

Figure 4.26 shows the evolution of the net after 56 time-units: the transition t_8 is firing (atomic-firing) and the special temporised transition t_9 is evolving within its second phase (consuming the time units that model the activity).

The coverability/reachability graph that shows the states reached by the model during 56 time-units is illustrated in Figure 4.27. This screen shot, together with the percentage of transition shoots, illustrated in the same Figure 4.27, allows to perform a very precise analysis of the evolution of the holon behaviour, answering to pertinent questions, such as the states that have been reached, the activities that have been performed and their relative frequency, and the history of the evolution of the holon. As an example, the shots percentage screen show that the transition t_9 has been shotted 16,7% and the transition t_7 8,3%, which means that to execute a production order, the transition associated to the release of a pallet is fired half of the times of the transition associated to the execution of a work order. This is correct, because we have considered at beginning that a production order comprises two work orders.



DELAY IN SECONDS : 1

SHOOTING TRANSITIONS

ORD	NRO	NAME
1:	T8	

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
P1	1	0	0	0	0	0	0	0	0	0
P2	0	1	0	0	0	0	0	0	0	0
P3	8	8	7	7	7	7	7	7	7	7
P4	0	0	1	0	0	0	0	0	0	0
P5	0	0	0	1	0	0	0	0	0	0
P6	0	0	0	0	1	0	0	0	0	0
P7	0	0	0	0	0	0	0	0	0	0
P8	0	0	0	0	0	2	1	0	0	0
P9	0	0	0	0	0	0	0	0	0	0

TOTAL OF SHOOTS: 24

NRO	NAME	%
T6		8.3
T7		8.3
T8		16.7
T9	Op_Ex	16.7
T10		8.3

Figure 4.27: States Reached and Percentage of Transition Shoots after 56 Time Units

All the information about the timed evolution of the holon can be summarised with a Gantt diagram, which is shown in Figure 4.28 and reflects the temporal sequence of the system operation.

In the Gantt diagram it is possible to verify the temporal sequence in the evolution of the dynamic behaviour of the task holon. For example, the transition t_9 shows the execution of the two work orders that belong to the production order, which takes 40 seconds. After the execution of all work orders, in this case two work orders, it is executed the transport of the part to the storage system, represented by the transition t_6, that takes 10 seconds.

The procedure for qualitative and quantitative analysis, described previously for the task holon, has been repeated for the others ADACOR holon Petri net-models, allowing the validation of the specifications (structural and behavioural) of the ADACOR holon classes.

The previous validation allowed to validate the correctness and the specifications of each individual Petri net model. The performance analysis of the whole system requires the application of the same methodology to a large Petri net, built upon the individual Petri nets, dependent of the number of the system elements, which is not feasible in practice. For this purpose, it will

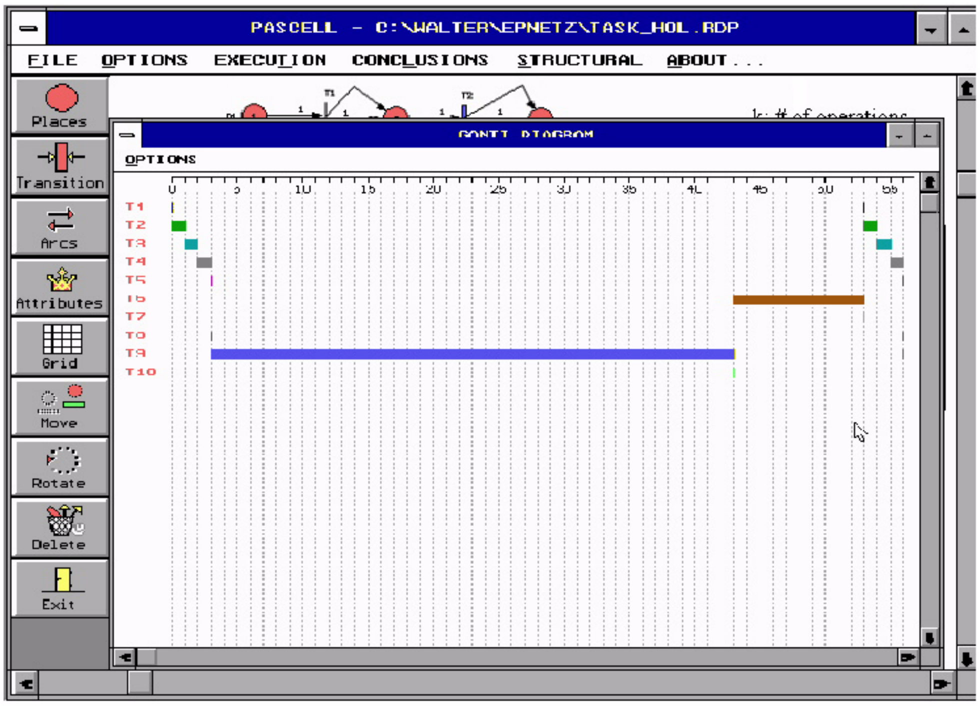


Figure 4.28: Gantt Diagram for the Performance Analysis

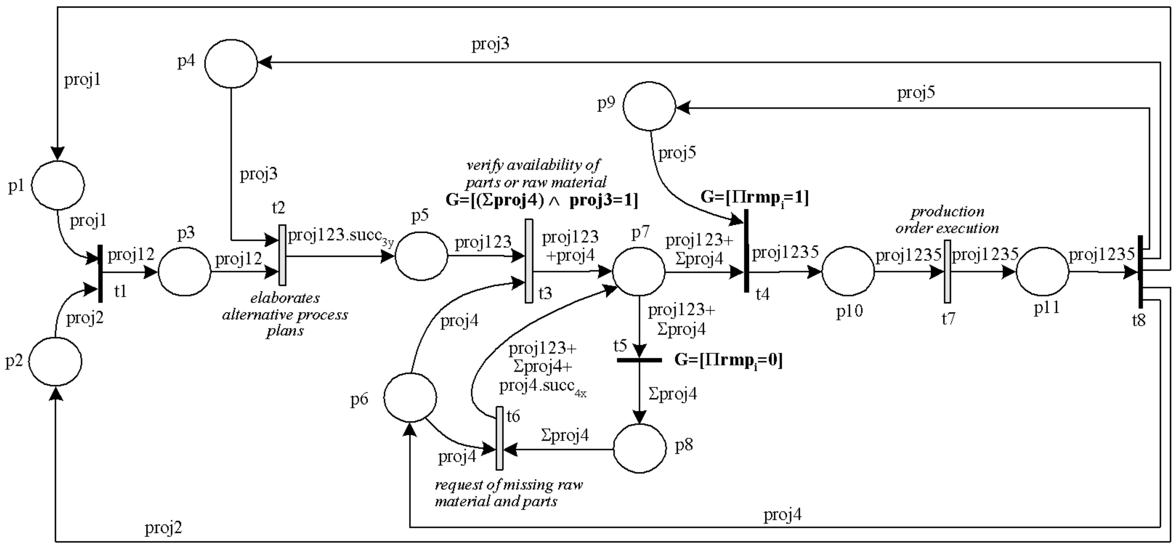
be used the experimental validation that will be described in the chapter 6.

4.8 Advanced Modelling using High-Level Petri Nets

In industrial manufacturing applications, the Petri net models may become very complex and difficult to handle, due to a set of weak points in using the proposed PN-formalism and other similar extensions of this tool [Vyatkin et al., 2001, Frey et al., 2002, Leitão et al., 2003d]. This is particularly true when the system presents many instances of the same component (e.g., different resources), being the model increased, in terms of structure and components, in a complex manner.

The use of high-level Petri net allows to reduce this complexity, by compressing the representation of states, actions and events, and to support the modelling of more complex and bigger systems [Colombo et al., 2001, Feldmann et al., 1996, Holloway et al., 1997]. Coloured Petri Nets (CPN), a type of high-level Petri net, are mathematical-graphical oriented formalisms for design, specification and validation of concurrent systems [Jensen, 1992, Zhou et al., 1999]. CPN have got their name because they allow to use tokens that carry data values and can be distinguished from each other, in contrast to the tokens of low-level Petri nets, which by convention are drawn as black-dots.

As an example of the application of CPN to model the ADACOR holon classes, the Appendix C shows the modelling of the product holon class. This model, illustrated in the Figure 4.29, allows to represent simultaneously all products available in the factory and their instances [Leitão et al., 2003d].



$M_0(p1)=\{\Sigma_{[i:1\text{-}n]}ph_i\}$; $M_0(p2)=\{\Sigma_{[i:1\text{-}m]}or_i\}$; $M_0(p4)=\{\Sigma_{[i:1\text{-}l]}wp_i\}$; $M_0(p6)=\{\kappa.\Sigma_{[i:1\text{-}q]}rmp_i\}$, $\kappa \in N$; $M_0(p9)=\{\Sigma_{[i:1\text{-}p]}th_i\}$

Figure 4.29: Coloured Petri Net Model for the Product Holon

In this case, each place of the model can contain a set of coloured tokens carrying a data value. The arcs have an attached function (expression), which describes how the state of the CPN changes when the transitions are fired. Guards are associated to the transitions and represent restrictions to the type of data value, i.e., coloured marks, that a transition can move during its firing. When a transition fires, the tokens must take values that match the arc expressions, and they must belong to a type that match also the guard associated to the transition.

In spite of its advantages to compress several instances in the same model, the analysis of CPN models is less understandable, and the structural and behavioural validation of these models is more complex than for low-level Petri nets formalism.

4.9 Summary

The formal modelling of the structural and behavioural specifications of holonic control systems assumes a critical aspect, having little attention devoted to it in the holonic manufacturing research community.

In this chapter, the dynamic behaviour of each ADACOR holon class is formally modelled, using a kind of Petri Net formalism tailored for production control modelling purposes. The individual model of each ADACOR holon class is built using special temporised transitions to model the execution of activities, that can be exploded into a more detailed and refined level, allowing the modelling in a top-down approach.

The edition, simulation and formal validation of the structural and behavioural specifications of the ADACOR holons, are also presented in this chapter, illustrated by the description of the formal validation of the task holon model. This formal validation comprises the verification of some structural properties of the model, such as the liveness and boundedness, and the execution of performance analysis.

The Petri net formalism used to model the ADACOR holons behaviour has a set of weak points, specially when the system presents many instances of the same component. A brief introduction of the application of high-level Petri net to model the dynamic behaviour of ADACOR holons, namely the ADACOR product holons, was presented.

In the next chapter, the mechanisms related to the disturbance management that supports the agile adaptation to unexpected disturbances, will be described. These mechanisms complement the behaviour models presented in this chapter, and are based in the ADACOR concepts, previously introduced.

Chapter 5

ADACOR Disturbance Management

"Good science is the ability to look at things in a new way and achieve an understanding that you didn't have before."
Hans Kornberg

The industrial manufacturing systems are dynamic environments, with new jobs arriving continuously to the system, and certain resources becoming unavailable and additional resources introduced at random times. The occurrence of unexpected disturbances leads to deviations from the initial and optimised plans and degrades the performance of the system. In these circumstances, the manufacturing control system should react quickly to unexpected disturbances, adapting the schedule plans as fast as possible, avoiding the risk of degradation of the production productivity.

The dynamic and agile reaction to disturbances is of critical importance in the development of the new generation of manufacturing control systems, being dependent of the system capability to adapt its control structure to the available resources.

Disturbance management scope includes not only the reaction to unexpected disturbances, but also the detection, identification and reaction to disturbances, as illustrated in Figure 5.1. In disturbance management, the decision-making mechanisms influence strongly the response of the system to the disturbance.

The first step is the detection of the disturbance, followed by its identification, classification and characterisation, using data and other information from other disturbances that occurred previously. In the presence of a disturbance, the system should react minimising the effects of the disturbance.

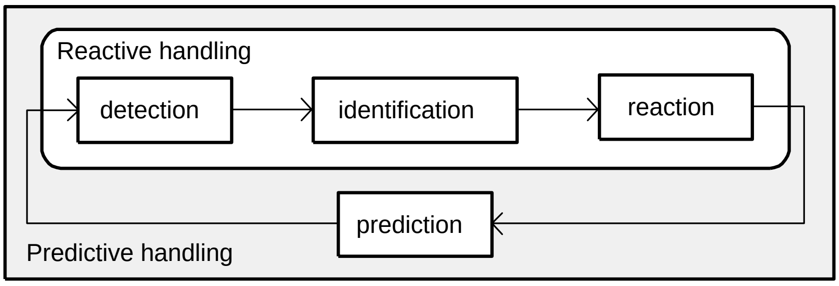


Figure 5.1: Steps in ADACOR Disturbance Management

Traditionally, disturbance management is reactive, i.e. takes corrective actions when the disturbance occurs. However, in order to achieve better performance, it is necessary to consider some prediction capability, closing the process with a feedback loop, as illustrated in Figure 5.1. A learning mechanism allow the system to learn from the past occurrence of disturbances, improving its performance in future reaction to disturbances, or even to decide to classify some disturbance occurrence patterns as normal behaviour in future production plans.

In this chapter, the four steps associated to the disturbance management, i.e. the detection of symptoms, identification of disturbances, mechanisms for reaction to disturbances and prediction of future unforeseen disturbances, will be discussed. Special attention is devoted to the prediction of occurrence of future disturbances, which could minimise the effects of the disturbance.

5.1 Detection of Symptoms

The detection and identification of disturbances is a major requirement of holonic manufacturing control systems.

The first step in disturbance management is the discovery or detection of symptoms that can lead to an existing disturbance. A symptom can be an work order delay, a rush order, a quality problem, but also an unexpected value in a sensor, such as high temperature in a component or abnormal in a cutting tool.

The detection of symptoms in ADACOR architecture uses passive monitoring and active notification mechanisms, which were described in previous chapters. The active notification is a rule-base mechanism implemented in each component to give the possibility of notifying the entities that have subscribed a specific event, when the event occurs. A holon that wants to monitor in an active form the occurrence of a particular event should subscribe the desired event. When the event or symptom occurs, the target holon checks the subscription list of events and sends the notification of occurrence of the symptom to all the holons that subscribed the service.

These mechanisms can be used to detect failure symptoms in the physical manufacturing machines (which requires the implementation of event notification features in the virtual resource that represents the real manufacturing resource), work order delays inside the factory and in deliver of materials by suppliers (which requires the subscription of active notification in an inter-enterprise platform) and problems with production quality parameters (which requires active notification by the quality management system).

The detection of symptoms using active notification is not possible if the available platforms do not provide active notification mechanisms. Passive monitoring can be used as an alternative to active monitoring, to overcome this problem. Passive monitoring does not involve any event driven mechanism, being the initiative of the entity that needs some specific information to request the required information (read the value of a sensor, for instance).

5.2 Identification of Disturbances

The detection of symptoms does not lead directly to the occurrence of a disturbance, being necessary to isolate the symptoms, to make a clear diagnosis of the detected symptoms and to identify the disturbance. Also in this phase it is important to analyse the potential impact of the identified disturbance in the system.

5.2.1 Characterisation of Disturbances

The characterisation of disturbances is based on a model for the generic characteristics of the disturbances, that defines the main attributes that will ensure the correct data collection. The information related to the occurrence of one disturbance is stored in a data structure, containing the code of the disturbance (a unique code for each disturbance), the disturbance type, the date of occurrence, the time spent to recover from the disturbance and the solution used to recover from the disturbance. Each disturbance type may have associated a list of symptoms a list of possible solutions to be applied to recover from the disturbance.

From the historical data, it is possible to measure the statistical significance of each type of disturbance in terms of frequency mean (number of times that the disturbance has occurred per time unit), downtime (mean value of the duration of recovery time) and the potential cost to the company.

5.2.2 Rule-Base Disturbance Detection

The identification of disturbances in ADACOR uses a rule-base mechanism applied to the set of symptoms.

A rule-based system consists of a rule-base, storing the domain-specific rules, and an inference engine that selects applicable rules according to the current facts. The rules are expressed in terms of IF - THEN statements, having the following form:

`Rule:` IF *conditions* THEN *actions*

The *conditions* part of the rule consists of one or more conditions that are matched against the current facts. When all conditions are simultaneously satisfied the rule is triggered. The *actions* part of the rule consists of a set of actions or conclusions that are executed when the rule is triggered.

The inference engine is responsible for this mechanism and for updating the list of applicable rules by adding or removing facts. The inference module continues its cycle, triggering the rules which conditions are satisfied, and executing their selected actions, until no applicable rule remains.

As described previously, each type of disturbance has associated a set of symptoms. The rule-base mechanism matches the detected symptoms with the symptoms associated to each disturbance type, defined in the knowledge base. As an example, the following rules can correlate symptoms and disturbances:

`Rule 1:` IF *wear is high* THEN *the cutting tool is unavailable*

`Rule 2:` IF *air compressed is low* THEN *grippers do not close*

The set of symptoms for each disturbance is continuously updated according to the knowledge acquired in previous disturbance occurrences. When the detected symptoms do not allow to reach any conclusion, the learning mechanisms may be programmed to discover the disturbance type in similar or analogue cases, or to request external intervention to teach the system. In both cases, knowledge is acquired and stored in the knowledge base.

5.2.3 Classification of Disturbances

It is important to analyse all the possible types of disturbances that exist in industrial manufacturing environments at the shop floor level, and in particular the main disturbances that can cause impact at the scheduling and planning level.

The disturbances at shop floor level can be grouped in two classes: internal and external.

Internal Disturbances

At the internal level, the disturbances are related to computational failures, operator errors, machine breakdowns, variability in machine performance (quality and production rate), unavailability of labour, layout re-configuration and delays in the material and information flow during the production. The computational failures, such as the failure to access to a database, the failure to open a file or a failure in the network, are not considered in this study, since in principle would be possible to recover automatically from the errors and they should not cause large impact in the system.

The **machine breakdown** and **unavailability of labour** cause the unavailability of the resources to execute work orders, leading to the decrease of production capacity and normally to the decrease of throughput. A machine failure can occur due a tool collision, a broken tool or a mistake in the machine program, leading to a temporary or longer out of service of the machine, which becomes unable to accomplish the allocated work orders during the downtime.

The reaction to a failure in a physical resource is dependent of the capability of repairing the failure. The non repairable components, such as electric bulbs or PC memories, are replaced in the minimum amount of time. The repairable components are those which is economically satisfactory to repair after the occurrence of a failure, such as a robot or a machine-tool. In the context of this work, only the repairable components will be considered.

A failure in a machine leads to problems at the scheduling and planning level, with secondary disturbances related to the work order delay and layout re-configuration. The objective in this case is to repair the machine and in parallel to find out alternative solutions to reduce the deviation from the initial plan. The effects of a machine failure may also be reflected in the part, which can be destroyed or not, and in the machine itself, which can be physically damaged or ready to continue the service. The state of the part and of the machine will determine the type of reaction to the failure. Figure 5.2 illustrates the machine failure problem, considering a failure in the machine R1 at instance t = 40s, the part being not damaged after the failure and the machine being out-of-service during 10 seconds for the corrective maintenance. Operations T1, B1 and D1 were delayed.

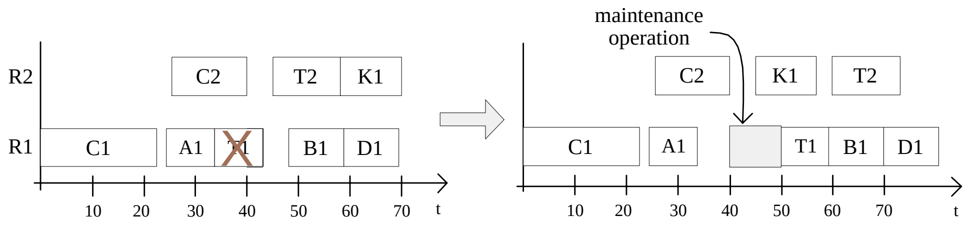


Figure 5.2: Re-scheduling due to a Machine Failure

The **quality inspection** can lead to the detection of parts that not respect the quality requirements of the product, due to operator errors or the variability of machine performance, requiring the need to execute a corrective maintenance intervention in the defective machine. Additionally, the parts that do not obey to the quality requirements may be rejected, being necessary to execute other parts.

The **layout re-configuration** is the re-organisation of the manufacturing resources available in the factory plant, due to the addition of a new resource or the removal of a resource. The addition of a resource causes small impact in the system, because it increases the number of alternative solutions for the execution of production orders. The removal of a resource leads to a more complex problem, since it may introduce conflicts in the system. In this case, the work orders allocated to the unavailable resource should be re-allocated to other available resources, as illustrated in the Figure 5.3, where it is considered that R2 and R3 are alternative resources, and the removal of resource R2 causes the re-scheduling of operations C2, T2 and E2.

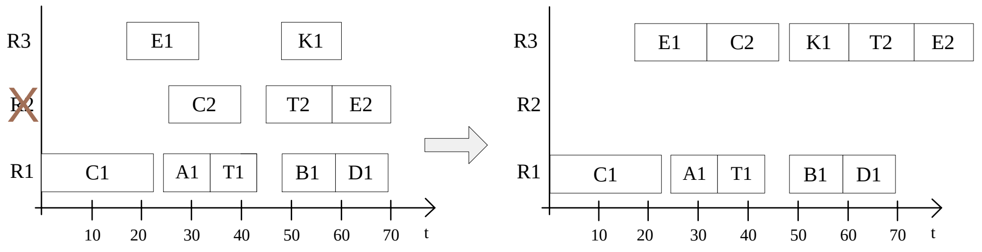


Figure 5.3: Re-scheduling due to the Remotion of a Resource

External Disturbances

At the external level, the disturbances are usually related to delays by suppliers in the delivery of raw materials or semi-finished parts, rush orders, cancellation or changes in existing orders, forecasting errors and demand variations.

The **delay** causes the need to re-schedule, delaying all production orders related to the delayed purchased order, and allowing the re-scheduling of all other production orders, trying to use the gaps open by the delayed orders.

The **cancellation** of a production order or work order may be due for example to a cancellation from the customer or to a failure that provoked the destruction of the part. This disturbance causes small impact in the system, because it is only necessary to release the work orders already allocated and to re-schedule the other work orders in order to optimise the local schedule, respecting the constraints related to the earliest and due dates, as illustrated in Figure 5.4, where operations T1 could be re-scheduled earlier, and so T2 and K1.

The modification of the order attributes, such as the change of temporal window to produce (earliest and due dates), may lead to a more complex problem, requiring the need to re-schedule all work orders.

The **introduction of rush orders** implies re-scheduling, to insert the production order in

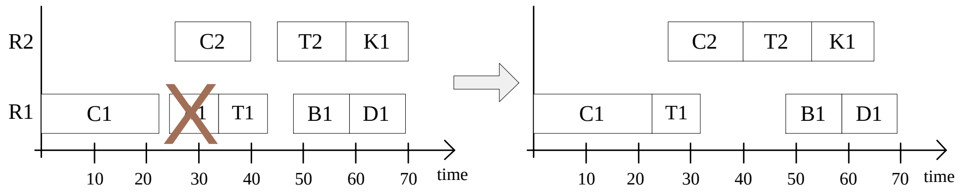


Figure 5.4: Work Order Cancel

the schedule fulfilling its earliest and due dates. This type of disturbance is a problem when it leads to temporal conflicts with other already allocated work orders or when it is a priority work order.

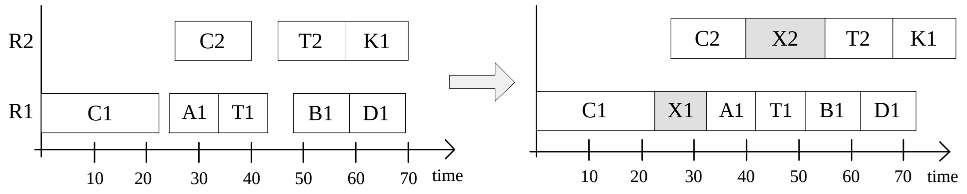


Figure 5.5: Introduction of a new Production Order

Figure 5.5 illustrates the introduction of a new X production order, comprising the work orders X_1 (d_{x1}=10s, to be executed in R1) and X_2 (d_{x2}=15s, to be executed in R2), and due date 60 time units.

5.2.4 Impact of Disturbances in the System

An important issue when analysing a disturbance is the assessment of its potential impact on production performance indicators. The impact of the disturbance is related to the propagation of the disturbance in the system and the associated consequences. Measuring the appropriate performance parameters (according to the production goals), it is possible to obtain an indicator about the impact of the disturbance in the system, and also to compare the impact provoked by different types of disturbances.

The level of impact is dependent of the type of disturbance and the physical and temporal conditions.

In terms of the disturbance type, each event will lead to specific consequences in the system, requiring different actions to be executed. As an example, the addition of a resource has lower impact in the system than the occurrence of a failure in a machine.

The physical location of the disturbance is critical in the definition of the impact, since a disturbance occurred in a critical path (bottleneck) has higher impact than a disturbance occurred in a machine with alternative paths.

At last, the level of the impact of the disturbance is also dependent of the moment in time when a disturbance occurs. As an example, the impact of a failure in a machine is higher if an alternative machine is out-of-service.

5.3 Reaction to Disturbances

Once the disturbance is identifyed, the next step is reacting properly by adapting the system behaviour in accordance to the unexpected disturbance. The reaction mechanisms, dependent of the type of disturbance, are based in the adaptive holonic control concepts defined in ADACOR architecture. The reaction mechanisms for the more important types of disturbances that normally exists at the shop floor level will be described next.

5.3.1 Mechanisms for the Machine Failure

The occurrence of a machine failure must be reflected in the models of the operational and task holons, and their interactions. The behaviour of the reaction mechanism to handle a machine failure, in the operational holon side, is modelled by the Petri net illustrated in Figure 5.6.

Diagnosis

In case of a machine failure, the operational holon executes a self-diagnosis using the rule-oriented mechanisms, to determine the state of the machine and the product after the failure, and to estimate how long the downtime will be. If the diagnostic detects an abnormal situation or does not provide a concrete conclusion, the operational holon requires an external maintenance intervention, represented by another operational holon, to elaborate a conclusion about the failure and if necessary to re-initialise the machine.

Based in the diagnostic or the maintenance intervention, different scenarios can occur leading to different consequences: the machine can become immediately available or stay out of service for a more delayed repair intervention, and the part may have been destroyed or not. Finally, the maintenance team may have to remove manually the part from the machine and place it in the machine or cell buffer. If the part is not destroyed and the machine is ready to re-execute the work order, no action has to be performed.

An example of this type of machine failure is an error during the download of a program to the machine. In spite of being able to re-execute immediately the work order, it is however possible that another scheduled work order(s) becomes delayed, being then treated as a work order delay disturbance.

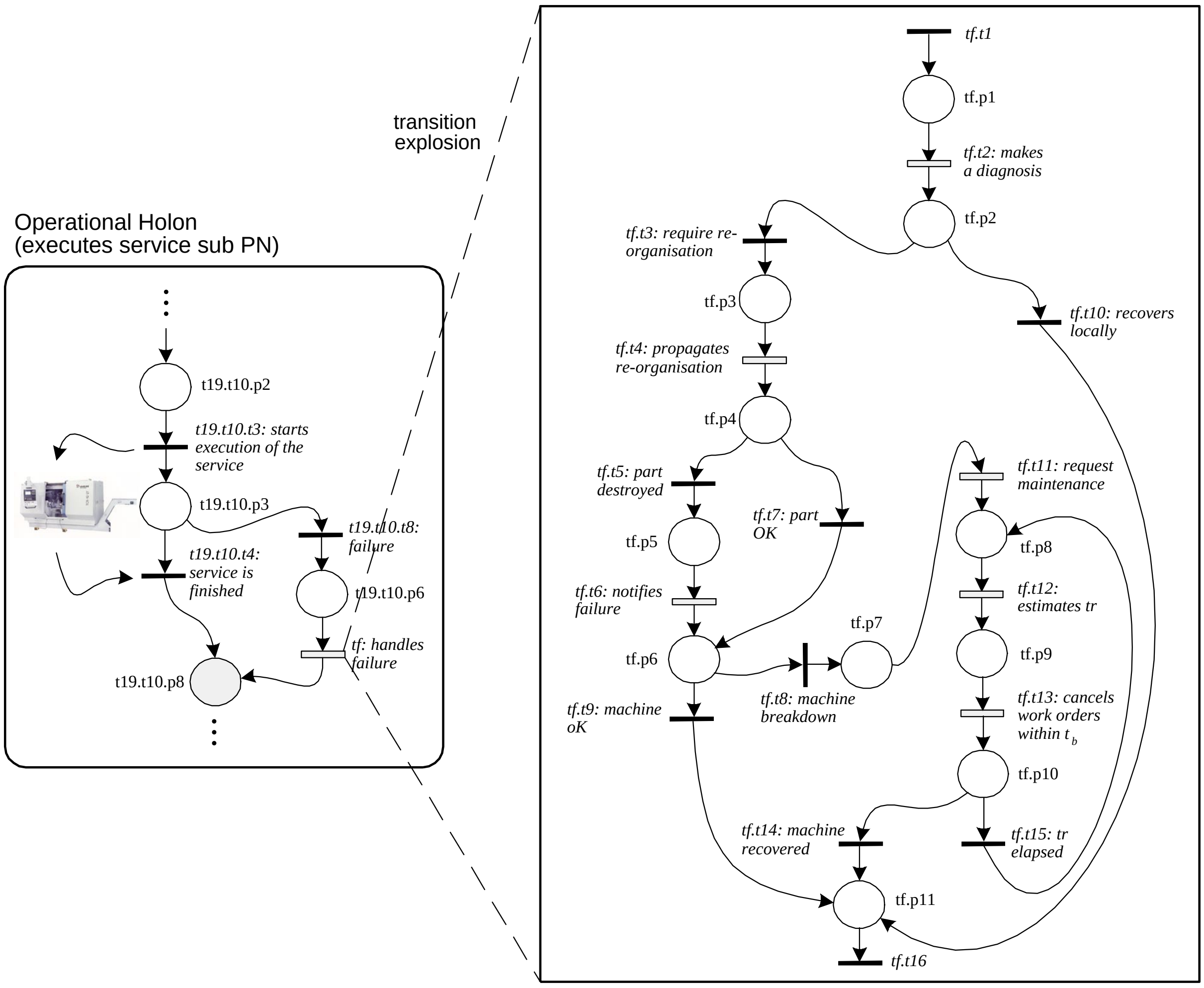


Figure 5.6: Petri Net Model for the Machine Failure (in an Operational Holon)

Re-organisation

If the occurrence of the machine failure provokes the need for external intervention (for example, to recover physically the machine) or re-organisation (for example, when the estimated downtime causes delays in the execution of work orders), the operational holon should adapt its behaviour, by re-organising its control structure into a heterarchical structure characterised by decentralised decision-making. For this purpose, it triggers the transition t_6 of the operational holon model to handle the transient state and propagates the need for re-organisation to the other holons.

The *propagates re-organisation* activity propagates the occurrence of the disturbance to the neighbour supervisor holon, giving the possibility of a global self-reorganisation. The procedure to execute the propagation of the re-organisation need consists of the deposit of pheromone in the supervisor holon, using the *propagate(pheromone)* message.

The main parameter associated to this message is the intensity of the pheromone, which corresponds to the estimated reestablishment time, the amount of time that the system will probably stay in the transient state. The reestablishment time is estimated by the operational holon, which detects the failure, and its value is dynamically adjusted to improve the system performance. The tuning of the reestablishment time uses the information of previous failures, indicating if the reestablishment time was sufficient or not to solve the problems resulting from the disturbance. The value of the reestablishment time should be kept as smaller as possible to avoid long transient states, but not so small that causes unstability in the recovery mechanism.

The supervisor holon then propagates the pheromone to the entities placed in its coordination domain and to the entities placed in a higher hierarchical level, allowing the evolution of the other operational holons into a heterarchical structure.

The operational holons, when sensing the pheromone odour, deregister from the supervisor holon for a period of time equal to the reestablishment time encoded in the pheromone intensity, starting the direct interaction between task and operational holons.

Notification of the Failure

After the propagation of the re-organisation need, the process evolves to handle the recovery of the failure.

The *notifies failure* activity is executed when the machine failure provokes the destruction of the part. The operational holon removes from its agenda the work orders associated to the destroyed part, and notifies the task holon about the destruction of the part, sending a *failure(work order)* message. This message is simultaneously sent to the supervisor holon notifying the occurrence of a machine failure.

The operational holon also executes a re-scheduling to optimise the execution plan.

Out of the Service

If the machine failure provokes the damage of the machine, the operational holon requires external intervention to repair the machine, by requesting a maintenance operation to the available operational maintenance holons.

The operational holon can forecast the possible required action(s) to recover physically the machine, based in the solutions used to recover from the previous occurrences, in order to contribute to the recovery of the machine breakdown, and pass this information to the operational holon that will be responsible to execute the maintenance operation.

Since the resource will be out-of-service for a longer period of time, the operational holon should forecast the recovery time, in order to handle its agenda by detecting the work orders that are planned to be executed during the expected downtime, and then cancel the actual allocation of these work orders, notifying the task holons.

For this purpose, the operational holon estimates two distinct time parameters: (1) t_b that defines the temporal window where work orders must be returned to the task holon, and (2) t_p, smaller than the first one, which is used to check if the machine has recovered, and to re-estimate and re-apply the time parameters if the machine has not recovered as forecasted. One possible way to calculate these two values is to determine the maximum recovery times required to solve 50% (t_p) and 90% (t_b) of the previous failures.

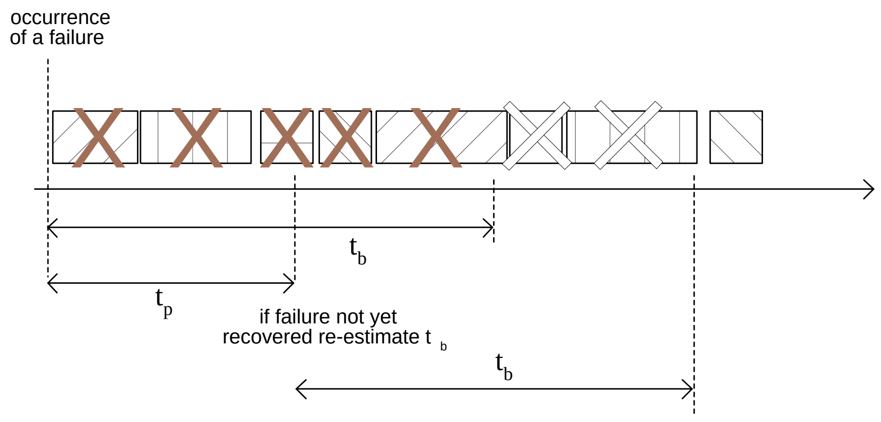


Figure 5.7: Cancellation of Work Orders within the Recovery Time Window

Using the estimated recovery times, the operational holon removes the work orders planned to be executed within the t_b time interval from its local agenda, sending a *take-back(work order)* message to the task holon for each of those work order, as illustrated in the Figure 5.7.

The process evolves along two distinct cases: if the machine is recovered, then it is available to return to its normal operation; if the recovery time t_p is elapsed before the recovery of the machine, then it is necessary to re-estimate the time parameters and to cancel the work orders that are planned to the new t_b time interval.

After the execution of the recovery action, the operational holon stores the relevant information about the failure, namely the recovery time and the solution used to recover the machine. The acquisition of this information allows to create new knowledge that will support future reaction to disturbances.

During the out-of-service state, the operational holon only accepts new work orders (either proposed by the supervisor holon or announced by the task holon), if they can be performed outside the the t_b time interval.

Behaviour of the Task Holon

The behaviour of the task holon in reaction to disturbances is also regulated according to the state of the product and the machine after the machine failure, as illustrated in Figure 5.8.

In case of work order failure it is necessary to execute two sets of actions in parallel: on one hand, it is necessary to remove the destroyed part from the machine, and to request a new pallet and a new part to be produced; on the other hand, it is necessary to re-allocate from the beginning all the work orders belonging to the production order.

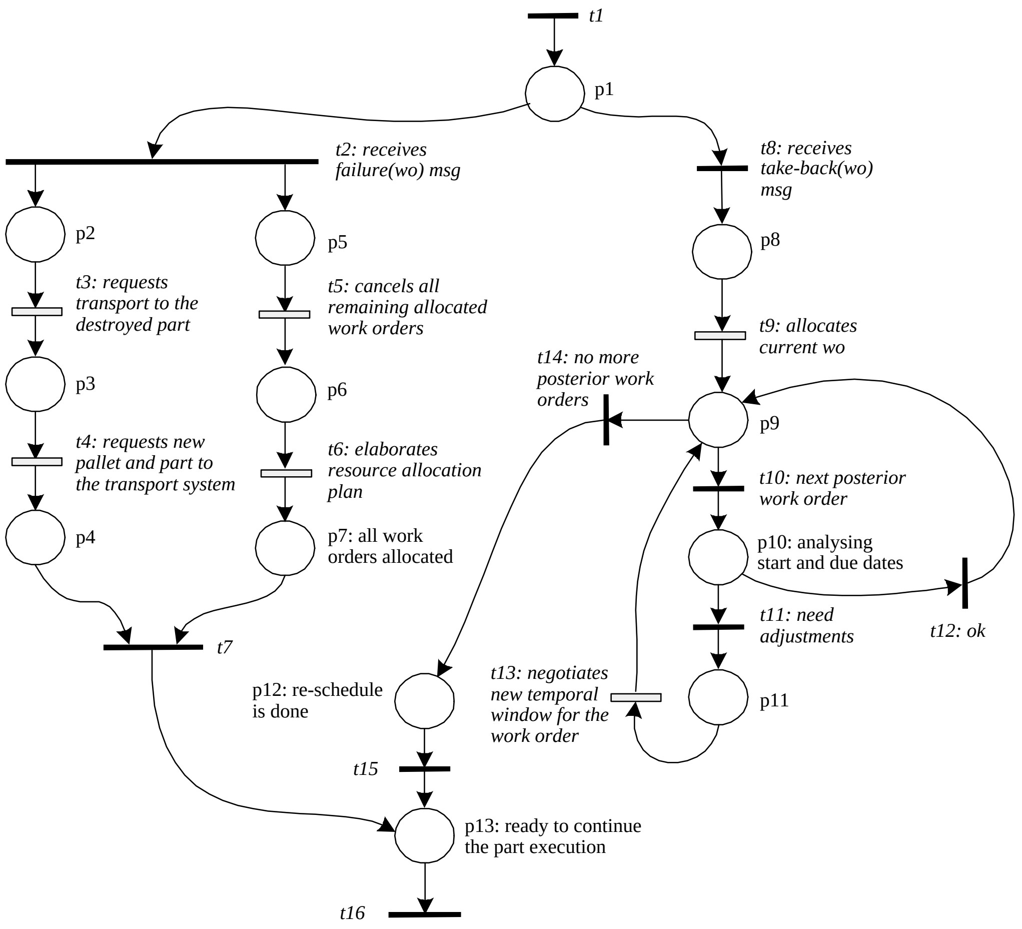

t1
p1
t2: receives failure(wo) msg
t8: receives take-back(wo) msg
p2
p5
p8
t3: requests transport to the destroyed part
t5: cancels all remaining allocated work orders
t9: allocates current wo
p3
p6
t14: no more posterior work orders
p9
t4: requests new pallet and part to the transport system
t6: elaborates resource allocation plan
t10: next posterior work order
p4
p7: all work orders allocated
p10: analysing start and due dates
t11: need adjustments
t12: ok
t7
p12: re-schedule is done
t13: negotiates new temporal window for the work order
p11
t15
p13: ready to continue the part execution
t16

Figure 5.8: Petri Net Model for the Disturbance Handling in a Task Holon

In case of the work order take back, the task holon re-schedules the work order and optimises the posterior work orders. Initially, the task holon analyses the initial alternative process plans provided by the product holon, combined with information obtained during previous allocation plan processes, in order to obtain more accurate and faster alternatives to react to the disturbance. The allocation of the taken back work order is performed by the *allocates current wo* activity, described in the resource allocation plan Petri net model.

The achieved allocation can lead to delays in the posterior work orders, requiring an adjustment of the temporal window to execute each work order through the negotiation with the associated operational holons responsible to execute the work orders.

5.3.2 Mechanisms for the Order Cancellation or Modification

The cancellation of a work order or production order can be considered as a simple disturbance at shop floor level that only requires the re-scheduling of the remaining work orders, in order to optimise the schedule.

After achieving the new schedule, the operational holon sends the *send(agenda)* message to the supervisor holon, allowing the synchronisation of both agendas. If it is the case, the material already used is sent to the waste, for recycle or to the automatic storage. It must be noticed that this type of disturbance may open free spaces in the agenda, allowing to execute earlier some work orders that were eventually delayed.

In case of modification of the order attributes, such as the temporal window to execute the order, it may be necessary to re-schedule the work orders, minimising deviations from the initial plan. In case of work orders that cannot fulfil the due date in the new schedule, it is necessary to notify the task holon, sending the *delay(work order, due date)* message and waiting for its agreement. In affirmative case, the schedule is confirmed, but otherwise, the work order is cancelled. The reaction to this kind of disturbance does not require the system re-organisation, only the local schedule optimisation.

Handling more complex modifications in the order attributes, such as orders with incomplete information, requires the development of more complex mechanisms that are not considered in this work. [Sousa et al., 1999a] and the references therein discuss the manipulation of incomplete information in the manufacturing domain.

5.3.3 Mechanisms for the Order Delay

A work order delay can occur after a disturbance, normally a machine failure or the introduction of a rush order, when the operational holon can not fulfil the pre-defined due date of a work order and proposes a new due date.

Detection of Work Order Delay

The procedure to handle the work order delay requires the dynamic detection of delays in the execution of the work orders. Each operational holon has a rule-based mechanism to detect all the unfinished work orders which scheduled start date parameter is earlier than the actual date, as illustrated in the Figure 5.9.

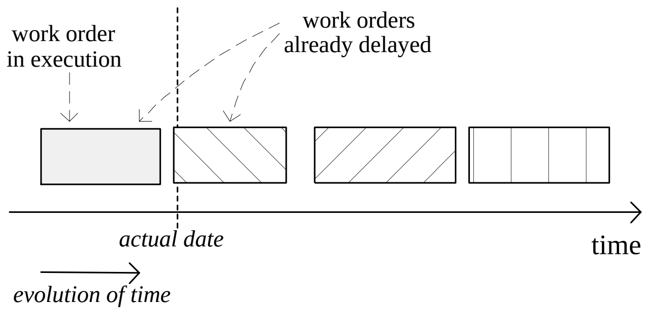


Figure 5.9: Detection of a Work Order Delay

The rule used to trigger the work order delay can be represented by,

Rule: IF $b_{ik} < actualDate$ THEN *Work Order Delay (wo_i)*

In certain applications can be useful to evaluate the progress in the execution of the work order and to detect the delay if the progress does not corresponds to the expected. This feature requires the division of the work order in several small steps and the capability to monitorise and to compare the progress of each step.

Reaction to the Work Order Delay

In case of work order delay the operational holon notifies the task holon about the delay, proposing a new date using the *delay(work order, due date)* message. The work order is removed from the local schedule only if the task holon does not accept the delay in the execution of the work order. At the same time, the operational holon propagates the need for re-organisation, using the *propagate(pheromone)* message.

The task holon, according to the Petri net model of the Figure 5.10, stores the information about the delay, in order to be considered in the future resource allocations processes.

The decision about the acceptance of the work order delay is dependent of the actual state of the work order. If the work order is already in execution, this notification is saw as an warning of delay, being necessary to re-schedule all the posterior work orders affected by the delay. If the work order is waiting for the execution, the task holon can try to find alternative resources to allocate the work order by asking other operational holons about their capacity to execute the work order.

Based in the proposals sent by the operational holons and in the estimated delay, the task holon decides if the proposal for the estimated delay is accepted or changes the allocation to

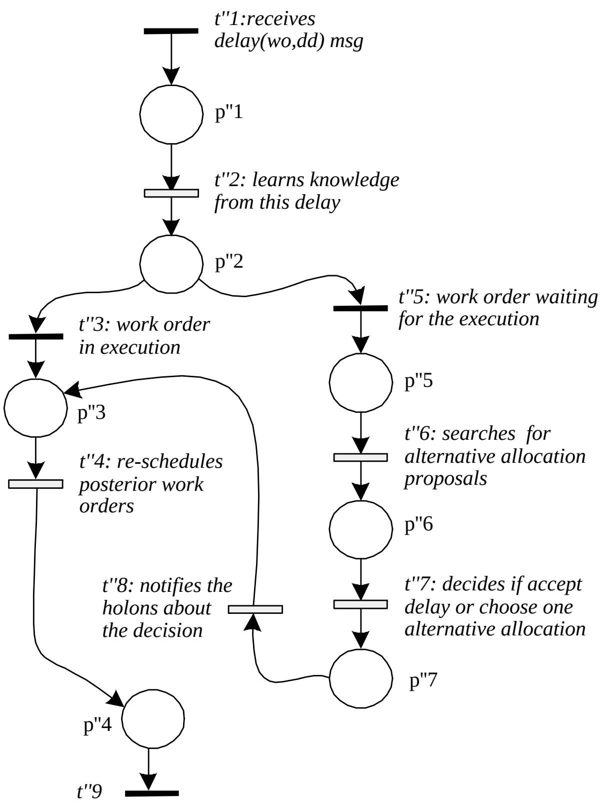

t"1:receives delay(wo,dd) msg
p"1
t"2: learns knowledge from this delay
p"2
t"5: work order waiting for the execution
t"3: work order in execution
p"5
p"3
t"6: searches for alternative allocation proposals
t"4: re-schedules posterior work orders
p"6
t"8: notifies the holons about the decision
t"7: decides if accept delay or choose one alternative allocation
p"7
p"4
t"9

Figure 5.10: Petri Net Model for the Work Order Delay (TH point of view)

another operational holon. Additionally, the task holon re-schedules the posterior work orders, adjusting their scheduled start and due dates.

5.3.4 Mechanisms for the Introduction of Rush Orders

A rush order is an order that arrives to the system and should be processed immediately, since it has a near due date. As the schedule plans are elaborated periodically by the supervisor holons, the treatment of these kind of orders leads to a disturbance in the system.

Distributed Scheduling

The rush production order, usually of high priority, causes the launching of a task holon described previously, which announces the work orders to the operational and supervisor holons in the system.

As the supervisor holon only executes the scheduling at periodic pre-defined moments, the need for immediate treatment of the rush order requires the re-organisation of the holons, entering in the transient state. For this purpose, the supervisor holon propagates the need for re-organisation to the neigbour holons, using the *propagate(pheromone)* message.

In the transient state, the task holon with the rush order interacts directly with the operational holons to allocate it, using the resource allocation mechanism already described. The problem appears if this rush production order has to be executed in a time window already occupied by other tasks, which requires a special negotiation to relax the other work orders and to introduce the new ones.

Since each order has associated a priority, w_i, the operational holons take this information in consideration. In the case that the rush order has maximal priority, it is a work order that must be executed as soon as the current work order is completed, the operational holon tries to re-schedule, relaxing the work orders that have minimal priority, i.e. those work orders that can be delayed beyond the due date, to find capacity to execute the rush work order.

In case of impossibility, the operational holons sends a *refuse(wo, list-TH)* message, indicating the task holons that have work orders allocated in the required temporal window.

Negotiation between Task Holons

In case of impossibility to find capacity to allocate the rush production order, the task holon has to negotiate with the task holons that have work orders allocated to the resources occupying the requested time window to execute the rush order, trying to convince them to release some time window.

This mechanism is based in the trade of rewards and penalties, the credits units, that represent the performance and the trust of a holon. The task holon can use the penalty value, that it will pay if does not fulfil the due date, to manage the problem of finding a time window to execute the rush work orders.

In the negotiation process, illustrated in Figure 5.11, the task holon that is responsible for the execution of a rush order, initially sends a *propose-reward(interval, reward)* message to the other task holons, requesting the desired time window and offering a reward. Each one of the other task holons analyse the offer and in case that the reward covers the penalty that the task holon may to pay for the delay, it accepts; otherwise, it rejects the proposal.

In case that all task holons reject the request, the task holon that has the rush order must increase the reward value and make another offer to all the task holons. The task holon should repeat this procedure until one task holon accepts the offer or the offered reward value reaches the maximum value, which is equal to the penalty to be paid in case of delay.

In case that one task holon accepts the offer, it will notify the operational holons of the decrease of priority to free the time window. In parallel, the task holon responsible for the rush order announces again the production order to the operational holons.

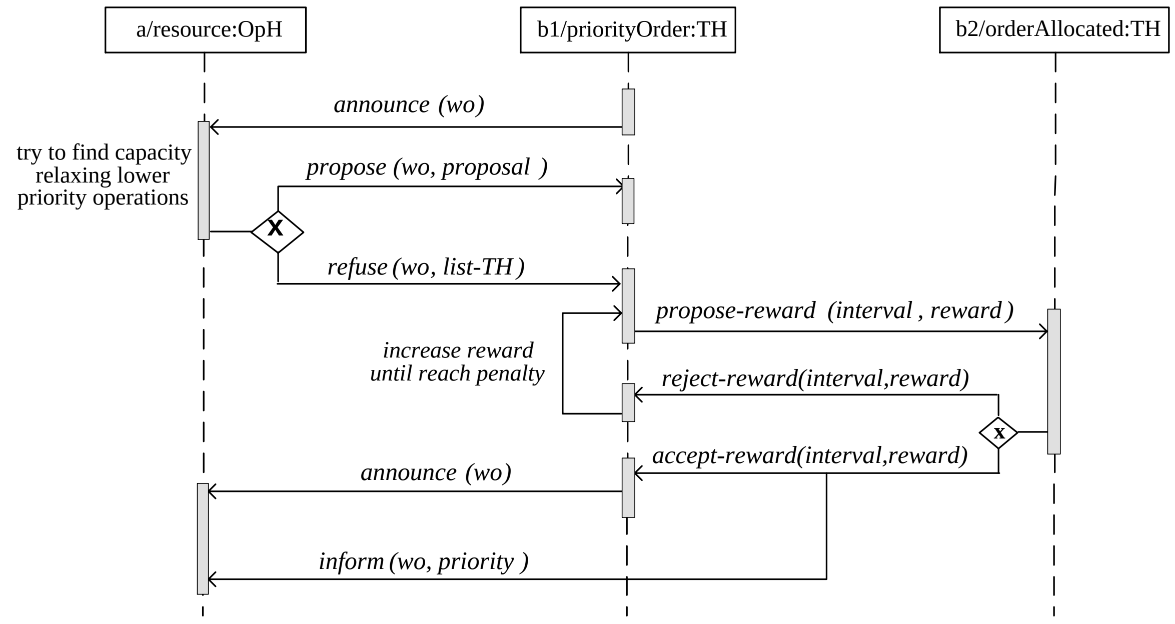

a/resource:OpH
b1/priorityOrder:TH
b2/orderAllocated:TH
announce (wo)
try to find capacity relaxing lower priority operations
propose (wo, proposal)
X
refuse (wo, list-TH)
propose-reward (interval , reward)
increase reward until reach penalty
reject-reward(interval,reward)
x
accept-reward(interval,reward)
announce (wo)
inform (wo, priority)

Figure 5.11: Interaction Protocol for the Negotiation between Task Holons

5.4 Prediction of Disturbances

Traditionally, the disturbance management mechanisms are purely reactive, i.e. the system only applies corrective procedures when the disturbance occurs. The improvement of the disturbance management by planning the production in advance, requires the existence of a predictive mechanism that estimates the occurrence of disturbances.

Some disturbances are not purely random processes, but they obey to some hidden patterns that may be difficult to identify.

5.4.1 Why to Predict Future Disturbances?

The estimation of the expected time for the occurrence of the next disturbance, t_f, allows to plan in advance the occurrence of the disturbance, for example by planning predictive maintenance operations according to the production convenience, or by reserving empty capacity according to the expected time to recover from the disturbance.

A pertinent question related to the disturbances is if they are really disturbances or just normal situations in the system. The real disturbances are those that are result from unpredictable events. Additionally, an event can be a disturbance at one moment, but in the future may become a normal event. As an example, illustrated in Figure 5.12, the events of type e_1 are disturbances, since they are not predictable in the analysed time window, but the events of type e_2 are considered disturbances only at the beginning, but with the increase of number of occurrences it is possible to establish a pattern and to extract the occurrence frequency, allowing to predict the future occurrence of the same events.

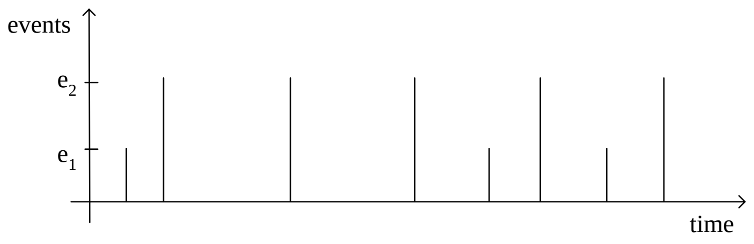


Figure 5.12: Example of a Range of Unexpected Events

The control system should be able to analyse the events and decide when an event is a real disturbance or a normal situation, requiring for this, the use of appropriated learning mechanisms associated to the holon behaviour. The objective is to find patterns in the occurrence of disturbances foreseeing the stochastic effects of industrial environment, making predictable the occurrence of future disturbances by planning in advance their occurrence. With the increase of predictability, the disturbances left to be real disturbances and became normal situations, since it is possible to plan their occurrence instead of reacting to their occurrence.

5.4.2 How to Predict Disturbances?

The prediction of future disturbances is based in understanding the gathered data to find a pattern in the historic disturbance data, which is not a simple and easy job.

In manufacturing systems it is normally possible to define the mean functioning time and the mean failure time. The Mean Time Between Failures (MTBF) provides the indication of the mean time a machine is operational between two consecutive failures, and allows forecasting the occurrence of the next disturbance.

However, the use of this simple average mathematical treatment to achieve the t_f parameter can lead to unsatisfactory results, since the objective is to find patterns in the historic events and not only the simple average. Additionally, in the manufacturing domain different types of disturbances are handled, besides machine failures, presenting different models to represent the disturbance sequence. This requires the use of more complex treatments that do not simple memorise the incoming information but understand and interpret the information supplied by the environment.

The recognition of patterns in the historical disturbance data, can be supported by several available approaches, such as unsupervised learning methods, frequency analysis, bayesian probability theory and neural networks. [Leitão and Restivo, 2003b] describes an approach for foreseeing the occurrence of future disturbances using an unsupervised learning mechanism based in the statistical clustering technique [Kubat et al., 1996], that predicts the time interval between consecutive disturbances.

5.4.3 Planning the Future using Prediction

In the planning and scheduling functions, as illustrated in the Figure 5.13, it is considered that t_f time after the last occurrence of the disturbance, a similar disturbance will happen.

In cases of machine failures, the prediction of future disturbances allows to support the planning of preventive maintenance operations, preparing the production for the effects of the disturbance. The planning of preventive maintenance operations is done according to the production convenience and based in the historic machine failures. The preventive maintenance operations allows to minimise the occurrence of machine failures and thus avoid the need to use corrective maintenance, which implies to stop the machine and in certain situations to stop the whole production system.

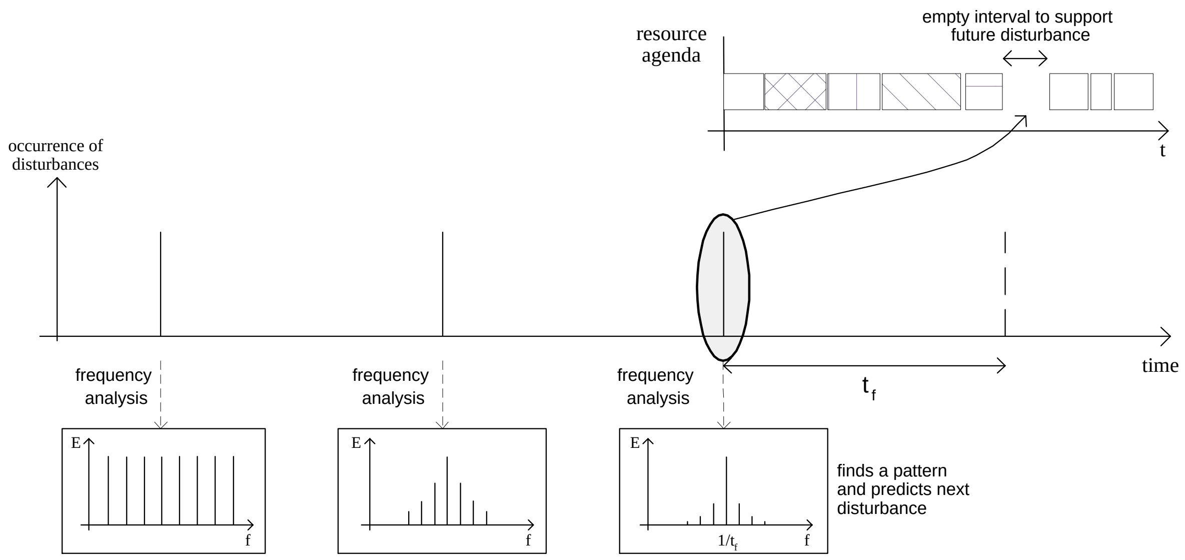

empty interval to support future disturbance
resource agenda
t
occurrence of disturbances
time
t_f
frequency analysis
frequency analysis
frequency analysis
E
f
E
f
E
$1/t_f$
f
finds a pattern and predicts next disturbance

Figure 5.13: Determination of Disturbance Patterns

The scheduling mechanism can also plan to request or to purchase the necessary components to execute the maintenance operations. In order to optimise the schedule, it is important to aggregate, if possible, the execution of several maintenance operations to different machines or systems. Additionally, if possible, it will be useful to aggregate the maintenance operations with the setup operations, preparing the machine or the system to the execution of next processing operations.

For another types of disturbance, the operational holon can consider a short period of empty capacity in the schedule, planned according to the predictable occurrence of the disturbance, allowing a flexible and agile treatment of the disturbance. The empty capacity interval is estimated taking in consideration the average value of the previous recovery time for the same disturbance type.

During the plan execution, two different scenarios can occur: the disturbance occurs and small modifications are required in the scheduling, since the disturbance was already predicted, or the disturbance does not occurs , and the prediction mechanism parameters should be adjusted. In this last scenario, slightly modifications in the scheduling are also possible, moving the work orders for the empty spaces.

5.5 Summary

In industrial manufacturing systems, characterised by stochastic and volatile demand, the occurrence of unexpected disturbances implies the degradation of optimised plans, leading to a decrease of the system performance parameters. In these circumstances, the response to change is a major aspect to consider.

The proposed adaptive holonic control approach comprises the stationary and transient states. The disturbance management associated to the transient state considers the detection of symptoms, identification, reaction and prediction of disturbances.

In this chapter, the mechanisms to react to disturbances, by selecting the proper behaviours, were analysed in detail. A simple prediction mechanism to support the production planning in advance, by increasing the predictability of occurrence of future events, was introduced as an extension of the traditional reaction mechanisms.

In the next chapter, the ADACOR architecture will be validated through the implementation of its concepts into a prototype, to allow the extraction of conclusions about the validity and applicability of the proposed concepts.

Chapter 6

Implementation and Experimental Validation

All truth passes through three stages. First, it is ridiculed.
Second, it is violently opposed. Third, it is being self-evident.
Arthur Schopenhauer

In the previous chapters, an adaptive manufacturing control architecture was described, based in the holonic manufacturing paradigm and addressing the agile reaction to unexpected disturbances at the shop floor level. The validation of the proposed architecture is required to verify the correctness and the applicability of its concepts.

The validation of the architecture can be made at two different levels: simulation of the model that represents the system and elaboration of experimental validation. The first approach was partially elaborated during the design of Petri net models for the ADACOR holon classes, with the structural, behavioural and performance analysis of these models. The second approach requires the application of architecture concepts into a case study, and posterior analysis of the manufacturing control system behaviour under several different manufacturing scenarios. This last approach will be described in this chapter.

Initially, a procedure to analyse and evaluate manufacturing control systems will be introduced, by identifying quantitative and qualitative indicators, and defining procedures for their measurement and evaluation. The implementation of the ADACOR holonic control concepts into a prototype will be described, focusing mainly the development platform and the implementation of the ADACOR holons. The experimental case study used to test the ADACOR concepts is also described, defining the production system and manufacturing scenarios. At last, the experimental results will be analysed and some conclusions about the validation of the ADACOR architecture are elaborated.

6.1 Performance Measurement

The assessment, evaluation and comparison of achievements of different manufacturing control systems and the cooperation between manufacturing control systems developers, require the definition of appropriate performance measurement frameworks. In this section, a performance measurement methodology will be introduced, and applied to evaluate the developed prototype.

6.1.1 Performance Measurement Overview

Performance measurement is the process of using a tool or a procedure to evaluate a concrete efficiency parameter of the system. As an example, a common way of evaluating a car engine performance is to calculate the kilometres per litre ratio, i.e. the ratio of the number of kilometres driven to the number of litres of petrol consumed. The term **measurement** implies that the approach being used is rigorous, systematic and quantifiable. Thus, the consumption has no meaning if it is not clear the scenario (the road conditions, the road type, etc.) and the control system (the driven).

A performance measurement procedure must be objective, based on scientific evidence and must not affect or distort results. In developing a performance measure, it must be tested to ensure that it is reliable (i.e. the use of the procedure results in the same reading regardless of who does the measuring or when and where the measurement is taken), valid (i.e. the tool measures what is intended) and standardised (i.e. the definition of standards, data elements, data collection, and data analyses are sufficiently precise and comprehensible that they can be understood and applied in the same way regardless of who refers to or applies them).

Performance measurement results describe an observed level of performance (such as throughput rate or number of industrial accidents per year) allowing to analysed the performance of a system and to compare the performance of different systems[1], but they don't tell why the performance is as it is. Results cannot reveal which factors account for differences in measured levels of performance.

Traditionally, the performance measurement approaches have a scope focused in the financial

[1]As example, nowadays the measure of productivity is in the order of the day, being used to compare and rank the economic performance level of each country.

aspects. [Neely, 1999] identified the main reasons why these types of measures are criticised:

- Encourage short-termism and local optimisation.
- Lack strategic focus and fail to provide data on quality, responsiveness and flexibility.
- Do not encourage continuous improvement.

A broad number of performance measurement systems (PMSs) has been proposed and in some cases deployed in practice, using parameters more relevant to the manufacturing area and that can be used to drive the production process. Among the most widely cited of these PMSs are: the Balanced Scorecard [Kaplan and Norton, 1996], the Performance Measurement Matrix [Keegan et al., 1989], and the Integrated Dynamic Performance Measurement System [Ghalayini and Noble, 1996]. These approaches present some disadvantages, some of them identifyed by [Bititci et al., 2002]:

- Most of the performance systems are historical and static, which does not reflect the dynamic aspects associated to the manufacturing changes.
- Only few performance measurement systems have an IT infrastructure, which leads to cumbersome and time-consuming data collection and reporting.
- These approaches focus in the performance measurement of the manufacturing system, and not in the associated control system.

Since the scope of this research work is the manufacturing control system, our intention is the definition of suitable set of performance indicators addressing the relevant aspects to evaluate a manufacturing control system. In these circumstances, it is proposed a methodology to evaluate the performance of manufacturing control applications, which comprises the following main steps [Leitão and Restivo, 2004]:

- Identification of a set of performance indicators adequate to manufacturing control.
- Proper definition of each identified performance indicator.
- Definition of a procedure to measure each performance indicator.

In manufacturing context, the performance parameters can be classified as qualitative or quantitative. The quantitative measures are based on different production performance indicators, such as the manufacturing lead time, the tardiness, the waiting time, the throughput and the WIP. The qualitative measures are of a more subjective nature and reflect properties of the manufacturing control solution, such as the agility, flexibility and robustness, which cannot be directly obtained from the production data. The choice of the performance measurements to be used depends on the business domain. For some applications the throughput is the major performance indicator, but other can consider the robustness or service quality the focus in the performance measurement.

In the following sections, the proposed methodology will be exemplified by identifying some performance indicators to evaluate manufacturing control systems, and defining procedures to measure those indicators.

6.1.2 Quantitative Performance Indicators

The analysis of the performance of a manufacturing control system, using quantitative measures, is based in statistical theory and is related to system measures, such as throughput and WIP, production resources measures, such as processing time and percentage of utilisation, and order measures, such as manufacturing lead time, tardiness and number of delayed orders.

The control systems running on multitasking platforms with parallel threads and processes communicating, associated to a certain degree of randomness in the allocation of operations, produce stochastic variations on the processing times, which requires the execution of the same experience several times to achieve more accurate results, allowing also to verify the repeatability of the control system. For this purpose, each experimental scenario will be executed $e = \{1, 2, ..., n\}$ times, being possible to extract the average and standard deviation values that characterise the experimental scenario.

Manufacturing Lead Time

The manufacturing lead time, L, is the total time required to process a given product (or part) through the factory plant, and comprises the setup time, the no-operation time (handling, intermediate storage, inspection, transport, etc.), the idle time and the processing time.

The lead time has highest importance in the analysis of manufacturing systems performance because it reflects production optimisation level, influencing the the factory plant productivity. As shorter is the manufacturing lead time as more products the production plant can produce in the same period.

The average and standard deviation of manufacturing lead time for each scenario, considering n experiences, is given by:

$$av(\overline{L}) = \frac{\sum_{e=1}^{n}(\overline{L}_e)}{n} = \frac{\sum_{e=1}^{n}\sum_{i=1}^{N} L_{ie}}{n \cdot N}$$

and

$$s^2(\overline{L}) = \frac{\sum_{e=1}^{n}[(\overline{L})_e - av(\overline{L})]^2}{n-1}$$

where $\overline{L}_e$ is the average of manufacturing lead time over the several orders within one experience.

Tardiness

During the execution of a manufacturing plan some parts can be delayed from the initial due dates, mainly due to the constraints of actual plant capacity and the unexpected disturbances occurred.

The tardiness, T_i, represents the delay associated to the execution of a production order and can be modelled by $T_i = max(0, C_i - D_i)$, where C_i is the completion time and D_i is the due date. This expression means that the tardiness is equal to the difference between the order completation date and the due date if the difference is positive; otherwise the tardiness is zero.

The analysis of the tardiness is done by applying the statistical treatment used to calculate the average and standard deviations of manufacturing lead time. The average tardiness, $av(\overline{T})$, represents the average of all the parts tardiness and shows how agile are the manufacturing control application to support the constraints and reaction to disturbances maintaining the due dates.

Throughput

The throughput, Q, is a real indicator about the productivity of a manufacturing system. Throughput can be defined in several distinct ways, being in this work defined as the number of items produced per time unit. In industrial environments, the measurement of throughput is performed in a continuous form, i.e. measuring and actualising dynamically the throughput value, at the end of production process.

For simulation purposes and in the context of this work, the throughput can be measured by analysing the results of experimental tests. In this way, for each experimental test, it is measured the batch time required to execute all the parts belonging to the test scenario, as ilustrated in the Figure 6.1. Thus, the throughput of each experience is equal to the ratio between the number of parts produced in the experience and the batch time.

The throughput of an experimental scenario is achieved by the mean value of all throughput values of performed experiences that belong to the scenario.

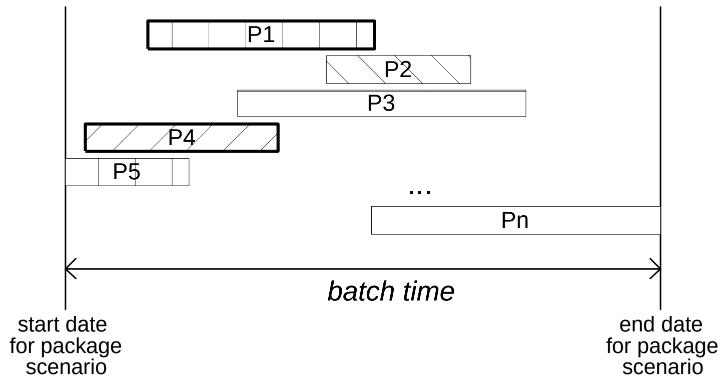

P1
P2
P3
P4
P5
...
Pn
batch time
start date for package scenario
end date for package scenario

Figure 6.1: Measuring the Throughput using the Batch Time

Resource Utilisation

Another important indicator for the analysis of the optimisation degree of the manufacturing control system is the percentage of utilisation of each resource. The resource utilisation can be defined as the percentage of processing time during a time interval.

Considering pU_j as the percentage of utilisation of a resource R_j, the average of resource capacity is given by,

$$pU = \frac{\sum_{e=1}^{n}(pU_{je})}{n}$$

Additionally, it is important to know the standard deviation of the capacity of all manufacturing resources to verify if the manufacturing load were well distributed by the resources or concentrated in few resources. Thus, the standard deviation of the capacity is given by

$$s^2(pU) = \frac{\sum_{e=1}^{n}[pU_{je} - pU]^2}{n-1}$$

Theoretically, the standard deviation value should be as smaller as possible, indicating the degree of distribution of the load over the available resources. This parameter allows to detect the existence of overloaded resources, which should be treated with special attention in the future by re-distributing some load to other resources.

Nowadays, these performance measures are becoming less important, as they don't reflect the dynamic response of the manufacturing system, that is, its ability to react to unexpected situations.

Predictability

A parameter that defines the performance of a control system is the predictability. Predictable shop floor controllers imply that the estimated due dates for the customer orders are reasonable, which in certain cases is more important than the performance parameters, for example, can be more important to have a predictable tardiness than a not reasonable estimated due date.

The predictability is dependent of several factors, such as the quantity of available information (i.e. knowing the actual load of the system and the number of work orders belonging to the production order), stochastic environment (i.e. differences in the duration of the execution of operations and the occurrence of disturbances) and non-linear dynamics of manufacturing system (i.e. the same experience performed several times can lead to different results).

A measure of the predictability can be achieved by repeating the same experience several times, extracting the standard deviation of the experiences. The predictability of an individual order for one scenario is given by:

$$s^2(L_i) = \frac{\sum_{e=1}^{n}[L_{ie} - av(L_i)]^2}{n-1}$$

where $av(L_i)$ is the manufacturing lead time of an order i averaged over different sample experiments. Using the value $s(L_i)$ it is possible to quantify the predictability of the orders for one scenario, which is given by:

$$\overline{s(L_i)} = \frac{\sum_{i=1}^{N} s(L_i)}{N}$$

Another important conclusion that can be extracted from the analysis of the percentage of utilisation of the resources is the repeatability of the manufacturing control system. The repeatability is given by,

$$r(pU) = \frac{\sum_{j=1}^{m} s(pU_j)}{m}$$

where $s(pU_j)$ is the standard deviation of the percentage of utilisation of a resource j over the several experiences and m is the number of resources. As smaller be the parameter $r(pU)$ as more repeatable is the manufacturing control system planning the production.

All parameters considered as quantitative performance measures are dependent of the manufacturing load, i.e. the number of manufacturing orders in the system. This dependency can be analysed by running different order book scenarios, to extract the behaviour of each performance indicator over the manufacturing load.

6.1.3 Qualitative Performance Indicators

Manufacturing systems have a behaviour that averages over time when the system spent most of its time in a steady state. The parameters associated to the dynamic behaviour of a manufacturing system, such as the capability to support the introduction of one new product in the factory plant and the reaction to the occurrence of machine breakdowns, have more importance in the measurement of the performance of a manufacturing control system. However, in performance measurement research, little attention has been devoted to the system dynamics, specially to the quantification of these parameters.

In dynamic states, the system exhibits transients that present shorter duration and different behaviour when compared with its normal operation. An important aspect when dealing with the occurrence of disturbances is the response effectiveness. The parameters more suitable to assess this effectiveness are: re-configurability, agility and robustness. In our opinion, these parameters, which are the major key parameters in the analysis of the dynamic response, are difficult to measure presenting a certain degree of subjectivity in their analysis. In this work, the identified qualitative performance indicators will be defined, and methodologies for there measurement will be proposed.

Re-configurability Parameter

The flexibility parameter is related to the capability of the manufacturing system and its control system to support the production change when the consumption or demand changes, and the capability of re-configuration in the case of malfunctions of some resources[2].

In spite of the little attention that has been devoted to the measurement of flexibility of manufacturing systems and their associated control systems, some authors have approached the issue. As example, [Ibarrondo and Mercader, 2001] describes an approach to measure the volume and mix flexibility[3], based in measuring the minimal and maximum capacity of the system, and [Tsourveloudis and Phillis, 1998] introduces a fuzzy logic framework to measure several types of flexibility: machine, routing, material handling system, product, operation, process, volume, expansion and labour flexibilities.

None of these types of flexibility are the focus of our work, since in this work it is looked for the flexibility associated to the control system, which can be defined as re-configurability. The re-configurability of the control system is the ability to support different manufacturing system configurations, by the addition or remotion of machines, i.e. different production systems

[2]A more detailed and complete definition of flexibility is already presented in the chapter 2.

[3]Calling the operational flexibility to the aggregation of both types of flexibility.

scenarios, with a small customisation affort.

It is assumed that there are two dimensions on the effort, the quality and the quantity of the effort, and that these two values can be measured/estimated for a certain of previsible scenarios. The measures to determine the re-configurability of the control system are of two different types: degree of complexity of the customisation and development time for the customisation [Leitão and Restivo, 2004].

The degree of complexity parameter is a subjective parameter that can be evaluated in a relative way and expressed in a scale from 0 to 100 percent, through questionnaires to the software developers and manufacturing system engineers. The development time can be measured counting the time spent when customising the application, or estimated by experts. These two parameters are highly subjective and not quantifiable for a given set of scenarios. For this purpose, it is proposed a fuzzy approach to this question, reflecting this qualitative nature.

The degree of complexity can be fuzzified into the fuzzy sets {Low, Medium, High}. Each one of the previous terms is a fuzzy variable defined on the base variable. Since a fuzzy set A is a collection of ordered pairs $A = \{(x, \mu(x))\}$, where the item x belong to the universe and $\mu(x)$ is its grade of membership in A, it is necessary to define the membership functions for each fuzzy variable.

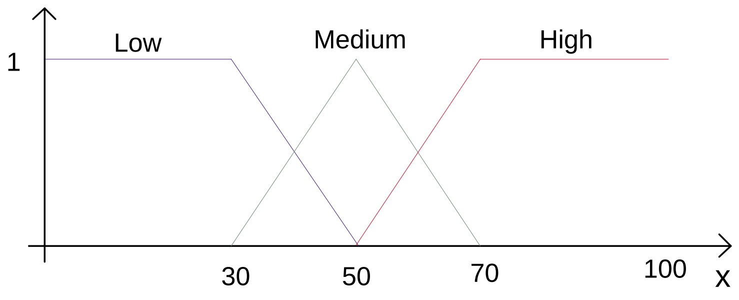


Figure 6.2: Membership Functions for the Degree of Complexity Variable

In case of the degree of complexity fuzzy variable the membership functions are illustrated in Figure 6.2. The membership functions for the Low and High fuzzy sets use a trapezoidal function type, and the membership function of Medium fuzzy set uses a triangular function type[4].

Applying a similar fuzzyfication procedure to the development time parameter, this may be represented in terms of the fuzzy sets {Low, Medium, High}.

[4]The use of triangular and trapezoidal functions type are the simplest way to define membership functions, but they not present a smoothness surface in the output parameter. The use of a Gaussian function type allows mainly to achieved smoothness in the output parameter.

The re-configurability parameter is also expressed in terms of the fuzzy sets {Low, Medium, High}, using similar membership functions to those defined for the degree of complexity variable. The fuzzy inference process is based in a fuzzy rule base. The reconfigurability fuzzy variable is determined according to the fuzzy rules, such as those expressed in the Table 6.1.

Table 6.1: Fuzzy Rules to Determine the Flexibility Parameter

Degree of Complexity	**Customisation Time**	**Reconfigurability**
Low	High	Low
-	Medium	Medium
Low	Low	High
Medium	High	Low
Medium	Low	High
High	High	Medium
High	Low	High

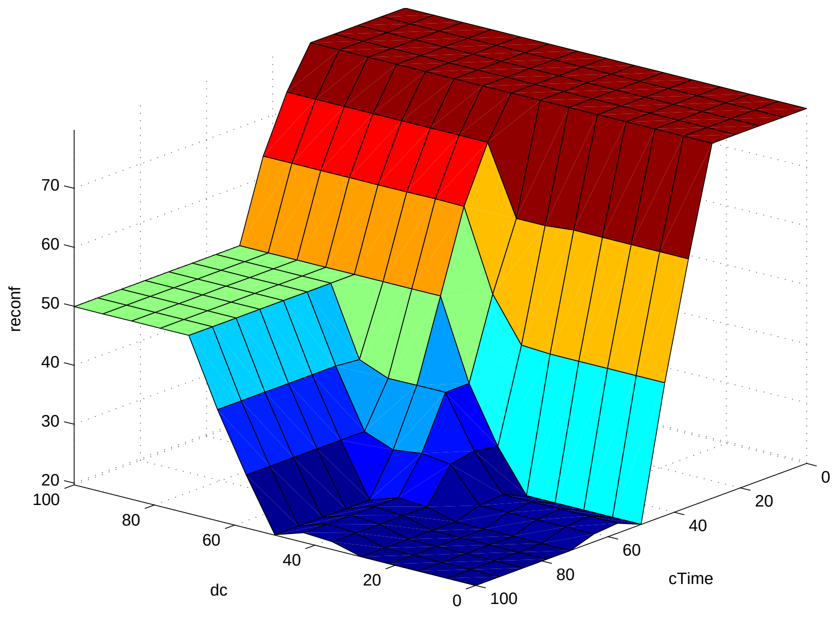


Figure 6.3: Surface of Possible Reconfigurability Solutions

Figure 6.3 shows the surface of possible solutions for the reconfigurability parameter, according to the variation of the two inputs of the system.

Robustness Parameter

As defined in the beginning of this work, the robustness of a control system is the capability to remain working correctly and relatively stable, even in presence of disturbances.

A robustness performance measurement procedure should measure how a system reacts to possible erroneous inputs or environmental factors that could affect the system. Ideally, it is necessary to exercise the system with all possible errors, conducting to an absolutely robust system. However, in reality, it is not possible to test all possible natural errors that can occur in the system (verifying the system operation and waiting for the occurrence of errors that occur infrequently is too time consuming).

Until now, there has been no effective approach to quantitatively measure the robustness of a manufacturing control system. Here, a simple and repeatable way to measure the manufacturing control system robustness is introduced.

The robustness performance measurement procedure will give a relative measure of robustness, having not a quantitative value. The procedure to measure the robustness comprises the introduction of possible errors (as many as possible) and the verification if the control system remains working correctly [Leitão and Restivo, 2004].

Figure 6.4 illustrates an example of a set of 20 what-if conditions that evaluates the robustness of a manufacturing control system. In this case, the degree of robustness can be extracted by attributing 5% for each correct test performed by the system according to the questionnaire. As example, a manufacturing control system application that fulfils only 10 what-if tests, will have a score of 50%. Using the score obtained by each manufacturing control system solution it is possible to rank the several solutions in terms of robustness.

At the moment, the best set of what-if questions is an open issue, which requires an additional effort to generalise the robustness measurement procedure.

Agility Parameter

The agility of a control system is the capability to react in a short period of time to the occurrence of unexpected disturbances. Thus, the agility is a time based property, being the time needed by the system to recover properly from the occurrence of a disturbance.

Agility is dependent of the manufacturing system flexibility, especially from the routing flexibility, since it is only possible to react with agility to the occurrence of disturbances if exists capacity and alternative resources in the manufacturing system. Thus, a system has low agility in case of low routing flexibility.

The proposed approach introduces the concepts of frequency and duration of the disturbances

What-if Tests to Evaluate the Control System Robustness

T1: During the normal operation the system remains stable?

T2: Introduce a new component in the system (e.g. a new machine). The system remains stable?

T3: Breakdown a centralised component of the system (such as a central scheduler or a supervisor). The system remains stable?

T4: Breakdown a local distributed component of the system (such as a resource, an order or a component that integrates a legacy system). The system remains stable?

T5: Introduce a failure in the network that links the holon responsible for the resource and the physical resource. The system remains stable?

T6: Remove the available resource(s) to execute a job (for example, a transporter device). The system remains stable?

T7: Test the graphical interface by clicking on the buttons and introducing inputs value with different data types (for example, inputing a string in a date field). The system remains stable?

T8: Introduce the occurrence of machine breakdowns. The system react to it?

T9: Introduce configuration files for a holon not respecting the XML format define for the configuration file. The system remains stable?

T10: Include a data type error in the configuration file (for example, introducing a string value in the field related to the estimated processing time). The system remains stable?

T11: Remove all possible resources to execute an operation. The system remains stable?

T12: In this case, the system is able to continue the resource allocation process (for example, waiting a certain time and announcing again the execution of the operation)?

T13: Send messages to the holons with no understandable contents. The system remains stable?

T14: Introduce one rush order. The system respond to it?

T15: Increase the number of production orders. The system (schedulers) remains stable?

T16: Plug in a new scheduling algorithm (or at least change the scheduling heuristic rule). The system remains stable?

T17: Add new rules to the decision component. The system remains stable?

T18: Change the layout configuration or resource skills. The system remains stable?

T19: For the normal number of holons in the system, verify if exists communication overhead (for example, during the resource allocation process, bid proposals are not received).

T20: Duplicate the number of holons in the system. The response remains acceptable?

Figure 6.4: Set of What-if Tests to Verify the Robustness of the Control System

to quantify the control system agility [Leitão and Restivo, 2004]. In Figure 6.5 it is illustrated the evolution of the throughput over the time. This evolution is affected with the occurrence of disturbances.

The Dirac's impulse in the disturbance graph represents the occurrence of a disturbance

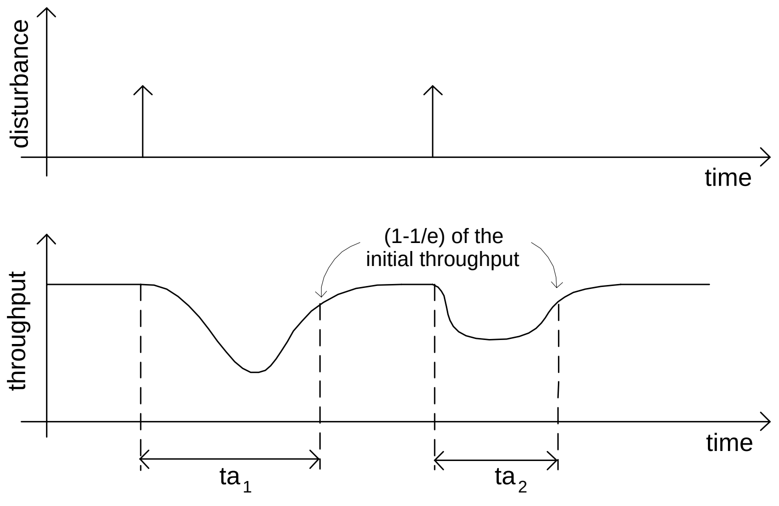


Figure 6.5: Measure of Agility of a Manufacturing Control System

which degrades the performance criteria, illustrated by the decrease of the throughput value. The parameter ta is the required time spent by the system to adapt its behaviour to the disturbance. This parameter is measured between the occurrence of the disturbance and the time when the system achieves a percentage of the initial throughput. A good indicator of this percentage is based in the time constant of RL and RC circuits, which is given by (1-1/e).

The parameter ta is a statistical value, calculated by determining the mean value of ta_i. The agility of a control system is translated by the ta parameter, being more agile as smaller the ta parameter is.

The agility can also be expressed in terms of the maximum number of disturbances, given by the inverse of ta, and it measures the maximum number of disturbances per time unit, which do not degrade the system performance below a certain level.

The measurement of agility using the previous approach requires the periodically measurement of a performance criteria, such as the throughput or the amount of processed material.

Loss of Productivity

Another approach to determine the agility is to run n experimental tests and to extract the agility parameter from the analysis of the loss of productivity in presence of disturbance scenarios.

Running a steady scenario it is extracted the throughput, which reflects the required time to produce a specific amount of items. Running a transient scenario, it is also measured the throughput, which is necessarily different, since the time required to execute the products is larger.

Having these two values of the throughput, it is possible to extract the loss of productivity by calculating a parameter that represents the percentage of the reduction of the throughput over the steady throughput:

$$lQ = (1 - \frac{Q_{transient}}{Q_{steady}}) * 100$$

The loss of productivity reflects indirectly how agile is the system, reacting to the disturbance, allowing to induct a relative value of the agility. The smaller the lQ value the higher will the agility of the system be.

This approach does not gives a quantitative value as expected since agility is a dynamic time-based parameter, but provides a good relative indicator of the agility, avoiding the complexity to measure periodically a performance indicator.

6.2 Implementation of the Prototype

The development of a holonic manufacturing control system based in the ADACOR architecture requires the implementation of those concepts into a prototype.

The ADACOR prototype uses agent technology to implement each holon that belongs to the holonic manufacturing control approach, taking advantage of its modularity, decentralisation and ccomponents re-use.

6.2.1 Agent Development Platform

Multi-agent systems can be adequately developed using usual object-oriented languages, such as Java. However, the development of multi-agent systems requires the implementation of features not supported by usual programming languages, such as message transport, encoding and parsing, yellow and white pages services, ontologies for common understanding and agent life-cycle management services, which increases the programming effort. To make easier the development of agent-based applications and to reduce the programming effort, it is important to use agent development platforms, which implement the previously referred features.

A significant set of platforms environments for agent development is available on commercial and scientific domain, providing a variety of services and agent models, which differences reflex of the philosophy and the target problems envisioned by the platform developers. Among the broad number of available agent development platforms, the following platforms were analysed [Barata et al., 2001]: Java Agent Development Framework (JADE)[5], FIPA-OS[6] and Java Agent Template (JatLite)[7]. A reference for a broad number of agent development tools can be found in

[5]JADE is provided by CSELT (http://jade.cselt.it/).

[6]FIPA-OS is provided by Emorphia (http://fipa-os.sourceforge.net).

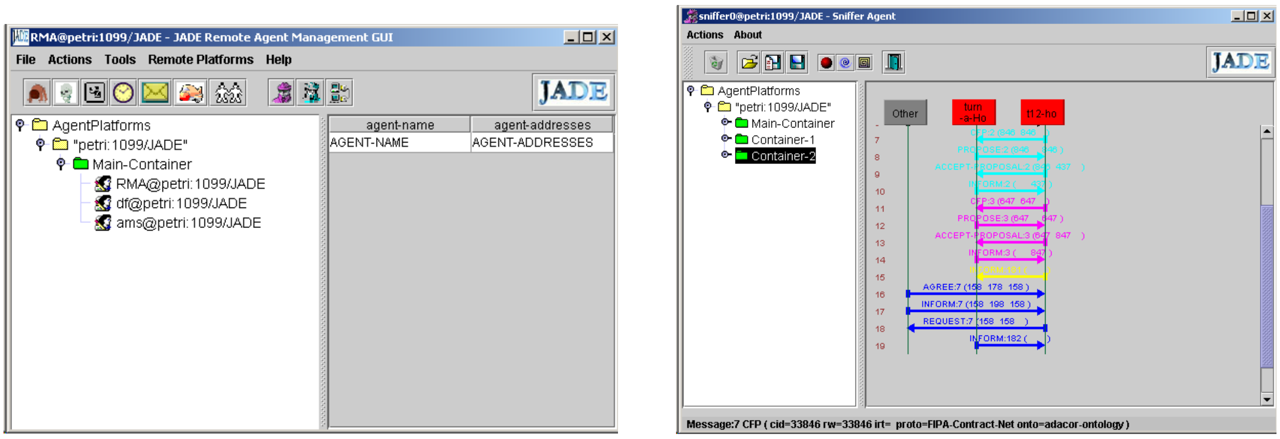
[7]JatLite is provided by University of Stanford (http://java.stanford.edu/).

the Agent Builder website (http://www.agentbuilder.com/AgentTools/) and a survey of some agent development platforms can be found in [Vrba, 2003].

The choice for an agent development platform obeyed to a set of criteria: to be a free platform, with good documentation and available support, ease of use, low programming effort, use of standards (such as FIPA), features to support the management of agent communities like white pages and/or yellow pages and facilities to implement rule oriented programming. In the current approach, the chosen platform was the JADE because it better responds to the mentioned requirements.

JADE aims to simplify the development of multi-agent systems by providing a set of system services and agents in compliance with the FIPA specifications: naming service and yellow-page service, message transport and parsing service, and a library of FIPA interaction protocols ready to be used [Bellifemine et al., 2002]. Additionally, JADE provides the mandatory components defined by FIPA to manage the agent platform, that are the Agent Communication Channel (ACC), the Agent Management System (AMS), and the Directory Facilitator (DF). The AMS provides white pages and agent life cycle management services, maintaining a directory of agent identifiers and states. The DF provides yellow pages services, and the capability of federation within other DFs on other existing platforms.

JADE uses the concept of *behaviours* to model concurrent tasks in agent programming [Bellifemine et al., 1999]. All agent communication is performed through message passing, where FIPA-ACL is the agent communication language to represent messages. JADE provides the FIPA SL (Semantic Language) content language and the agent management ontology, as well as the support for user-defined content languages and ontologies that can be implemented, registered with agents, and automatically used by the framework.



Message:7 CFP (cid=33846 rw=33846 irt= proto=FIPA-Contract-Net onto=adacor-ontology)

Figure 6.6: Graphical User Interfaces of the Remote Monitoring and Sniffer Agents

The agent platform provides a Graphical User Interface (GUI) for the remote management of the platform, allowing to monitor and control the status of agents, for example to stop and re-start agents, Figure 6.6. JADE provides also a set of graphical tools to support the debugging phase, usually quite complex in distributed systems, such as the Dummy, Sniffer and Introspector agents. The *Dummy Agent* is a monitoring and debugging tool that allows to edit, compose and send ACL messages to agents, and to receive and view messages from agents. The *Sniffer Agent* is a debugging tool that allows to track messages exchanged in a JADE agent platform using a notation similar to UML sequence diagrams, Figure 6.6. The *Introspector Agent* allows to monitor and control the life-cycle of a running agent, its exchanged ACL messages and the behaviours in execution.

JADE offers an easy and full integration with JESS (Java Expert System Shell)[8], where JADE provides the shell of the agent and guarantees (where possible) the FIPA compliance, while JESS is the engine of the agent that performs all the necessary reasoning [Bellifemine et al., 2002].

JADE also provides other features such as good documentation and an active mailing list to support technical problems.

6.2.2 Implementation of Individual Holons

The implementation of the ADACOR holon classes uses the JADE framework. Each ADACOR holon is a simple Java class that extends the Agent class provided by the JADE framework, inheriting basic functionalities, such as registration services, remote management and send/receive ACL messages [Bellifemine et al., 2002]. These functionalities were extended with features that represent the specific behaviour of the holon. As an example, the structure of the operational holon is,

```
import jade.core.*;
...
public class OperationalAgent extends Agent {
  Rete decisionEngine;  // rule-based engine
  ClientPIC pic;
  ...
  // The WaitForMessagesBehaviour waits for the messages from other agents.
  class WaitForMessagesBehaviour extends SimpleBehaviour {
    private boolean finished = false;
    ACLMessage msg=new ACLMessage(ACLMessage.INFORM);
    ...
    public WaitForMessagesBehaviour(Agent a) {
       super(a);
```

[8]JESS is provided by Sandia Laboratories, and available at http://herzberg1.ca.sandia.gov/jess/.

```
      }
      public void action() {
         msg = blockingReceive();    // wait for an ACL message
         if(msg != null){
            // start the HandleReceivedMessagesBehaviour
            addBehaviour(new HandleReceivedMessagesBehaviour(myAgent, msg));
         }
      }
      ...
    } // end of the class WaitForMessagesBehaviour
    ...
    protected void setup() {
      // Read the resource characteristics, stored in a file
      try {
          resource=fs.loadResourceConfig(pathName + ".xml");
      }catch (IOException e) {
          System.err.println("Error in handling the file.");
      }
      // Read the organisational structure configuration file
      try {
          resource=fs.loadOrganisationalStructureAsOperational(resource,fileName);
          ... // Registration with the appropriate DF
      }catch (IOException e) {
          System.err.println("Error in handling the file.");
      }
      ...
      // launches PIC component
      pic = new ClientPIC(PhysicalResource,"900");
      // launches rule base component
      decisionEngine = new Rete();
      try {
          ...
          decisionEngine.executeCommand("(batch Decision/ResDecision.clp)");
          ...
      } catch (JessException ex) { System.err.println(ex); }
      // launches ComC component
      addBehaviour(new WaitForMessagesBehaviour(this));
    }
  } // end of the class OperationalAgent
```

The behaviour of each holon uses multi-threading programming, over the concept of JADE's behaviour, allowing to execute several actions in parallel. Thus, when the holon is created, the first actions are its initialisation (such as to read the configuration files and to load the knowledge behaviour) and the registration into a federation according to the organisational structure. Afterwards, the holon's components, i.e. the communication, decision and physical interface components, are started.

The behaviours launched in the startup and those posteriorly invoked within these behaviours are also provided by the operational holon package in the form of Java classes.

The communication between distributed holons is done over the Ethernet network, using TCP/IP protocol and is asynchronous, i.e. a holon that sends a message continue its execution without the need to wait for the response. The messages specified in the ADACOR architecture are encoded using the FIPA-ACL communication language to achieve normalised communication between the ADACOR holons, being the content of the messages formatted according to the FIPA-SL0 language. The meaning of the message content is standardised according to the AdacorOntology.

The WaitForMessagesBehaviour behaviour is a Java class that is waiting for the arrival of messages, using the *blockingReceive()* method to block the behaviour until a message arrives. The arrival of a message will trigger a new behaviour to handle the message.

The heterogeneous manufacturing environment requires that the holons are implemented on a wide variety of interoperable control platforms over different operating system platforms. To show this ability, the holonic control prototype runs over the Windows 2000, Windows XP and Linux operating systems, with the holons spanned by several PCs.

6.2.3 Implementation of ADACOR Ontology

The ADACOR ontology was translated to Java classes according to the JADE guidelines that follows the FIPA specifications for the development of ontologies. The main class of the ADACOR ontology is illustrated by the following code extract:

```
public class AdacorOntology {
  public static final String RESOURCE = "RESOURCE";
  public static final String WORKORDER = "WORKORDER";
  ...
  private static void initInstance() {
    try {
      ...
      theInstance.addRole( // Adds WORKORDER role (concept)
        WORKORDER,
            new SlotDescriptor[] {
            new SlotDescriptor("woid", Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE,
                               Ontology.M),
            new SlotDescriptor("precedence", Ontology.PRIMITIVE_SLOT,Ontology.STRING_TYPE,
                               Ontology.O),
            ...
        }, WorkOrder.class);
        theInstance.addRole( // Adds RESOURCE role (concept)
          RESOURCE,
            new SlotDescriptor[]{
            new SlotDescriptor("resid", Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE,
                                Ontology.M),
            new SlotDescriptor("state", Ontology.PRIMITIVE_SLOT, Ontology.STRING_TYPE,
                                Ontology.O),
            ...
        },  Resource.class);
        ...
        theInstance.addRole(  // Adds ALLOCATED role (predicate)
          ALLOCATED,
            new SlotDescriptor[]{
            new SlotDescriptor(Ontology.FRAME_SLOT, RESOURCE, Ontology.M),
            new SlotDescriptor(Ontology.FRAME_SLOT, WORKORDER, Ontology.M)
        }, Alocated.class);
      }   // end of try
      ...
    } //end of initInstance
  }
```

In addition to this main class, it was necessary to develop one Java class for each concept and predicate defined in the ontology, which is also available in the AdacorOntology package. As an example, the Java class for the Resource concept is illustrated by the following code extract:

```
public class Resource {
   private String _resourceid;  // resource identification
   private String _state;       // state of the resource
   ...
   // Methods required to use this class that represents the RESOURCE concept
   public void setResourceId(String resourceid) {
      _resourceid=resourceid;
   }
   public String getResourceId() {
```

```
            return _resourceid;
        }
        public void setState(String state) {
            _state=state;
        }
        public String getState() {
            return _state;
        }
        ...
    }
```

The methods defined in each individual class, *get*() and *set*() methods in the previous example, allow to handle the data related to the object.

6.2.4 Decision-Making Mechanisms

The current ADACOR approach to the decision component, illustrated in Figure 6.7, uses declarative and procedural approaches to represent knowledge and to regulate the holons behaviour. The knowledge base of each ADACOR holon is dependent of its type, objectives, skills and behaviour.

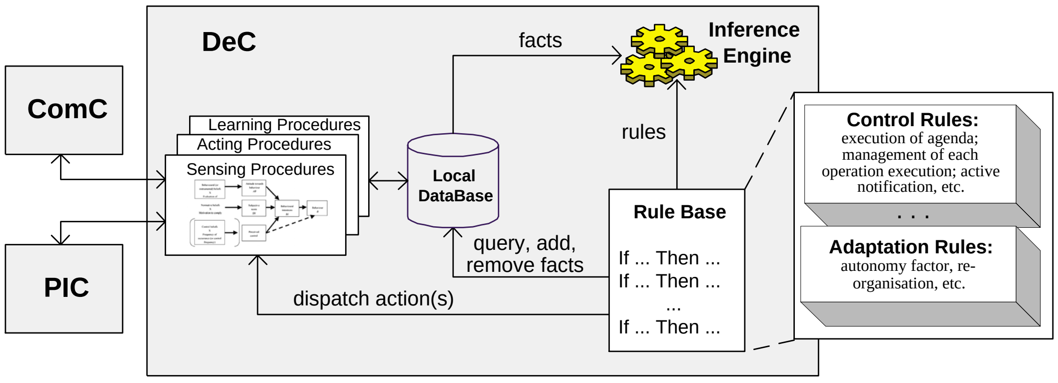


Figure 6.7: Decision Component Architecture

The crucial element in the decision component is the rule-based system, which applies declarative knowledge, expressed in a set of rules, to regulate the holon's behaviour. The advantage of this type of knowledge-based system is related to the simple and very comprehensive way to represent the reasoning capability of one holon. The simplicity and the associated high abstraction level of this approach compensates the typical weaknesses of these systems to handle with incomplete, incorrect and uncertain knowledge, and to implement complex systems, that require a large number of rules that can lead to a very slow system.

For this purpose, it is used the JESS tool, which is a rule oriented programming infrastructure based in the CLIPS (C Language Integrated Production System) language and uses the Rete algorithm as inference engine [Friedman-Hill, 1999]. JESS handles data structures, functions and rules, requiring the use of a *clp* file to store the application knowledge base.

Each ADACOR holon class has a *clp* file containing its knowledge base. The decision mechanisms that are common to all the ADACOR holons classes, such as the active notification, are placed in a special common *clp* file. The implementation of the knowledge base comprises the definition of the domain declaration and the behavioural rules.

The domain declaration part defines the objects referred in the behavioural rules and in the short-term memory. The objects (that can be physical objects or concepts) are defined by a set of attributes that represents the properties of the object.

Figure 6.8 illustrates an extract of the knowledge base definition. As an example, the object WorkOrder is defined in the domain declaration part as having the following attributes: *woID* is the code of the work order, *state* is the current state of the work order, *precedent* is the precedence of the work order, *resourceName* is the name of the resource that is responsible to execute the work order, and *location* is the location of the resource.

The objects defined in the domain declaration will be used in the local database component to store the facts that represent the short-term knowledge. This type of knowledge represents the current state of the holon at a particular moment.

The set of rules defined in the knowledge base represents the behaviour of the ADACOR holon. As an example, Figure 6.8 represents a behaviour rule defined in the task holon implementation. The rule has three conditions: the first condition is satisfied if the fact Transport is true; the second condition determines the name of the operation that is currently in execution; at last the third condition gets the information related to the operation in execution. When the rule is selected and executed, three actions are executed: the facts *fact1* and *fact3* are removed from the knowledge base; the second action is related to the modification of the state of the work order to TRANSPORT; the third action triggers a behaviour in the JADE environment that will be in charge to execute the transportation of the part, passing the name of the work order and the path for the transport operation.

As an example of how is implemented the invoked actions triggered by the rule-base mechanism, the JADE behaviour that will be responsible to execute the transport is a Java class called ExecuteTransport that has the following structure:

```
// This class is invoked by the Jess engine to execute a transport operation
class ExecuteTransport implements Userfunction {
  public String getName() { return "ExecuteTransport"; }
  public Value call(ValueVector v, Context context) throws JessException {
```

```
% knowledge base
% -----------------------------------------------------------
% domain declaration
(deftemplate WorkOrder         "List of jobs to be executed"
   (slot woID: string)
   (slot state: {NOT-ALLOCATED, ALLOCATED, TRANSPORT, LOADING, RUNNING})
   (slot precedence: string)
   (slot resourceName: string)
   (slot location: string)
)
(deftemplate Executing        "Stores the state of a task execution"
   (slot taskState: {FREE, EXECUTION}  (default FREE))
   (slot jobInExecution: string         (default nil))
)
...
% -----------------------------------------------------------
% behaviour rules
...

(defrule Transport         "Will start the transport of the part"
   ?fact1 <- (Transport)
   ?fact2 <- (executing (jobInExecution ?wo))
   ?fact3 <- (WorkOrder (woID ?wo)(state ?state)(resourceName ?res)
             (location ?location)(precedence ?precedent))
=>
   (retract ?fact1 ?fact3)
   (assert (WorkOrder (woID ?wo)(state TRANSPORT)(resourceName ?res)
           (location ?location)(precedence ?precedent)))
   (ExecuteTransport ?wo ?*transport-path*)
)
...
```

Figure 6.8: Example of the Task Holon Knowledge Base

```
ACLMessage msg_out = new ACLMessage(ACLMessage.INFORM);
ArrayList result=new ArrayList();
List l;
...
String woName = v.get(1).stringValue(context);
String transport_path = v.get(2).stringValue(context);
// fills the header of the message
msg_out.setSender(getAID());
msg_out.setLanguage(SL0Codec.NAME);
msg_out.setOntology(AdacorManufOntology.NAME);
// determines destination
String targetLocation=findDestination(transport_path);
// verifies if destination is the storage or another workstation
if (!targetLocation.equals("Storage")){
        // is a transport operation to a workstation
        // asks the OpH (workstation) if the part can be delivered
        ...
        try{
           fillContent(msg_out, l);
        }
        catch (FIPAException fe) {
           System.err.println("Exception in fillContent: " + fe.getMessage());
        }
        send(msg_out);
      }
      else {
        // is a transport operation to the storage
        // searches for holons that can execute the transport operation
        Vector list_types = new Vector();
        list_types.add("TRANSPORTER");
        result = findAvailableAgents(list_types);
        ...
        // sends msg to the transporter requesting the execution of a transport operation
        ...
        try{
           fillContent(msg_out, l);
        }
        catch (FIPAException fe) {
           System.err.println("Exception in fillContent: " + fe.getMessage());
        }
        send(msg_out);
        // sends a subscription to be notified when the work order is finished
        msg_subs.setSender(getAID());
        ...
        send(msg_subs);
      }
      return new Value(v.get(0).stringValue(context), RU.STRING);
   }
 } // end of the class ExecuteTransport
```

Another type of connection between the both control worlds (the rule-base using JESS and the agent-based application using JADE) is done by introducing commands lines embedded in the Java program, that invoke JESS commands to handle the stored knowledge (such as to make asserts of facts). The following extract of code illustrates this type of connection.

```
...
Rete decisionEngine = Rete();
...
decisionEngine.executeCommand("(assert (WorkOrder (woID " + value1 + ")
    (state " + value2 + ")(precedence " + value3 + ")))");
...
```

The previous command line, used in the Java program elaborated upon the JADE platform, allows to assert a new fact in the knowledge base. In this case, a new work order is introduced.

ADACOR holons uses also procedural knowledge to represent the holon's knowledge and behaviour. This type of knowledge is embodied in procedures, that are triggered as actions by some rules, each one being responsible for the execution of a particular set of actions. The scheduling algorithm is an example of this type of knowledge representation, being its choice dependent from the production system and production criteria.

In the prototype, the centralised scheduling mechanism, described in Appendix B, deals with the multiple machines and multiple jobs problem, and is a simple algorithm that guarantees a rapid and reliable scheduling. As the ADACOR architecture is built upon functional blocks, the centralised scheduling algorithm can be easily modified, by plugging more powerful scheduling algorithms, such as based on lagrangian relaxation or taboo search.

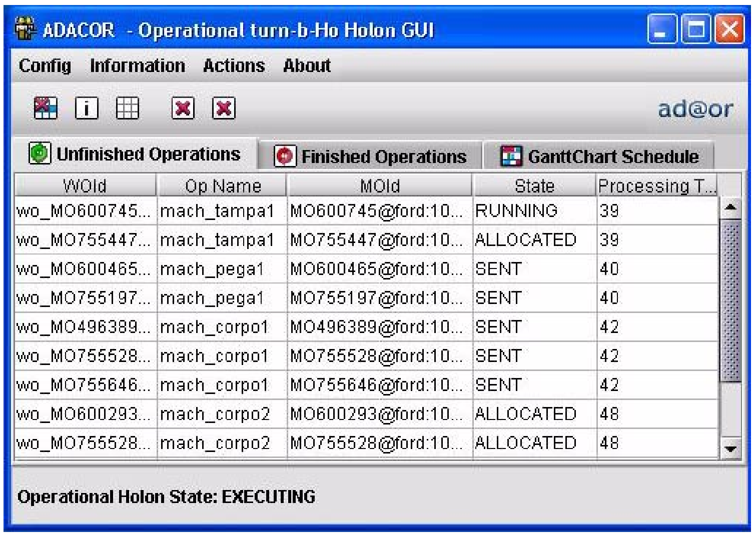
In a similar way, it was developed the required mechanisms to implement the distributed scheduling. For this purpose, it was developed a scheduling engine for each operational holon, that addresses the scheduling of multiple jobs to a single machine, based in scheduling heuristics, such as EDD and SPT. It was also developed the mechanisms for the elaboration of the price to execute a work order, to be proposed by an operational holon to a task holon, and for the evaluation of the proposals, both described in Appendix B.

Some procedures are related to the acquisition of information by handling the arrival messages from other holons or getting local information through the access to the physical manufacturing resource.

6.2.5 Graphical User Interfaces

In the ADACOR prototype, the operational and supervisor holons have graphical user interfaces to support the interaction with the user.

The graphical user interface for the operational holons is represented in Figure 6.9. This interface allows to visualise the local schedule using a Gantt chart, to visualise the work orders executed by this resource, to configure some operational holon parameters, such as the scheduler type and the activation of the autonomy factor, and to visualise statistical information related to



WOId	Op Name	MOId	State	Processing T...
wo_MO600745...	mach_tampa1	MO600745@ford:10...	RUNNING	39
wo_MO755447...	mach_tampa1	MO755447@ford:10...	ALLOCATED	39
wo_MO600465...	mach_pega1	MO600465@ford:10...	SENT	40
wo_MO755197...	mach_pega1	MO755197@ford:10...	SENT	40
wo_MO496389...	mach_corpo1	MO496389@ford:10...	SENT	42
wo_MO755528...	mach_corpo1	MO755528@ford:10...	SENT	42
wo_MO755646...	mach_corpo1	MO755646@ford:10...	SENT	42
wo_MO600293...	mach_corpo2	MO600293@ford:10...	ALLOCATED	48
wo_MO755528...	mach_corpo2	MO755528@ford:10...	ALLOCATED	48

Operational Holon State: EXECUTING

Figure 6.9: Graphical User Interface of an Operational Holon

the resource performance, such as the degree of utilisation, the number of work orders executed and the number of work orders delayed.

Each holon uses log files to store the relevant data during its life cycle, to support posterior analysis or to execute backup procedures in case of holon crash. In case of operational holons, this log file stores information about rejected, cancelled, failed and executed work orders.

The graphical user interface for the supervisor holon is represented in Figure 6.10.

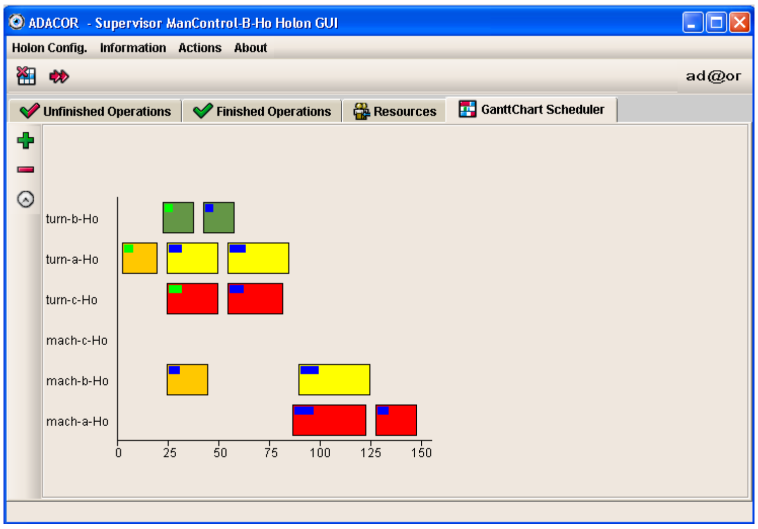


Figure 6.10: Graphical User Interface of a Supervisor Holon

This interface allows to visualise the global schedule using a Gantt chart, to visualise the work orders executed by each lower-level resource, to visualise the resources under its coordination domain and their characteristics, and to configure some holon parameters, such as the schedule algorithm.

The production of a product requires to launch a task holon, in order to supervise the execution of the product. The task holons does not have graphical user interface but uses a log file to store the relevant information associated to the execution of the production order. At the end of the production order execution, the task holon passes back the relevant information to the product holon and makes a statistical report about the performance of the execution of the production order, calculating, amongst others, the manufacturing lead time, tardiness, processing time and idle time.

6.2.6 Configuration of ADACOR Holons

The characteristics of each identified holon were configured using a XML (eXtensible Markup Language)-based configuration file. These characteristics are loaded to the holon behaviour during its startup, through the access and interpretation of the data stored in the configuration file.

Each ADACOR holon class requires a different type of configuration data.

In the product holon it is necessary to introduce the product data model and the process plan associated to the product, as illustrated in the Figure 6.11.

The description of the product structure is represented by a list of objects formatted according to a data structure, which includes the name of the sub-part, the number of parts necessary to produce the product and the estimated time to produce the part.

As the production of the product requires the execution of several operations over the parts, the process plan defines the sequence of operations that should be executed to produce the product. In this way, the process plan contains a list of operations formatted according to an appropriated data structure, which mainly describes the name of the operation, a brief description about the operation, the estimated time to execute the operation, the reference to the part, a list of requirements associated to the operation, and the name of the operations that have precedence over this operation.

Each operational holon that represents a physical resource has specific characteristics, which should be mapped in a XML-based configuration file, as illustrated in the Figure 6.12. The data structure represents the resource model, which attributes reflect mainly the type of resource, the list of skills that the resource possesses and its location. Another relevant information to the operational holon is the indication of the organisational file that contains the data related to the organisational control structure.

The organisational structure of the factory plant is defined in a XML-based configuration

```
< product >
    < product name > ... < /product name >
    < bom >
        < part >
           < part name > ... < /part name >
          < quantity > ... < /quantity >
          < time > ... < /time >
       < /part >
        ...
         < part >
             < part name > ... < /part name >
             < quantity > ... < /quantity >
             < time > ... < /time >
         < /part >
         ...
    < /bom >
    < process plan >
         < operation >
                < operation name > ... < /operation name >
                < description > ... < /description >
                < exectime > ... < /exectime >
                < part > ... < /part >
                < requirement  >
                    < name > ... < /name >
                    < value > ... < /value >
                < /requirement >
                ...
                < quantity > ... < /quantity >
                < precedence > ... < /precedence >
         < /operation >
         ...
    < /process plan >
< /product >
```

Figure 6.11: XML-based Product Data Model

```
< resource >
    < resource name > ... < /resource name >
    < model > ... < /model >
    < supplier > ... < /supplier >
    < type > ... < /type >
    < organisational file > ... < /organisational file >
    < skills >
        < name > ... < /name >
        < value > ... < /value >
    < /skills >
    ...
    < location > ... < /location >
< /resource >
```

Figure 6.12: XML-based Resource Data Model

file that comprises the information related to the cell organisation and to the shop layout, as illustrated in the Figure 6.13. This organisational structure describes the possible manufacturing cells and associated coordinator entities, which will be converted into supervisor holons. With this organisational structure XML-based file, the operational holons can find their supervisor holons and their auxiliary resources. Also, the supervisor holon can find the list of holons that are in its coordination domain.

```
< factory organisation >
    < coordinator > ... < /coordinator >
    < cell description >
        < cell >
            < cell name > ... < /cell name >
            < location > ... < /location >
            < coordinator > ... < /coordinator >
            < main resource >
                < resource name > ... < /resource name >
            < /main resource >
            < auxiliar resource >
                < auxiliar name > ... < /auxiliar name >
              ...
            < /auxiliar resource >
        < /cell >
        ...
    < /cell description >
    < cell distances >
        < distance >
            < source > ... < /source >
            < target > ... < /target >
            < value > ... < /value >
        < /distance >
        ...
    < /cell distances >
< /factory organisation >
```

Figure 6.13: XML-based Organisational Structures Model

A manufacturing resource can play two different roles in the organisational structure: (1) a main resource, acting as a principal resource, for example a CNC machine or an AGV, or (2) an auxiliary resource, which is a resource that collaborates with a main resource, such as a loading/unloading robot.

The configuration file defines also the location of each cell and the distances between the cells. In case of transporters resources, these values can be used to determine the costs of the material transport operations.

6.3 Experimental Case Study

An experimental case study has been used to validate the ADACOR architecture concepts. This validation has two different objectives:

- Validate the implementation, to verify if the system works as it was specified, either in normal operation or in presence of disturbances.
- Evaluate the performance, to conclude about the merits of the proposed concepts.

The case study used in this work will be described according to the framework defined by [Cavalieri et al., 1999]. This benchmark framework for manufacturing control defines three design issues that must be properly specified to define the case study. The three axes are [Cavalieri et al., 1999]:

- **Production system**, which provides a description of the structural and technological features of the case study, focusing on the specification of the static features.
- **Manufacturing scenarios**, which describe the dynamic features.
- **Performance measures**, which defines the criteria for evaluating the different indicators used to evaluate the performance of a single implementation.

Each one of these vectors will be described in the next sections.

6.3.1 Production System

The description of the production system that will be used as case study comprises the definition of the structural features that describe the physical configuration of the production system (i.e. the physical resources and the layout), and the technological features that describes the process plans of the available products at the factory plant.

The production system layout, illustrated in Figure 6.14, comprises the flexible manufacturing system of the IDIT[9]'s platform, extended with two virtual manufacturing cells, that will provide the flexibility in achieving alternative production routings. The flexible manufacturing system of IDIT's platform, which was already used in the past to implement a manufacturing cell controller based in a hierarchical control approach [Leitão, 1996, Quintas and Leitão, 1997], is organised as a set of four physical cells: material storage and transportation cell, inspection cell, assembly cell and flexible manufacturing cell.

The inspection cell is responsible for the execution of inspection, maintenance, setup and recovery operations, assembly of tools for the CNC machines, calibration of tools and grippers, and palletising and depalletising of the materials that circulate in the shop floor. This cell is constituted by human operators, a calibrate machine AR2000GA from Elbo Controlli, a palletising table and several equipments to support the maintenance operations.

The material storage and transportation cell is responsible for the transportation of materials within the shop floor and for the temporary storage of materials (that could be raw materials,

[9]IDIT (Instituto de Desenvolvimento e Inovação Tecnológica) inherited the flexible manufacturing system platform implemented at CIM Center of Porto under the ESPRIT project EP-5629.

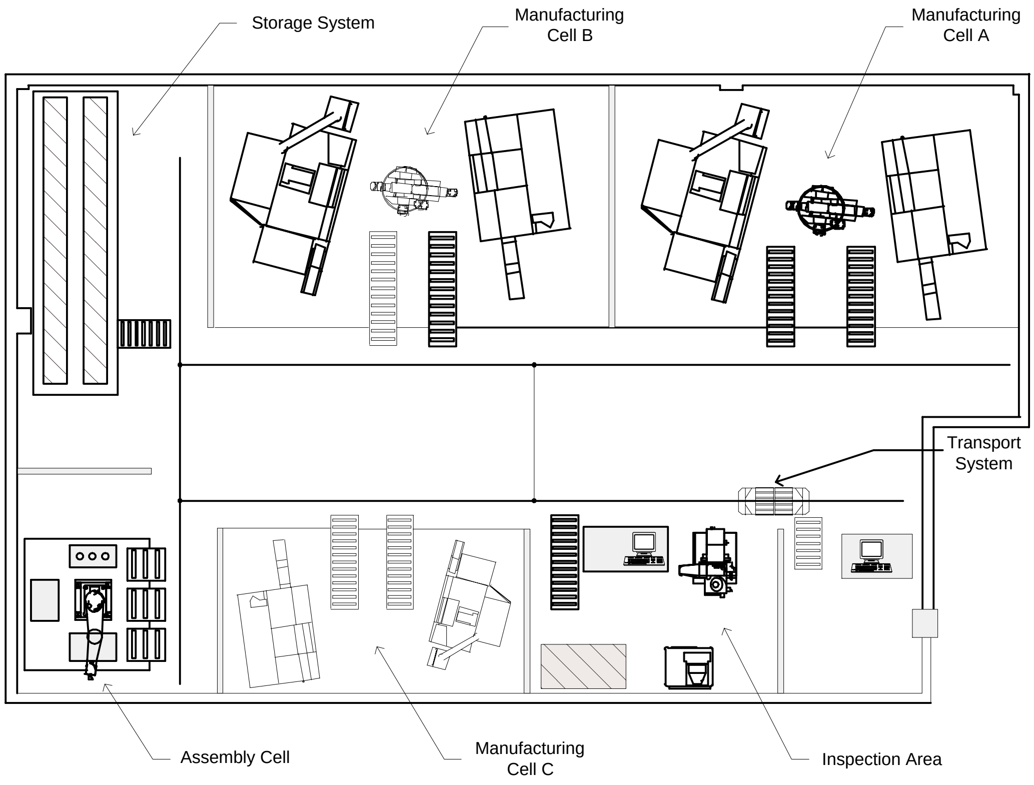


Figure 6.14: Plant Layout of the Case Study Production System

semi-finished products or final products). This cell has an AGV EFAGV-200-2R-B and an AS/RS system, both supplied by Efacec, Automação e Robótica, SA. The presence of an AGV allows to support variable routing of the products flow.

The assembly cell has the objective to assembly components to achieve the final products. This cell has a four-axis SCARA robot Adept Three from Adept Technology. Coupled to the robot, exists a CCD camera from PULNIX, associated to the artificial vision system Cognex 4200EX from Cognex Corporation.

The flexible manufacturing cell A has two CNC machines and an anthropomorphic robot for the load/unload of these machines. One of these machines is a turning center Lealde TCN10, with a SIEMENS Sinumerik 880T controller and equipped with a tool magazine for 12 tools. The other machine is a milling center Kondia B500, with a FANUC 16MA numerical control, and equipped with a tool magazine for 18 tools. The robot is a KUKA IR163/30.1 with a SIEMENS RC3051 controller, supported by a gripper magazine with capacity to four grippers.

The test platform will consider two additional virtual functional areas, designated by areas B and C, which intend to provide alternative machines for the execution of the processing operations, aiming to test the response of the control system to the occurrence of unexpected disturbances. The area B comprises a milling machine and a turning machine with the same characteristics of the machines of the area A. Additionally, it comprises an ABB IRB1400 load/unload robot. The area C comprises a turning machine similar to the others turning machines and a drilling machine that does not exist in any other functional area.

Each machine has a set of tools (such as cutting or inspection tools) that guarantees the capabilities to execute different types of operations. The execution of setups is not considered in this case study, being assumed that the tools are permanently stored in each machine. Thus, the turning machine of the area A has the tA, tB and tC tools type, and the milling machine of the area A has the tV and tH tools type. The turning machine of the area B has the tB, tC and tD tools type, and the milling machine of the area B has tX, tY and tZ tools type. At last, the turning machine of the area C has tB, tC and tD tools type, and the drilling machine has the tK and tW tools type.

The production system contains other types of resources, namely buffers and containers. Each machine has its input/output buffer, which is a de-coupling point in face of the transport system. The containers bring the material to be processed in the machine or the cell and take away the pieces produced. An intermediate buffer (located as a part of the main storage facility) is considered with infinity capacity to store the work in progress.

In the factory plant, four different products can be produced, named *Base*, *Body*, *Cover* and *Handle*, which assembled can create two different final products: *Box* and *Ashtray*. The *Ashtray* product comprises the assembly of the *Base* and the *Body* sub-products, and the *Box* product comprises the assembly of all designed sub-products.

Each product has a process plan constituted by non-preemptable operations according to the Figure 6.15. The flexibility in this flexible manufacturing system is achieved by the alternative routings plans according to the on-line factory capacity.

The specification of the process plan for each product involves, amongst others, the definition of the raw material, the list of operations, and the precedences between operations. Each operation is characterised by a brief description, the required type of machine that can perform the operation and the estimated execution time.

The description of the cutting tools in the requirements of each operation uses an abbreviate notation, in order to simplify its analysis. As an example, the tool tD is a roughing cutting tool with the following characteristics: a PCLNL2020K12 rigid clamp, and a CNMG120404-QM415 insert.

Product name: HANDLE		Material: d = 45 mm, l = 70 mm	
Op Id	**Operation**	**Requirements**	**Mfg time**
op-p1	Turning of side 1	{type,turn}, {axes,3}, {tools,(tD,tC)}	40
op-p2	Turning of side 2	{type,turn}, {axes,3}, {tools,(tD,tC)}	50
op-p3	Execution of the drills	{type,drill}, {axes,4}, {tools,(tK)}	63

Product name: COVER		Material: d = 95 mm, l = 40 mm	
Op Id	**Operation**	**Requirements**	**Mfg time**
op-t1	Turning of side 1	{type,turn}, {axes,3}, {tools,(tB,tC)}	39
op-t2	Turning of side 2	{type,turn}, {axes,3}, {tools,(tB,tC)}	51
op-t3	Execution of the flower face	{type,mill}, {axes,5}, {tools,(tX,tY)}	119
op-t4	Execution of the drills	{type,drill}, {axes,4}, {tools,(tK)}	61

Product name: BODY		Material: d_e = 95 mm, d_i = 75 mm, l = 60 mm	
Op Id	**Operation**	**Requirements**	**Mfg time**
op-c1	Turning of side 1	{type,turn}, {axes,3}, {tools,(tD,tC)}	42
op-c2	Turning of side 2	{type,turn}, {axes,3}, {tools,(tD,tC)}	48
op-c3	Execution of the flower face	{type,mill}, {axes,5}, {tools,(tX,tY)}	125

Product name: BASE		Material: d = 90 mm, l = 35 mm	
Op Id	**Operation**	**Requirements**	**Mfg time**
op-b1	Turning of side 1	{type,turn}, {axes,3}, {tools,(tB,tA)}	35
op-b2	Turning of side 2	{type,turn}, {axes,3}, {tools,(tB,tA)}	55

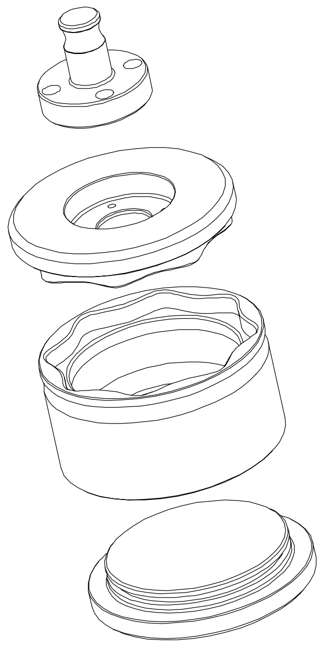


Figure 6.15: Process Plans for the Available Products in the Production System

6.3.2 Manufacturing Scenarios

The definition of manufacturing scenarios comprises the description of the plant, operational and control scenarios.

Plant Scenarios

The plant scenario describes the aspects related to the functioning of the physical resources, such as setup times, transportation times and availability, and disturbance models.

In this case study, it is considered that:

- No setups will be executed, since each machine is equipped with the required tools to execute a range of operations.
- The transport operations are performed by a single AGV and orders are queued by order of arrived. The execution of each transport operation takes 5 seconds.
- Work orders are executed in the exact estimated manufacturing time.

The experimental scenario considers three plant scenarios:

- The first plant scenario considers that no unexpected disturbance will occur.

- The second plant scenario considers the introduction of machine failures, associated to a turning machine; there is one failure every four operations, the part is destroyed and the machine is breakdown during 60 seconds for the recovery procedures.

- The third plant scenario considers the same disturbance model as in previous scenario, but now applied simultaneously to two distinct turning machines.

Operational Scenarios

The operational scenario defines the order mix. Each individual book of orders comprises the production of 6 production orders, distributed by 2 bodies, 2 bases, 1 handle and 1 cover, involving 17 operations (see Figure 6.15).

Different operational scenarios are achieved by considering different sets, consisting of one, two, three and four books of orders. As an example, the scenario comprising 4 individual books of orders, will be constituted by the production of 24 production orders, distributed by 8 bodies, 8 bases, 4 handles and 4 covers, amounting to 68 operations.

Each operational scenario considers that all the production orders belonging to the same book of orders arrive to the production system at the same time, but different books of orders arrive sequentially to the production system.

Control Scenarios

The proposed ADACOR control approach performance is compared with other two different control approaches, using the same implemented prototype: heterarchical-like and hierarchical-like control approaches.

In the hierarchical-like control approach, the holons are organised in a hierarchical control structure, using the supervisor holon as the shop floor controller.

In heterarchical-like control approach, the holons run on a completely decentralised control structure, without the presence of supervisor holons. In this approach, the task and operational holons interact directly for each (re-)scheduling decision.

In the ADACOR holonic control approach, the holons are organised in a hierarchical control structure, using the supervisor holon to act as shop floor controller and enabling the autonomy factor of each operational holon to support the agile re-organisation of the control structure in case of emergency.

During the experimental tests, both local and centralised scheduling algorithms are tuned to minimise the lead time by using the SPT heuristic.

6.3.3 Prototype Holonification

The development of the holonic control application for the case study requires the holonification of the manufacturing components, i.e. the identification and implementation of the manufacturing holons, from the elements present in the case study. In the next sections, the manufacturing holons will be identified, and the specific components will be developed.

Identification of Manufacturing Holons

At this stage, it is necessary to identify the operational, product and supervisor holons in the case study production system.

The first step is to find the set of resources available at the shop floor. Analysing the description of the production system, the following resource elements were found, forming the $\mathcal{R}$ set: h_1 (human operator), t_1 (AGV device), t_2 (AS/RS system), to_i (several tools available in the factory), m_1 (assembly robot), a_1 (CCD camera), pm_1 (turning machine of area A), pm_2 (milling machine of area A), m_2 (handling robot of area A), pm_3 (turning machine of area B), pm_4 (milling machine of area B), m_3 (handling robot of area B), pm_5 (turning machine of area C), pm_6 (drilling machine of area C), and a_i (several buffers available in the factory plant).

Not all the identified resources are candidates to be associated to holons. Analysing the physical dependencies between the identified objects, it is possible to verify that:

- The CCD camera is dependent of the assembly robot, being possible to aggregate both resources.
- The tools are proprietary of the machines and not shared by the several physical resources, allowing to aggregate each tool to the appropriated CNC machine.
- The existing buffers are not considered in the final set of resources, since it is assumed that each buffer serves only one machine.

After the analysis of the physical dependencies, it is possible to define the set $\mathcal{R}'$ containing the aggregated components: h_1, t_1, t_2, m_1, pm_1, pm_2, m_2, pm_3, pm_4, m_3, pm_5 and pm_6. Each element of the set $\mathcal{R}'$ is mapped into an operational holon, configured by means of a XML-based file with the description of its main characteristics. Table 6.2 illustrates the main skills of each operational holon, focusing on its type and set of tools.

The product holons are the available products at the factory plant. There are six product holons: $box-Ho$, $ashtray-Ho$, $body-Ho$, $base-Ho$, $handle-Ho$ and $cover-Ho$, the first two being assemblies of the remaining ones. For each one of these product holons, a configuration

Table 6.2: List of Skills of Each Operational Holon

$\mathcal{R}_i$	**Description**	**OpH**	**Skills**
h_1	Human operator	$operator - Ho$	{type,maintenance}
t_1	AGV device	$transp - Ho$	{type,transporter}
t_2	AS/RS system	$storage - Ho$	{type,buffer}
m_1	Assembly robot	$assembly - Ho$	{type,assembler}
pm_1	Turning of area A	$turn - a - Ho$	{type,turn}, {axes,3}, {tools,(tA,tB,tC)}
pm_2	Milling of area A	$mach - a - Ho$	{type,mill}, {axes,5}, {tools,(tV,tH)}
m_2	Robot of area A	$robot - a - Ho$	{type,handle}
pm_3	Turning of area B	$turn - b - Ho$	{type,turn}, {axes,3}, {tools,(tB,tC,tD)}
pm_4	Milling of area B	$mach - b - Ho$	{type,mill}, {axes,5}, {tools,(tX,tY,tZ)}
m_3	Robot of area B	$robot - b - Ho$	{type,handle}
pm_5	Turning of area C	$turn - c - Ho$	{type,turn}, {axes,3}, {tools,(tB,tC,tD)}
pm_6	Drilling of area C	$drill - c - Ho$	{type,drill}, {axes,4}, {tools,(tK,tW)}

file was elaborated, containing the information related to the product structure and the process plan.

The identification of supervisor holons can be done by analysing the description of the hierarchical levels in the platform control. In this case, only one supervisor holon for the shop floor control was considered ($FactoryControl - Ho$).

Implementation of Virtual Resources

The implementation of operational holons that represent physical manufacturing resources requires the development of wrappers interfaces, supporting the integration of those resources. In the ADACOR approach, the virtual resource concept was introduced to make transparent the intra-holon interaction.

The development of a virtual resource for each manufacturing device that belongs to the case study, encompasses the implementation of the services at the server side, that will be invoked on the client side (PIC component from the operational holon). The client ignores the details of this implementation and each virtual resource can be re-used by other similar resources or holonic control applications.

[Leitão et al., 2003a] describes the implementation of two different virtual resources to integrate two different automation resources, one for a PLC and another one for an industrial robot. These two virtual resources implement the same services in a different way according to the particularities of each resource. In the client side, whatever the resource to be accessed, the invocation is made as an unique way and looks like the following command line for the case of a *start* service:

```
int ret = resource.start(programName);
```

where *resource* is the identifier of the virtual resource that represents the real automation device that is intended to access.

In this section and aiming to test the virtual resource concept in real manufacturing scenarios, the virtual resource for the load/unload robot presented at area B will be described. The services provided by the virtual resource were developed using the RobComm ActiveX supplied by ABB [RobComm, 1999], and accessed through TCP/IP. The major problem was how to access to the ActiveX from a Java program, since the ActiveX components are adequate to be manipulated by Windows-based programming environments. To overcome this problem, it was used the JIntegra[10] tool to convert the ActiveX component into a Java package [Leitão et al., 2003a].

The development of the virtual resource for the load/unload robot, is illustrated by the description of the implementation of the *start* and *read* services. The implementation of the *start* service is summarised bellow.

```
public int start(String progName){
   ...
   try {
      returnCode=h.s4Run();
      returnCode=h.s4ProgramLoad(prgID,progName);
      returnCode=h.s4Start(prgID,procedure,nOfCycle,runMode);
   }
   catch(IOException ioe){returnCode=determineErrorCode();}
   return (returnCode);
}
```

This service comprises the execution of three commands: s4Run that turns on the robot motors, the s4ProgramLoad that loads a specified program to the robot controller and the s4Start that starts the execution of the loaded program. The service returns null in case of success or a positive integer in case of an error. The value of the error is determined using the method determineErrorCode, since the *start* service involves the execution of three actions over the physical resource, with possibility to be failed in any one of those actions.

Another example is related to the implementation of the *read* service in the virtual resource, which associated code is summarised bellow.

[10]JIntegra is provided by Intrinsyc. More information can be obtained at http://j-integra.intrinsyc.com/

```
public int read(String var,String type) {
   String[] vname=new String[1];
   vname[0]=var;
   short[] progNo=new short[1];
   progNo[0]=0;
   short varvalue=0;
   ...
   try {
      varvalue=h.s4ProgramNumVarRead(vname,progNo);
   } catch(IOException ioe) {System.out.println("Problem: " + ioe);}
   return ((new Short(varvalue)).intValue());
}
```

The input parameters of the *read* service are the *var* parameter which represents the variable name and the *type* parameter which represents the type of the variable. In this case, after the declaration of the variables, the method s4ProgramNumVarRead, provided by the RobComm ActiveX, is executed. The *read* service returns an integer value.

The platform used to support the client-server interaction was the CORBA. The analysis of the experimental implementation of the resource integration, by comparing the performances of CORBA, RMI and RMI-IIOP, is described in [Leitão et al., 2003a].

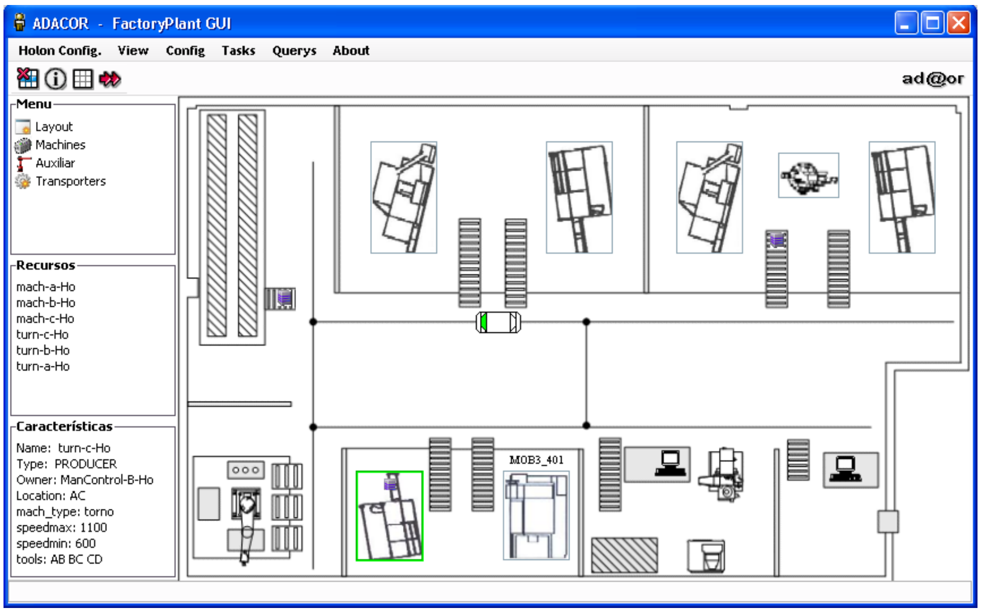
Auxiliar Tools

During the implementation of the ADACOR prototype, it was developed a set of auxiliar tools to support the configuration, operation and debugging of the manufacturing control application, providing functionalities to configure products and to supervise the factory plant in an integrated and global view.

As the global visualisation of all activities in the factory plant is difficult due to the distribution of graphical user interfaces, the ADACOR Factory Plan Supervisor (AFPS) is used to monitorise, in an integrated way, the production activities in the factory plant, which graphical user interface is represented in Figure 6.16.

This tool allows to visualise the production process by enabling the visualisation of the manufacturing resources present in the factory plant, indicating their state and characteristics. In this tool, the visualisation of the transport resource has animation capabilities to help the understanding of the material flow in the factory plant, indicating the direction of the movement and its actual load.

In order to support the introduction of new products in the system and to launch production orders to the factory plant, it is used the ADACOR Product Manager (APM) agent, which

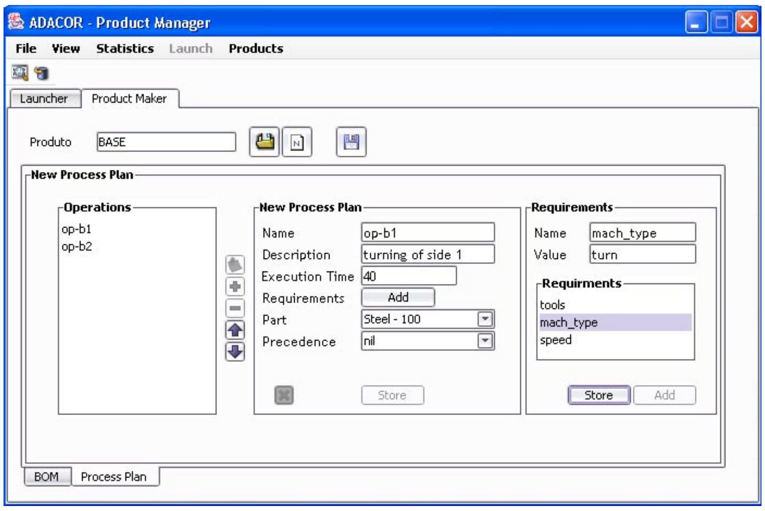

ADACOR - FactoryPlant GUI
Holon Config. View Config Tasks Querys About
ad@or
Menu
Layout
Machines
Auxiliar
Transporters
MOB3_401

Recursos
mach-a-Ho
mach-b-Ho
mach-c-Ho
turn-c-Ho
turn-b-Ho
turn-a-Ho

Características
Name: turn-c-Ho
Type: PRODUCER
Owner: ManControl-B-Ho
Location: AC
mach_type: torno
speedmax: 1100
speedmin: 600
tools: AB BC CD

Figure 6.16: Graphical User Interface of the ADACOR Factory Plant Supervisor

graphical user interface is represented in the Figure 6.17.



Produto: BASE

Operations: op-b1, op-b2

Name	op-b1
Description	turning of side 1
Execution Time	40
Requirements	Add
Part	Steel - 100
Precedence	nil

Name	mach_type
Value	turn

Requirments: tools, mach_type, speed

Figure 6.17: Graphical User Interface of the ADACOR Product Manager

This tool allows to define new products, by introducing the structure of the product and the process plan that defines the sequence of operations to execute the product. It also allows to launch individual or pre-defined sequences of production orders, making easier the execution of experimental tests.

6.3.4 Experimental Plan

The flexible manufacturing platform used as case study involves a significant number of different machines from different vendors and equipped with different controllers. The use of the real system is impracticable due to the long development time associated to the development of virtual resources for each individual machine. Additionally, it would also be impossible to reproduce the same conditions for a number of experimental tests when trying to compare alternative control systems [Saint-Germain et al., 2003].

One way to overcome this problem it to use a emulation platform that behaves like the real system, Figure 6.18. For the control system it is indistinguishable to be connected to the emulation platform or to the real system.

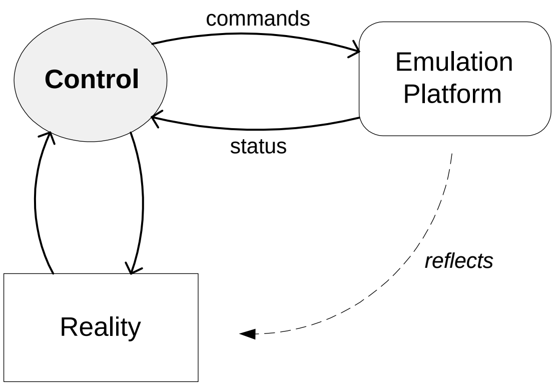


Figure 6.18: Emulation and Real System (Adapted from [Saint-Germain et al., 2003])

In the developed emulation platform, a resource emulator is used to imitate each machine presented in the case study production system, using finite state machines.

The experimental tests executed to validate the ADACOR architecture were done by testing the three control approaches under the defined manufacturing (plant and operational) scenarios, executing 6 experiences for each one.

Initially, an operational scenario comprising 3 book of orders corresponding to 18 production orders was fixed, to test all the possible combinations of control scenarios and plant scenarios, measuring the lead time, throughput, repeatability of resource utilisation, tardiness, and agility.

Then, for two selected plant scenarios (one scenario with no disturbances and one scenario with disturbances), all possible combinations of control and operational scenarios were tested, measuring the lead time and throughput. With this set of experimental tests it is possible to analyse the relation of the performance indicators with the manufacturing load.

6.4 Results

The results obtained from the experimental tests, using the plant and operational scenarios described previously, are presented and discussed in the next sections.

6.4.1 Evaluation of Quantitative Performance Indicators

The first set of experimental tests evaluates the behaviour of the ADACOR control approach, by comparing its performance with the hierarchical-like and heterarchical-like control approaches, focusing in the analysis of the quantitative indicators.

The experimental test was done using 18 production orders (51 operations), returning the time required to execute the package of orders, the manufacturing lead time, the tardiness and the resource utilisation.

Scenario with no Disturbances

In a scenario without the presence of unexpected disturbances, the system operates in a predictable context. The results of this experimental test are summarised in the Figure 6.19. As previously described, in these stable scenarios the holons of the ADACOR control approach are organised in a hierarchical structure, presenting the same behaviour as the hierarchical-like control, therefore showing the same experimental values.

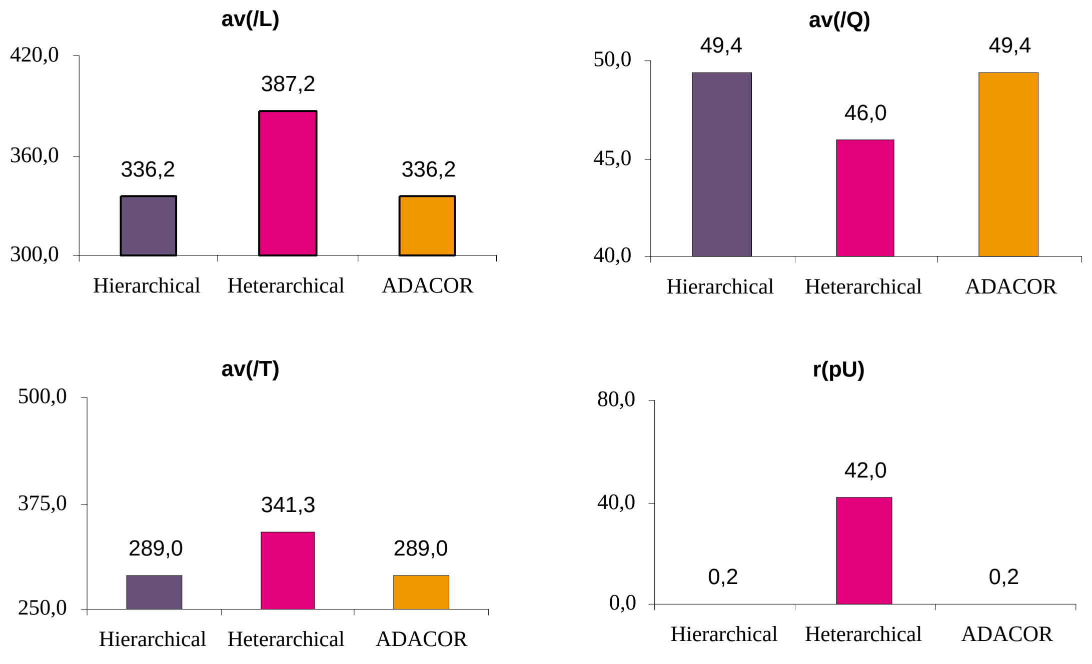


Figure 6.19: Performance of Evaluated Control Approaches for Scenarios with no Disturbance

Analysing the results obtained in this experimental test, it is possible to verify that in scenarios with no disturbance the hierarchical-like and ADACOR control approaches present smaller values of mean manufacturing lead time (336,2) and higher values of the throughput (49,4) than the heterarchical-like control approach (respectively 387,2 and 46,0).

The better performance presented in those approaches results from the better production planning achieved by the centralised entities, i.e. a supervisor holon that elaborates optimised production plans.

The predictability of each control approach, which gives an indicator about how much the control approach is repetitive and predictive, can be verified by analysing the $r(pU)$ parameter, which is the standard deviation of the percentage of utilisation of each resource over the several experiences belonging to the experimental test, and is a common indicator of the repeatability of the production planning.

Analysing this parameter it is clear that the predictability in hierarchical-like and ADACOR control approaches is better than in the heterarchical-like control approach. In fact, in the heterarchical-like control approach the global schedule is achieved by the interaction of operational holons that have a partial view of the entire system, making more difficult the global optimisation.

The type of production has also strong impact in the degree of production optimisation achieved. It was verified that for operations with small processing times, the better performance and global optimisation presented at hierarchical-like and ADACOR control approaches is less visible, while in cases of operations with long processing time, the non-optimised schedules presented in the heterarchical-like control approach lead to even worse optimisation and performance. This is due to the fact that long processing times are more sensitive to weak global optimisation.

Disturbance Scenario

The second experimental test considers the occurrence of unexpected disturbances, according to a probabilistic disturbance model which defines that only in the *turn-b* machine can occur failures, with a probability of 25%. In case of failure, the part is destroyed being necessary to re-allocate all the work orders to execute the product; additionally, the machine becomes unavailable during 60 seconds. The results obtained in this experimental test are summarised in the Figure 6.20.

The first conclusion extracted from these experimental results is the degradation of performance in the presence of disturbances. This degradation affects all the performance indicators.

Analysing the lead time and throughput parameters, it is possible to verify that the hierarchical-

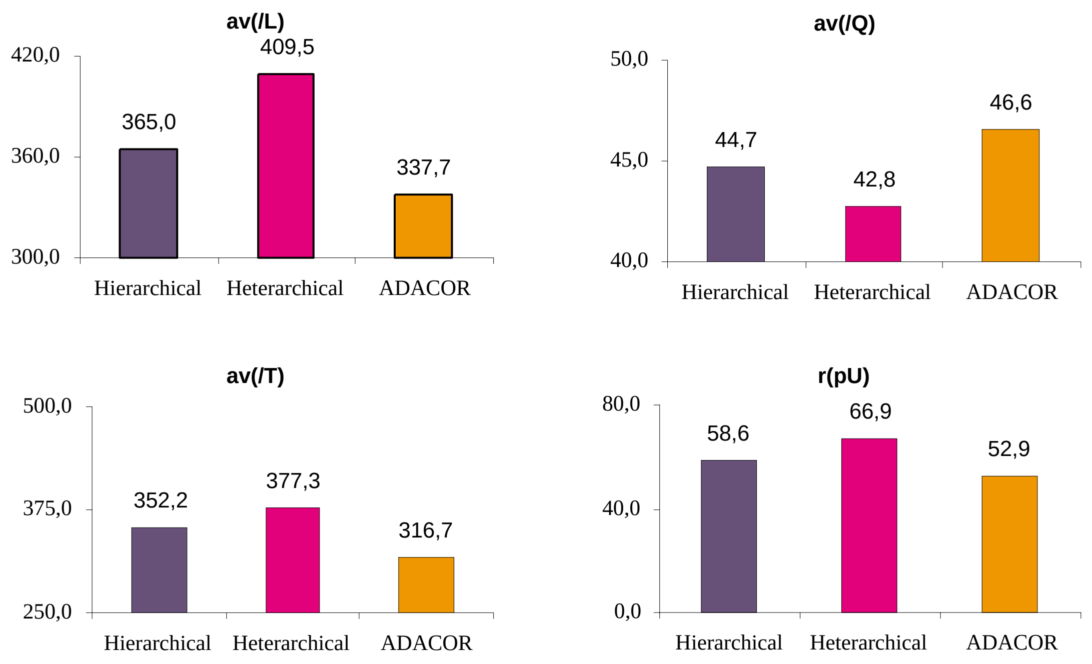


Figure 6.20: Performance of Evaluated Control Approaches for Disturbance Scenarios

like control approach still presents better performance than the heterarchical-like control approach. However, it is possible to verify that the difference of performance between hierarchical-like and heterarchical-like control approaches has been reduced significatively, specially in terms of the throughput parameter (reduction of 8,7% for the lead time and 55,9% for the throughput).

The proposed ADACOR holonic control approach presents promising performance results, since it shows better response to the disturbance scenario, illustrated by smaller value of manufacturing lead time (337,7) and higher value of throughput (46,6), than the hierarchical-like and heterarchical-like control approaches.

The occurrence of disturbances increases the entropy and unpredictability of the control system, degrading its predictability. It was verified that in disturbance scenarios the differences between the predictability exhibited by the several evaluated control approaches are smaller (i.e. between 52,9 to 66,9).

The second disturbance model was used to compare the response of the three control approaches to the different levels of entropy caused by the occurrence of disturbances. The disturbance model is similar to the previous one, but now applied to two turning machines, $turn-b$ and $turn - c$. Figure 6.21 illustrates the results obtained during the execution of this experimental test.

The experimental results associated to this disturbance model confirm the observations done

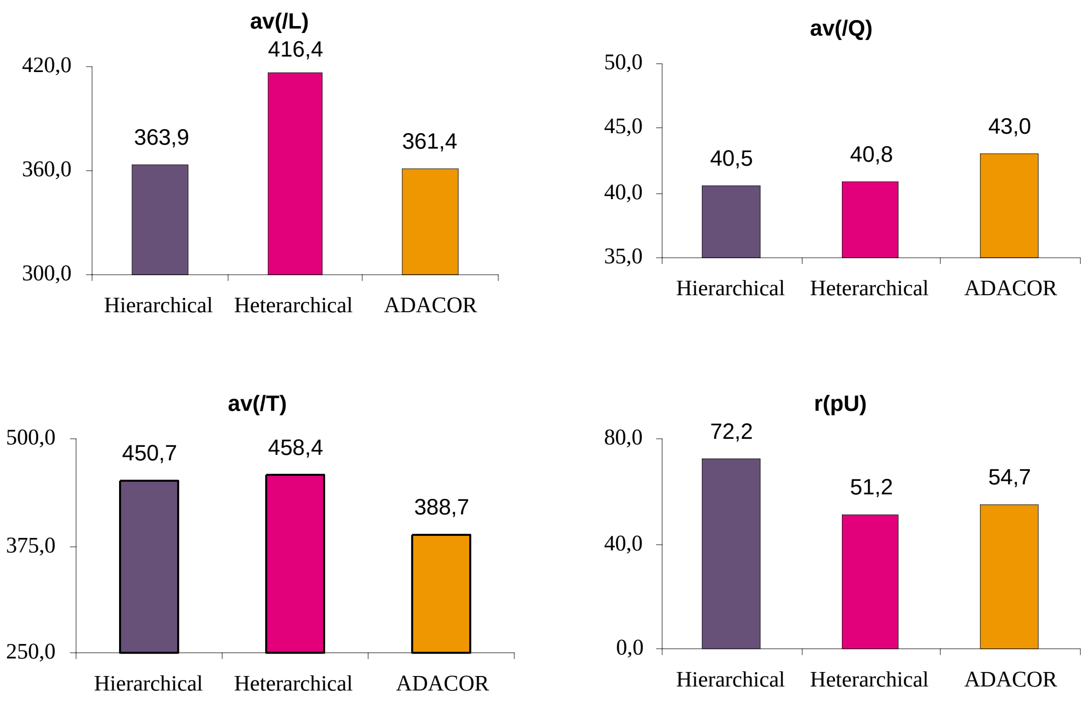


Figure 6.21: Performance of Evaluated Control Approaches for the 2nd Disturbance Model

during the previous experimental test.

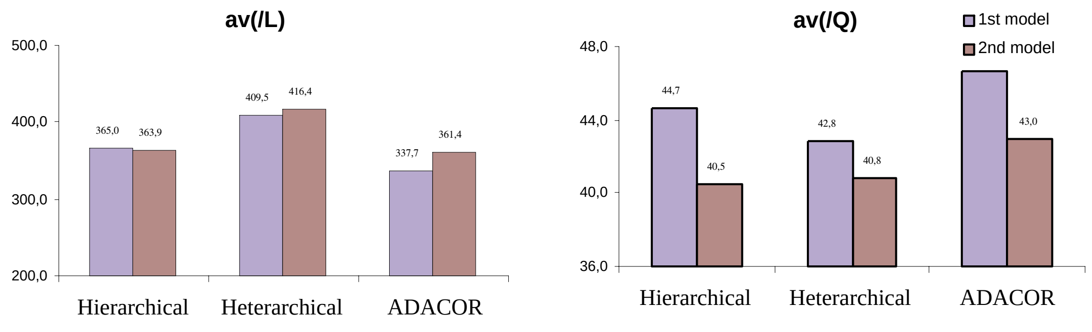


Figure 6.22: Comparison of the Performance According to Different Disturbance Models

As illustrated in the Figure 6.22, that shows the comparison of the manufacturing lead time and throughput parameters achieved by each control approach in both disturbance scenarios, it is also clear that the performance of each control approach suffers with the increase of entropy associated to the disturbance model.

Analysis of Resource Utilisation

The analysis of the resource utilisation reflects how the control system can distribute the load by the available resources in the factory plant. For the purpose of this analysis, only the three turning machines ($turn - a$, $turn - b$ and $turn - c$) were considered, since they are alternatives to execute the operations belonging to the package of available products. The average and the standard deviation of the percentage of utilisation of these turning machines, for the stable and first disturbance scenarios are summarised in the Figure 6.23.

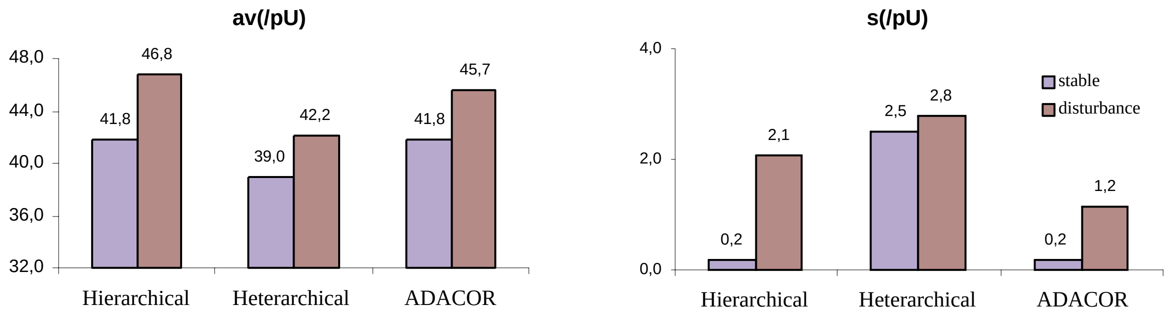


Figure 6.23: Experimental Results of the Resource Utilisation

Analysing the experimental results, it is possible to verify that the hierarchical-like and ADACOR control approaches present higher percentage of resource utilisation than the heterarchical-like control approach, either in the stable and disturbance scenarios, which demonstrates that they present better production plan optimisation. It is also observed that the percentage of resource utilisation is higher in disturbance scenarios than in the stable scenarios. This fact is justified by the execution of additional work orders launched to the shop floor, after the occurrence of machine failures that destroyed the part.

The analysis of the standard deviation of the percentage of resource utilisation, which gives an idea about how the planning and control system distributes the load by the available resources, allows to verify that the heterarchical-like control approach presents the worse behaviour in the load distribution, as expected. On the other hand, the ADACOR control approach presents even smaller values for the standard deviation than the hierarchical-like control approach.

6.4.2 Evaluation of Qualitative Performance Indicators

In this section, the ADACOR control architecture is evaluated by analysing a qualitative performance parameter, the agility.

The method to measure the agility of a manufacturing control system requires the continuous measurement of the throughput. In this experimental test, this method was not used, due to its complexity. In alternative, it was evaluated the loss of productivity parameter that reflects how agile the control system is.

The loss of productivity of each control approach is calculated by the ratio between the reduction in throughput and the throughput measured in the stable scenario. Additionally, the same procedure was used to calculate the loss of productivity in case of the second disturbance model. The comparison of the loss of productivity values is illustrated in the Figure 6.24.

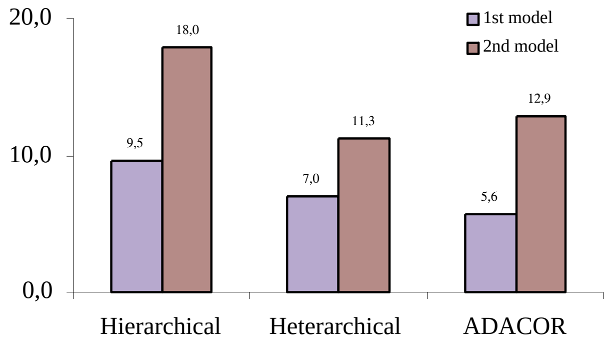


Figure 6.24: Loss of Productivity of the Evaluated Control Approaches

Analysing the loss of productivity values, it is possible to verify that the ADACOR control approach presents similar values to those exhibited by heterarchical-like control approach. As expected, the hierarchical-like control approach presents the higher loss of productivity.

The agility can be inducted by analysing the loss of productivity. The experimental results show that the ADACOR control approach presents the same levels of agility to those presented by the heterarchical-like control approach. In more drastic scenarios, illustrated with the second disturbance model, the levels of agility presented by the several control approaches are reduced.

Analysing simultaneously the agility and throughput, the results obtained in these experimental tests confirm that the ADACOR control approach combines the hierarchical and heterarchical best features, presenting similar values of agility to the heterarchical approach, but better production optimisation.

6.4.3 Evaluation of Quantitative Parameters over Manufacturing Load

The second set of experimental tests intends to analyse the correlation between the manufacturing load of the system and the manufacturing control performance parameters.

This experimental test was done with four different levels of plant load: 6 production orders

(17 operations), 12 production orders (34 operations), 18 production orders (51 operations) and 24 production orders (68 operations), each experimental test being executed 6 times. The first disturbance model was used.

Analysis of the Manufacturing Lead Time

The statistical results of the manufacturing lead time for the four values of manufacturing load, and using the three different control approaches, are summarised in the Figure 6.25.

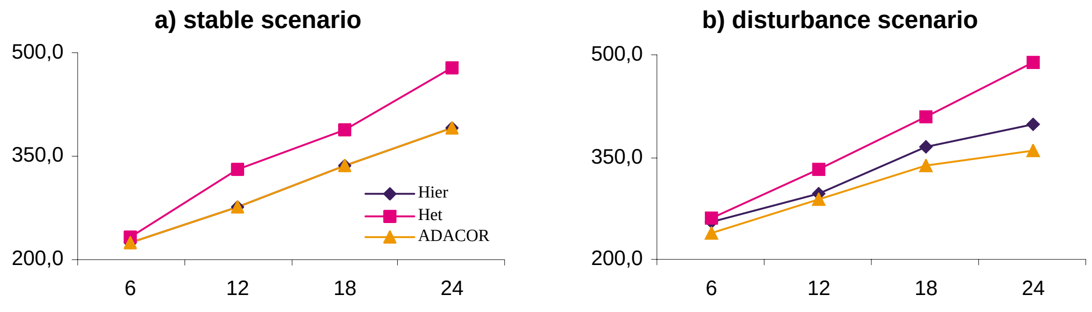


Figure 6.25: Evolution of Manufacturing Lead Time over the Manufacturing Load

Analysing the experimental results it is possible to verify that the manufacturing lead time increases with the increase of the manufacturing system load, and that the scenario with no disturbances and the scenario with disturbances present similar patterns.

It is also possible to verify that the dependency of the lead time with the manufacturing load is probably linear for stable and disturbance scenarios. Using the linear regression we found the equations that regulates the relation of the manufacturing lead time with the manufacturing load. The linear equations for the several approaches are illustrated in Table 6.3.

Table 6.3: Linear Regression for the Manufacturing Lead Time

	Stable Scenario	**Disturbance Scenario**
Hierarchical	165,9 + 9,4 t (r = 0,99997)	201,9 + 8,4 t (r = 0,99919)
Heterarchical	160,2 + 13,2 t (r = 0,99476)	181,6 + 12,7 t (r = 0,99976)
ADACOR	165,9 + 9,4 t (r = 0,99997)	202,9 + 6,9 t (r = 0,98816)

The analysis of linear regression values obtained for the several control approaches allows to elaborate some conclusions.

In the stable scenario, the hierarchical-like and ADACOR control approaches present higher initial values and smaller slope than the heterarchical-like control approach, implying that the higher the manufacturing load the better will be the performance of the hierarchical-like and ADACOR control approaches, in terms of manufacturing lead time.

The linear regression values associated to the disturbance scenario have higher initial values and slightly smaller slopes than those presented in stable scenarios. This observation allows to conclude that any control approach will present worse lead time in disturbance scenarios than in stable scenarios. The ADACOR control approach presents also smaller slope than the hierarchical-like and heterarchical-like control approaches implying that the higher the manufacturing load the better will be the performance of ADACOR control approach.

Analysis of the Throughput

The experimental results of the relation of the throughput with the manufacturing load, for the three control architectures, are graphically summarised in the Figure 6.26.

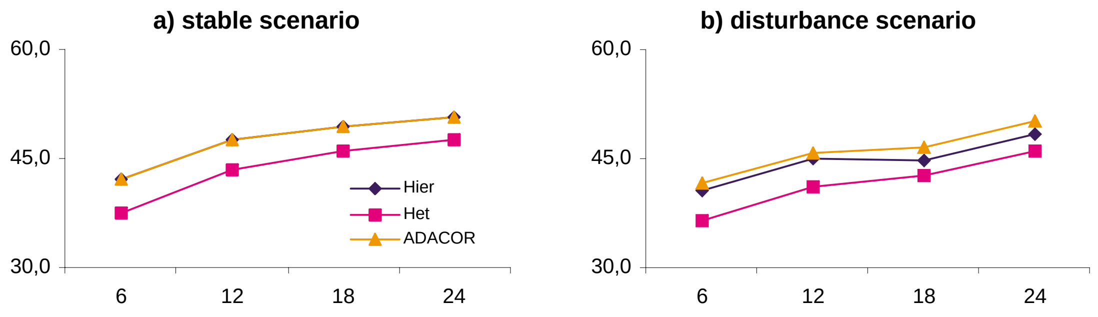


Figure 6.26: Evolution of Throughput over the Manufacturing Load

The analysis of the experimental results allows to verify that the throughput increases slightly with the manufacturing load, presenting a similar pattern to the manufacturing lead time.

Using linear regression, it is possible to find the linear equations for the several control approaches, as illustrated in the Table 6.4.

Table 6.4: Linear Regression for the Throughput

	Stable Scenario	**Disturbance Scenario**
Hierarchical	40,5 + 0,47 t (r = 0,9441)	38,8 + 0,39 t (r = 0,9378)
Heterarchical	35,3 + 0,56 t (r = 0,9562)	34,0 + 0,51 t (r = 0,9849)
ADACOR	40,5 + 0,47 t (r = 0,9441)	39,5 + 0,44 t (r = 0,9733)

The analysis of the linear regression results for the stable scenarios allows to verify that the hierarchical-like and ADACOR control approaches present higher initial values of throughput and smaller slope values than the heterarchical-like control approach. This observation allows to conclude that in stable scenarios, the hierarchical-like and ADACOR control approaches will present better performance, in terms of throughput, than the heterarchical-like control approach.

Analysing the linear regression values obtained for the disturbance scenario, it is possible to verify that the ADACOR control approach presents the highest initial value and the ADACOR and heterarchical-like control approaches present quite similar slope values, higher than the slope presented by the hierarchical-like control approach. This observation allows to conclude that the ADACOR control approach will present better performance in disturbance scenarios with higher manufacturing loads.

The heterarchical-like control approach will present similar values of throughput to ADACOR control approach, for both evaluated scenarios, for higher manufacturing loads.

6.5 Summary

This chapter intended to validate the proposed holonic manufacturing control architecture through the implementation of its concepts into a case study and the analysis of the manufacturing control response to several different manufacturing scenarios. The experience gained during the prototype implementation, debugging and testing, allows to extract some conclusions.

The use of agent technology to implement the holonic manufacturing control prototype brings some important benefits: the software necessary to develop the application is simpler to write, to debug and to maintain, due to the smaller size of each distributed component. The use of Java language contributes for the platform independency, which is mandatory in manufacturing environment, due to its heterogeneous environment, with the need of interaction between applications running in different platforms.

The use of JADE agent development tool brings several advantages in the development of holonic and multi-agent systems, such as the reduction of the development time and complexity. For this fact contributes the good documentation, efficient technical support and the set of functionalities provided by the platform that simplifies the development of multi-agent systems, such as the communication infra-structure, yellow and white pages and debugging tools.

The experimental tests allow to verify that the ADACOR control system works as specified, either in normal operation or in presence of disturbances, which was one of the major objectives of the experimental validation. During the experimental tests it was also tested the prototype re-configurability, i.e. the system response to the introduction and remotion of manufacturing components. The use of the plug and produce concept, allows that each ADACOR holon works autonomously, not requiring the need for additional re-design, re-program and re-start of other components. This feature helps the re-configurability of the manufacturing control architecture, essential to support different production systems or different book of products.

The experimental results reported in this chapter, which are preliminary due to the immature stage of the prototype implementation and to the limited number of experiences and scenarios, shows that the ADACOR control approach presents the better performance represented by high values of agility (reflected by the analysis of the loss of productivity parameter), combined with high values of production optimisation (illustrated by the throughput indicator).

These results show that the ADACOR concepts are promising, requiring a further development of the prototype, for example using more powerful learning mechanisms, and the execution of more experiences (for example testing the prototype under more different manufacturing scenarios).

Chapter 7

Conclusions and Future Work

"In theory, there is no difference between theory and practice. But, in practice, there is."
Jan L.A. van de Snepscheut

The manufacturing companies at the beginning of 21th century have to face a dynamic environment where economical, technological and customer trends changes rapidly, requiring the increase of flexibility and agility to react to unexpected disturbances, maintaining the productivity and quality. The traditional manufacturing control systems are adapted on a case-by-case basis, requiring an expensive and huge time-consuming effort to develop, maintain or re-configure. The missing re-configurability is derived from the lack of agility to support emergency (change and unexpected disturbances). The challenge is thus to develop innovative, agile and adaptive architectures for distributed manufacturing control systems, using emergent paradigms and technologies that can provide the answer to those requirements.

7.1 Conclusions

The proposed manufacturing control architecture addresses the improvement of performance in industrial scenarios characterised by the frequent occurrence of unexpected disturbances at shop floor level.

ADACOR architecture is based on a set of autonomous and cooperative holons, each one having autonomy, cooperation, decision-making and self-organisation capabilities. ADACOR defines four types of manufacturing holons: product, task, operational and supervisor holons. The supervisor holon is an innovative aspect of the ADACOR approach, which introduces coordination and global optimisation in decentralised control systems, coordinating several operational and supervisor holons. The supervisor holon is also responsible for the group formation, based in clusters of holons, combining synergies, aggregating skills and offering the combined services to external entities in the manufacturing system. These groups can be formed to build up a shop floor, a manufacturing cell, or a machine equipped with a set of tools, assuming the supervisor holon the control role in each group.

The supervisor holon and the self-organisation capability associated to each ADACOR holon allow the adaptive production control, to combine the global production optimisation with the agile reaction to disturbances. In normal operation, the supervisor holon supervises and regulates the activity of the holons under its coordination domain, while when a disturbance occurs, these holons may have to find their way without the help of the supervisor holon.

The improvements resulting from the introduction of ADACOR can be observed at two different levels: design and operation.

At design level, taking advantage of using descentralised systems, the design, maintenance, expansion and re-use of manufacturing control applications is simpler than in traditional approaches.

At the operation level, the merits of the proposed adaptive holonic production control approach is to provide a mechanism to adapt the manufacturing control system structure to unexpected disturbances and to return to normal operation when the disturbance is repaired. With the new solution, the optimisation of the centralised control approach in scenarios absent of disturbances is combined with the agility of the heterarchical control systems in unstable scenarios characterised by the frequent occurrence of unexpected disturbances.

The ADACOR architecture can also be used for applications where the objective is just the production optimisation or the agility, since the ADACOR control system can be organised in hierarchical, heterarchical or other control structures with small adjustments, such as the introduction or remotion of supervisor holons and enabling/disabling the autonomy factor associated to the operational holons.

As result of this research work, some contributions for the state of the art have been achieved.

The first contribution is concerned to the achievement of **global optimisation in decentralised structures**, by the introduction of the supervisor holon, which allows to have coordination features at different levels of control hierarchy, acting as cell controller or shop floor controller, and the agility of reaction to disturbances presented in the heterarchical approaches.

The second main contribution is related to the **adaptive holonic control** that uses the supervisor role and the self-organisation concept over decentralised systems. Each operational holon has associated an autonomy factor that regulates the level of autonomy of the holon, and its behaviour in presence of hierarchies. This concept, associated to the propagation mechanisms based in pheromone-like techniques and the supervisor role, allows a dynamic self-reconfiguration of the control structure.

The manufacturing control applications are normally complex systems, difficult to understand and to design. This work contributes also for the **formal specification and validation** of these systems by modelling the dynamic behaviour of the ADACOR holon classes and their interactions, using a Petri Net formalism tailored for holonic control system modelling.

The **set-up and re-configuration** of the manufacturing control application are important aspects to consider in the analysis of the control approach. The re-configuration during the application life cycle, for example the addition of new components, can be a frequent scenario in manufacturing systems, and should be possible without the need of re-designing, re-programming and re-initialising other components. Since the ADACOR architecture is designed as a set of autonomous and cooperative holons, each one designed to adapt its behaviour in order to achieve its own goals, the re-configuration of the structure in ADACOR control approach is easier than in the traditional approaches, because it is possible to build distinct and independent holons that can be placed transparently in a distributed environment, making easier the expansibility of the control application.

Another contribution is the **integration of physical automation devices** using the virtual resource concept, that allows a completely transparent and independent connection, making easier the integration of physical devices. Using the proposed resource integration mechanism, the development of the manufacturing control application is independent from the particularities of each real manufacturing device.

Also, the addition of intelligence to a holon, for example to make decisions, manage disturbances or learn, is a transparent process for the holon and can be viewed as a **plug-in** module, which makes easier the development of manufacturing control applications. The introduction of learning mechanisms in the holon capabilities, allows the dynamic improvement of its behaviour and performance, taking into account previous experiences.

7.2 Future Work

At the end of this research work, we identified several issues that could be further researched as well as new and promising research windows related to this research domain. In this section, some important research topics and suggestions for future work will be described.

The **product holon** does not plays a crucial role in the ADACOR architecture, being only necessary to achieve the integration of process planning, scheduling and execution processes. The product holon has not been completely specified and designed requiring the implementation of other features, such as its graphical user interface, short-term process planning and learning mechanisms, in order to offer more powerful capabilities.

The design of the **virtual resource mechanism** should be further studied, mainly focusing in the development of generic components for the virtual resources and the definition of guidelines to develop the specific components, based in the particularities of the machines. These guidelines comprise the definition of virtual machine models that identify the variables and services available in the machine, using for example a MMS sub-set of services. Additional aspects that should be tested are the use of distributed object platforms and the associated real time performance. The further validation of the virtual resource concept is required, through the integration of more automation devices within the manufacturing control system, using different connection platforms, such as fieldbus, OPC and Industrial Ethernet.

ADACOR production control is based on the behaviour of individual ADACOR holons, implemented as **building blocks**, similar to Legos^{TM}components, that can be improved in the future, by the addition of improved learning mechanisms and scheduling algorithms.

In ADACOR architecture a basic **manufacturing ontology** was developed to support the inter-operability between ADACOR holons. However, to address the inter-operability between different holonic applications platforms, i.e., for two or more different holonic applications communicate, a common manufacturing ontology is required. An alternative solution is to have neutral and generic parsers that can translate the knowledge from one ontology to another.

In industrial manufacturing applications, the ordinary **Petri net models** become highly complex and difficult to handle, and the developed Petri net models for the ADACOR holon classes present some limitations, namely the presence of many instances of the same component. As future work, the ADACOR holon classes and their interactions should be modelled using High-Level Petri net, applying the same methodology as illustrated in the modelling of the ADACOR product holon. Also, a future work could be the development of coordination models, integrating and synchronising the individual Petri net models.

In a holonic enterprise all levels should be linked and integrated being possible to exchange data along the **vertical structure of the enterprise**, from the shop floor to the factory and inter-factory levels. Future research work could determine if ADACOR holonic concepts can be extended to these levels.

The evaluation and the comparison of **manufacturing control systems performance** requires frameworks that define test scenarios and normalised performance indicators, decoupling the control system performance from the performance of the other components of the manufacturing system, and from the particular implementation of the architecture concepts.

Since the agent technology is a suitable approach for the implementation of holonic manufacturing control applications and FIPA specifications are commonly used, work towards the **inclusion in FIPA of specific requirements** to the manufacturing control systems, such as no pre-emption of operations, event notification, unsubscription of a service, appropriate protocols for manufacturing domain and mechanisms for the integration of physical manufacturing devices, is required.

In conclusion, the development and implementation of agile, flexible and intelligent holonic manufacturing systems, as described in this thesis, present a great potential due to the achievements of more agility and robustness and higher product flexibility.

ADACOR architecture is adequate for control systems satisfying the requirement of dealing with important disturbances during a long period of time, as it is the case of industries competing in modern dynamic environments, where decentralised decision-making may have clear advantages over centralised realisations.

Bibliography

[Ackoff, 1989] Ackoff, R. (1989). From Data to Wisdom. *Journal of Applies Systems Analysis*, 16:3–9.

[Andersson, 1997] Andersson, N. (1997). *On Modelling and Implementing Shop Floor Control Systems*. PhD thesis, Chalmers University of Technology, Sweden, Department of Production Engineering.

[APICS, 1995] APICS (1995). *APICS Dictionary*. American Production and Inventory Control Society, Inc., 8th edition.

[Arai et al., 2001] Arai, T., Aiyama, Y., Sugi, M., and Ota, J. (2001). Holonic Assembly System with Plug and Produce. *Computers in Industry*, 46(3):289–299.

[Arentsen, 1995] Arentsen, A. (1995). *A Generic Architecture for Factory Activity Control*. PhD thesis, University of Twente, The Netherlands.

[Ariza et al., 2001] Ariza, T., Fernández, F., and Rubio, F. (2001). Implementing a Virtual Manufacturing Device for MMS/CORBA. In *Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation*, volume 2, pages 711–715, Nice, France.

[Baker, 1998] Baker, A. (1998). A Survey of Factory Control Algorithms which Can be Implemented in a Multi-Agent Heterarchy: Dispatching, Scheduling and Pull. *Journal of Manufacturing Systems*, 17(4):297–320.

[Balasubramanian et al., 2001] Balasubramanian, S., Brennan, R., and Norrie, D. (2001). An Architecture for Metamorphic Control of Holonic Manufacturing Systems. *Computers in Industry*, 46(1):13–31.

[Banaszak and Krogh, 1990] Banaszak, Z. and Krogh, B. (1990). Deadlock Avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows. *IEEE Transactions on Robotics and Automation*, 6(6):724–734.

[Barata and Camarinha-Matos, 2003] Barata, J. and Camarinha-Matos, L. (2003). Coalitions of Manufacturing Components for Shop Floor Agility - The CoBaSA Architecture. *International Journal of Networking and Virtual Organisations*, 2(1):50–77.

[Barata et al., 2001] Barata, J., Camarinha-Matos, L., Boissier, R., Leitão, P., Restivo, F., and Raddadi, M. (2001). Integrated and Distributed Manufacturing, a Multi-agent Perspective. In *Proceedings of the 3rd Workshop on European Scientific and Industrial Collaboration*, pages 145–156, Enschede, Netherlands.

[Bauer et al., 1991] Bauer, A., Bowden, R., Browne, J., Duggan, J., and Lyons, G. (1991). *Shop Floor Control Systems - From Design to Implementation.* Chapman & Hall.

[Bellifemine et al., 2002] Bellifemine, F., Caire, G., Trucco, T., and Rimassa, G. (2002). JADE Programmer's Guide.

[Bellifemine et al., 1999] Bellifemine, F., Poggi, A., and Rimassa, G. (1999). JADE, A FIPA-compliant Agent Framework. In *Proceedings of Fourth International Conference on the Practical Application of Intelligent Agents and Multi-Agents*, pages 97–108, London.

[Bititci et al., 2002] Bititci, U., Nuderupati, S., and Turner, T. (2002). Web Enabled Performance Measurement Systems; Management Implications. *International Journal of Operations and Production Management*, 22(1):1273–1287.

[Black, 1991] Black, J. (1991). *The Design of the Factory with a Future.* McGraw Hill.

[Boissier et al., 1998] Boissier, R., Epivent, M., Gressier-Soudan, E., and Horn, F. (1998). Providing Real-Time Object Oriented Industrial Messaging Services. In *Proceedings of the IFIP European Conference on Object Oriented Programming*, Brussels.

[Boissier et al., 2001] Boissier, R., Gressier-Soudan, E., Laurent, A., and Seinturier, L. (2001). Enhancing Numerical Controllers, using MMS Concepts and a CORBA-based Software Bus. *International Journal of Computer Integrated Manufacturing*, 14(6):560–569.

[Bongaerts, 1998] Bongaerts, L. (1998). *Integration of Scheduling and Control in Holonic Manufacturing Systems.* PhD thesis, Katholieke Universiteit Leuven, Belgium.

[Bongaerts et al., 1998] Bongaerts, L., Monostori, L., McFarlane, D., and Kádár, B. (1998). Hierarchy in Distributed Shop Floor Control. In *Proceedings of the First Open Workshop of the Esprit Working Group on Intelligent Manufacturing Systems*, pages 97–113, Lausanne.

[Brennan et al., 1997] Brennan, R., Balasubramanian, S., and Norrie, D. (1997). A Dynamic Control Architecture for Metamorphic Control of Advanced Manufacturing Systems. In Gopaladrishnan, B., Murugesan, S., Struger, O., and Zeichen, G., editors, *Proceedings of the International Symposium on Intelligent Systems and Advanced Manufacturing*, pages 213–223.

[Brennan et al., 2002] Brennan, R., Fletcher, M., and Norrie, D. (2002). An Agent-based Approach to Reconfiguration of Real-Time Distributed Control Systems. *IEEE Transactions on Robotics and Automation*, 18(4):444–451.

[Britannica, 2003] Britannica (2003). Encyclopaedia Britannica. http://search.britannica.com/.

[Browne et al., 1984] Browne, J., Dubois, D., Rathmill, K., Sethi, S., and Stecke, K. (1984). Classification of Flexible Manufacturing Systems. *The FMS Magazine*, pages 114–117.

[Brussel et al., 1999] Brussel, H. V., Bongaerts, L., Wyns, J., Valckenaers, P., and Ginderachter, T. (1999). A Conceptual Framework for Holonic Manufacturing Systems: Identification of Manufacturing Holons. *Journal of Manufacturing Systems*, 18(1):35–52.

[Brussel et al., 2000] Brussel, H. V., Valckenaers, P., Wyns, J., Peeters, P., and Bongaerts, L. (2000). Holonic Manufacturing Systems, Architectural and Manufacturing Control Issues. In *Proceedings of 2nd CIRP International Seminar on Intelligent Computation in Manufacturing Engineering*, pages 19–29, Capri, Italy.

[Brussel et al., 1998] Brussel, H. V., Wyns, J., Valckenaers, P., and Bongaerts, L. (1998). Reference Architecture for Holonic Manufacturing Systems: PROSA. *Computers In Industry*, 37(3):255–274.

[Bussman and McFarlane, 1999] Bussman, S. and McFarlane, D. (1999). Rationales for Holonic Manufacturing Control. In *Proceedings of the Second International Workshop on Intelligent Manufacturing Systems*, pages 177–184, Leuven, Belgium.

[Bussmann, 1998] Bussmann, S. (1998). An Agent-Oriented Architecture for Holonic Manufacturing Control. In *Proceedings of the First Open Workshop of the Esprit Working Group on Intelligent Manufacturing Systems*, pages 1–12, Lausanne, Switzerland.

[Bussmann et al., 2000] Bussmann, S., Jennings, N., and Wooldridge, M. (2000). On the Identification of Agents in the Design of Production Control Systems. In Ciancarini, P. and Wooldridge, M., editors, *Agent-Oriented Software Engineering*, volume 1957 of *Lecture Notes in Computers Science*, pages 141–162. Springer-Verlag.

[Bussmann and Schild, 2001] Bussmann, S. and Schild, K. (2001). An Agent-based Approach to the Control of Flexible Production Systems. In *Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation*, volume 2, pages 481–488, Antibes, France.

[Camarinha-Matos and Afsarmanesh, 1999] Camarinha-Matos, L. M. and Afsarmanesh, H. (1999). Infrastructures for Virtual Enterprises: a Summary of Achievements. In Camarinha-Matos, L. and Afsarmanesh, H., editors, *Proceedings of the PRO-VE'99 - IFIP International Conference On Infrastructures for Virtual Enterprises*, pages 483–490. Kluwer Academic Publishers.

[Cavalieri et al., 1999] Cavalieri, S., Bongaerts, L., Taisch, M., Macchi, M., and Wyns, J. (1999). A Benchmark Framework for Manufacturing Control. In *Proceedings of the Second International Workshop on Intelligent Manufacturing Systems*, Leuven, Belgium.

[Chirn and McFarlane, 2000] Chirn, J.-L. and McFarlane, D. (2000). A Holonic Component-Based Approach to Reconfigurable Manufacturing Control Architecture. In *Proceedings of the International Workshop on Industrial Applications of Holonic and Multi-Agent Systems*, pages 219–223, London, UK.

[Cho and Wysk, 1995] Cho, H. and Wysk, R. (1995). Intelligent Workstation Controller for Computer Integrated Manufacturing: Problems and Models. *Journal of Manufacturing Systems*, 14(4):252–263.

[Christensen, 1994] Christensen, J. (1994). Holonic Manufacturing Systems - Initial Architecture and Standard Directions. In *Proceedings of the First European Conference on Holonic Manufacturing Systems*, Hannover, Germany.

[Christensen, 2002] Christensen, J. (2002). IEC 61499 Architecture: Engineering Methodologies and Software Tools. In Marík, V., Camarinha-Matos, L., and Afsarmanesh, H., editors, *Knowledge and Technology Integration in Production and Services: Balancing Knowledge and Technology in Product and Service Life Cycle*, pages 221–228. Kluwer Academic Press.

[Chryssolouris, 1987] Chryssolouris, G. (1987). An Approach to Intelligent Manufacturing Systems. In *CIM Review Artificial Intelligence in Manufacturing*. Springer-Verlag.

[Colombo, 1998] Colombo, A. (1998). Development and Implementation of Hierarchical Control Structures of Flexible Production Systems Using High-Level Petri Nets. In Feldmann, K., editor, *Manufacturing Automation Series*. MeisenbachVerlag Bamberg.

[Colombo and Carelli, 1997] Colombo, A. and Carelli, R. (1997). Petri Nets for Designing Manufacturing Systems. In Tzafestas, S., editor, *Computer-Assisted Management and Control of Manufacturing Systems*, chapter 11. Springer-Verlag.

[Colombo et al., 1997] Colombo, A., Carelli, R., and Kuchen, B. (1997). A Temporized Petri Net Approach for Designing, Modelling and Analysis of Flexible Production Systems. *The International Journal of Advanced Manufacturing Technology, Springer Verlag*, 13(3):214–226.

[Colombo et al., 2004] Colombo, A., Schoop, R., and Neubert, R. (2004). An Agent-based Intelligent Control Platform for Industrial Holonic Manufacturing Systems. *IEEE Transaction on Industrial Electronics (IEEE-IES)*, Accepted to be published.

[Colombo et al., 2001] Colombo, A. W., Neubert, R., and Schoop, R. (2001). A Solution to Holonic Control Systems. In *Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 489–498, Sophia/Nice, France.

[Colombo et al., 2002] Colombo, A. W., Neubert, R., and Sussmann, B. (2002). A Coloured Petri Net based Approach Towards a Formal Specification of Agent-Controlled Production Systems. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, Tunisia.

[Deen, 2003] Deen, S., editor (2003). *Agent-Based Manufacturing: Advances in the Holonic Approach*. Springer Verlag Berlin Heidelberg.

[Denkena et al., 2002] Denkena, B., Tönshoff, H. K., Zwick, M., and Woelk, P. (2002). Process Planning and Scheduling with Multi-agent Systems. In Marík, V., Camarinha-Matos, L., and Afsarmanesh, H., editors, *Knowledge and Technology Integration in Production and Services: Balancing Knowledge and Technology in Product and Service Life Cycle*, pages 339–348.

[Desrochers and Al-Jaar, 1995] Desrochers, A. and Al-Jaar, R. (1995). *Applications of Petri Nets in Manufacturing Systems-Modeling, Control and Performance Analysis*. IEEE Press.

[Diltis et al., 1991] Diltis, D., Boyd, N., and Whorms, H. (1991). The Evolution of Control Architectures for Automated Manufacturing Systems. *Journal of Manufacturing Systems*, 10(1):63–79.

[Duffie and Piper, 1986] Duffie, N. and Piper, R. (1986). Non-Hierarchical Control of Manufacturing Systems. *Journal of Manufacturing Systems*, 5(2):137–139.

[Dumant et al., 1998] Dumant, B., Horn, F., Tran, F., and Stéfani, J.-B. (1998). Jonathan: An Open Distributed Processing Environment in Java. In Davies, N., Raymond, K., and Seitz, J., editors, *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Midlleware'98)*, pages 175–190.

[Fanti et al., 1997] Fanti, M., Maione, B., Mascolo, S., and Turchiano, B. (1997). Event-based Feedback Control for Deadlock Avoidance in Flexible Production Systems. *IEEE Transactions on Robotics and Automation*, 13(3):347–363.

[Farquhar et al., 1996] Farquhar, A., Fikes, R., and Rice, J. (1996). The Ontolingua Server: A Tool for Collaborative Ontology Construction. Technical report, Stanford KSL 96-26.

[Feldmann et al., 1996] Feldmann, K., Schnur, C., and Colombo, A. (1996). Modularised, Distributed Real-Time Control of Flexible Production Cells, Using Petri Nets. *Control Engineering Practice, International Journal of IFAC*, pages 1067–1078.

[Ferber, 1999] Ferber, J. (1999). *Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence*. Addison-Wesley.

[Finin et al., 1993] Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McKay, D., McGuire, J., Pelavin, R., Shapiro, S., and Bech, C. (1993). *Specification of the KQML Agent-Communication Language*. DARPA Knowledge Sharing Initiative.

[FIPA, 2003] FIPA (2003). Foundation for Intelligent Physical Agents. http://www.fipa.org/.

[Fisher, 1999] Fisher, K. (1999). Agent-Based Design of Holonic Manufacturing Systems. *Journal of Robotics and Autonomous Systems, Elsevier Science B.V.*, (27):3–13.

[Fletcher et al., 2002] Fletcher, M., Marík, V., and Vrba, P. (2002). Design Issues in Holonic Inventory Management and Material Handling Systems. In Marík, V., Camarinha-Matos, L., and Afsarmanesh, H., editors, *Knowledge and Technology Integration in Production and Services: Balancing Knowledge and Technology in Product and Service Life Cycle*, pages 271–280. Kluwer Academic Publishers.

[Fletcher et al., 2003] Fletcher, M., McFarlane, D., Lucas, A., Brusey, J., and Jarvis, D. (2003). The Cambridge Packing Cell - A Holonic Enterprise Demonstrator. In Marík, V., Müller, J., and Pechoucek, M., editors, *Multi-Agent Systems and Applications III, 3rd International/Central and Eastern European Conference on Multi-Agent Systems*, volume 2691 of *Lecture Notes in Artificial Intelligence*, pages 533–543. Springer-Verlag.

[Frey et al., 2002] Frey, G., Minas, M., and John, K. (2002). Steuerungsentwurf mit Perinetzen. In *SPS Magazin*, number 4/5, pages 44–47. Verlag Marburg.

[Friedman-Hill, 1999] Friedman-Hill, E. (1999). JESS, The Java Expert System Shell. Sandia National Laboratories, Livermore, CA.

[Gangemi et al., 2002] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L. (2002). Sweetening Ontologies with DOLCE. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*, Siguenza, Spain.

[Gerwin, 1993] Gerwin, D. (1993). Manufacturing Flexibility: A Strategic Perspective. *Management Science*, 39:395410.

[Ghalayini and Noble, 1996] Ghalayini, A. M. and Noble, J. (1996). The Changing Basis of Performance Measurement. *International Journal of Operations and Production Management*, 16(8):63–80.

[Giarratano and Riley, 1998] Giarratano, J. and Riley, G. (1998). *Expert Systems: Principles and Programming*. PWS Publishing Company, third edition.

[Glover and Laguna, 1997] Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers.

[Goldman and Rosenschein, 1996] Goldman, C. and Rosenschein, J. (1996). Mutually Supervised Learning in Multiagent Systems. In Weiss, G. and Sen, S., editors, *Adaptation in Multi-Agent Systems*, Lecture Notes in Artificial Intelligence, pages 85–96. Springer-Verlag.

[Goldman et al., 1995] Goldman, S., Nagel, R., and Preiss, K. (1995). *Agile Competitors and Virtual Organisations*. Van Nostrand Reinhold, New York.

[Gou et al., 1998] Gou, L., Luh, P., and Kyoya, Y. (1998). Holonic Manufacturing Scheduling: Architecture, Cooperation Mechanism, and Implementation. *Computers in Industry*, 37(3):213–231.

[Groover, 1987] Groover, M. (1987). *Automation, Production Systems and CIM*. Prentice-Hall.

[Gruber, 1995] Gruber, T. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human and Computer Studies*, 43(5/6):907–928.

[Guarino, 1998] Guarino, N. (1998). Formal Ontology and Information Systems. In *Proceedings of the First International Conference on Formal Ontologies in Information Systems*, pages 3–15, Trento, Italy.

[Gullander, 1999] Gullander, P. (1999). *On Reference Architectures for Development of Flexible Cell Control Systems*. PhD thesis, Chalmers University of Technology, Sweden.

[Haberman, 1969] Haberman, A. (1969). Prevention of System Deadlocks. *Communications of the ACM*, 12(7):373–377.

[Haddad and Couvreur, 1988] Haddad, S. and Couvreur, J. M. (1988). Towards a General and Powerful Computation of Flows for Parametrized Coloured Nets. In *Proceedings of the 9th European Workshop on Applications and Theory of Petri Nets*, Venice, Italy.

[Harrison, 1992] Harrison, A. (1992). *Just in Time Manufacturing in Perspective*. Prentice-Hall.

[Hasegawa et al., 1994] Hasegawa, T., Gou, L., Tamura, S., Luh, P. B., and Oblak, J. M. (1994). Holonic Planning and Scheduling Architecture for Manufacturing. In *International Working Conference on Cooperating Knowledge Based Systems*, pages 125–139, Keele, U.K.

[Heikkilä et al., 1997] Heikkilä, T., Jarviluoma, M., and Juntunen, T. (1997). Holonic Control for Manufacturing Systems: Design of a Manufacturing Robot Cell. *Integrated Computer Aided Engineering*, 4:202–218.

[HMS, 2004] HMS (2004). Holonic Manufacturing Systems Consortium web site. http://hms.ifw.uni-hannover.de.

[Holloway et al., 1997] Holloway, L., Krogh, B., and Giua, A. (1997). Survey of Petri Net Methods for Controlled Discrete-Event Systems. *Discrete-Event Systems: Theory and Applications, Kluwer Academics*, 7(2):151–190.

[Holobloc, 2003] Holobloc (2003). Holobloc web. http://www.holobloc.com.

[Höpf, 2002] Höpf, M. (2002). Agent-Controls for Manufacturing - Technical Marketing Perspectives. In *Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society*, pages 2974–2977, Sevilla, Spain.

[Horrocks et al., 2001] Horrocks, I., Harmelen, F., and Patel-Schneider, P. (2001). DAML+OIL. http://www.daml.org/2001/03/daml+oil-index.html.

[Ibarrondo and Mercader, 2001] Ibarrondo, J. and Mercader, J. (2001). Measuring Operational Flexibility. In *Proceedings of the Fourth Stimulating Manufacturing Excellence in Small and Medium Enterprises International Conference*, pages 292–302, Alborg, Denmark.

[ISO/IEC9506-1, 1992] ISO/IEC9506-1 (1992). *Industrial Automation Systems - Manufacturing Message Specification, Part 1 - Service Definition.*

[JASS, 2001] JASS (2001). Special Issue on Industrial Applications of Multi-Agent and Holonic Systems. *Journal of Applied Systems Studies*, 2(1).

[Jensen, 1992] Jensen, K. (1992). *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, volume 1 of *Monographs on Theorical Computer Science*. Springer-Verlag.

[Kaplan and Norton, 1996] Kaplan, R. S. and Norton, D. P. (1996). *The Balanced Scorecard: Translating Strategy into Action.* Harvard Business School Press.

[Keegan et al., 1989] Keegan, D., Eiler, R., and Jones, C. (1989). Are Your Performance Measures Obsolete? *Management Accounting (US)*, pages 45–50.

[Kidd, 1994] Kidd, P. T. (1994). *Agile Manufacturing: Forging New Frontiers.* Addison-Wesley.

[KIF, 2003] KIF (2003). Knowledge Interchange Format. http://logic.stanford.edu/kif/kif.html.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by Simulated Annealing. *Science*, 220:671–680.

[Koestler, 1969] Koestler, A. (1969). *The Ghost in the Machine.* Arkana Books, London.

[Kruth et al., 1996] Kruth, J., Detand, J., Zeir, G. V., Kempenaers, J., and Pinte, J. (1996). Methods to Improve the Response Time of a CAD System that Generates Non Linear Process Plans. *Advances in Engineering Software*, 25:9–17.

[Kubat et al., 1996] Kubat, M., Bratko, I., and Michalski, R. (1996). A Review of Machine Learning Methods. In Michalski, R., Bratko, I., and Kubat, M., editors, *Machine Learning and Data Mining: Methods and Applications*, pages 1–72. John Wiley & Sons.

[Labrou et al., 1999] Labrou, Y., Finin, T., and Peng, Y. (1999). Agent Communication Languages: The Current Landscape. *IEEE Intelligent Systems*, 14(2):45–52.

[Leitão, 1996] Leitão, P. (1996). Modelização, Concepção e Implementação de Controladores de Células de Fabrico Flexível. Master's thesis, Faculdade de Engenharia da Universidade do Porto. (in Portuguese).

[Leitão et al., 2003a] Leitão, P., Boissier, R., Casais, F., and Restivo, F. (2003a). Integration of Automation Resources in Holonic Manufacturing Applications. In Marík, V., McFarlane, D., and Valckenaers, P., editors, *Holonic and Multi-Agent Systems for Manufacturing*, volume 2744 of *Lecture Notes in Artificial Intelligence*, pages 35–46. Springer-Verlag.

[Leitão et al., 2003b] Leitão, P., Colombo, A., and Restivo, F. (2003b). A Formal Validation Approach for Holonic Control System Specifications. In *Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation*, volume 1, pages 203–210, Lisboa, Portugal.

[Leitão et al., 2003c] Leitão, P., Colombo, A., and Restivo, F. (2003c). An Approach for the Formal Specification of Holonic Control Systems. In Marík, V., McFarlane, D., and Valckenaers, P., editors, *Holonic and Multi-Agent Systems for Manufacturing*, volume 2744 of *Lecture Notes in Artificial Intelligence*, pages 59–70. Springer-Verlag.

[Leitão et al., 2003d] Leitão, P., Colombo, A., Restivo, F., and Schoop, R. (2003d). Formal Specification of Holonic Control System ADACOR, using High-Level Petri Nets. In *Proceedings of the First IEEE International Conference on Industrial Information (INDIN2003)*, Alberta, Canada.

[Leitão and Restivo, 1999] Leitão, P. and Restivo, F. (1999). A Layered Approach to Distributed Manufacturing. In *Proceedings of Advanced Summer Institute International Conference in Life Cycle Approaches to Production Systems: Management, Control and Supervision*, Leuven, Belgium.

[Leitão and Restivo, 2002a] Leitão, P. and Restivo, F. (2002a). A Holonic Control Approach for Distributed Manufacturing. In Marík, V., Camarinha-Matos, L., and Afsarmanesh, H., editors, *Knowledge and Technology Integration in Production and Services: Balancing Knowledge and Technology in Product and Service Life Cycle*, pages 263–270. Kluwer Academic Publishers.

[Leitão and Restivo, 2002b] Leitão, P. and Restivo, F. (2002b). Agent-based Holonic Production Control. In *Proceedings of the 3rd International Workshop on Industrial Applications of Holonic and Multi-Agent Systems*, pages 589–593, Aix en Provence, France. IEEE Computer Society.

[Leitão and Restivo, 2002c] Leitão, P. and Restivo, F. (2002c). Holonic Adaptive Production Control Systems. In *Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society*, pages 2968–2973, Sevilla, Spain.

[Leitão and Restivo, 2003a] Leitão, P. and Restivo, F. (2003a). Identification of ADACOR Holons for Manufacturing Control. In *Proceedings of the 7th IFAC Workshop on Intelligent Manufacturing Systems*, pages 109–114, Budapest, Hungary.

[Leitão and Restivo, 2003b] Leitão, P. and Restivo, F. (2003b). Towards Autonomy, Self-Organisation and Learning in Holonic Manufacturing. In Marík, V., Müller, J., and Pechoucek, M., editors, *Multi-Agent Systems and Applications III, 3rd International/Central and Eastern European Conference on Multi-Agent Systems*, volume 2691 of *Lecture Notes in Artificial Intelligence*, pages 544–553. Springer-Verlag.

[Leitão and Restivo, 2004] Leitão, P. and Restivo, F. (2004). The Use of Qualitative Indicators For Performance Measurement in Manufacturing Control Systems. In *Submited to the 11th IFAC Symposyum on Information Control Problems in Manufacturing*, Salvador Bahia, Brasil.

[Lin and Solberg, 1992] Lin, G. and Solberg, J. (1992). Integrated Shop Floor Control using Autonomous Agents. *IIE Transactions*, 24(3):57–71.

[Loom, 2004] Loom (2004). Loom Project Home Page. http://www.isi.edu/isd/LOOM/LOOM-HOME.html.

[Luck et al., 2003] Luck, M., McBurney, P., and Preist, C. (2003). *Agent Technology: Enabling Next Generation Computing - A Roadmap for Agent-Based Computing.* AgentLink.

[Luh et al., 2000] Luh, P., Zhao, X., and Thakur, L. (2000). Lagrangian Relaxation Neural Networks for Job Scheduling. *IEEE Transactions on Robotics and Automation*, 16(1):78–88.

[Marapoulos, 1995] Marapoulos, P. (1995). Review of Research in Tooling Technology, Process Modelling and Process Planning, Part II: Process Planning. *Journal of Computer Integrated Manufacturing Systems*, 8(1):13 –20.

[Marík et al., 2002] Marík, V., Fletcher, M., and Pechoucek, M. (2002). Holons & Agents: Recent Developments and Mutual Impacts. In *Multi-Agent Systems and Applications II*, Lecture Notes in Computer Science, pages 233–267. Springer-Verlag Heidelberg.

[Marík et al., 2003] Marík, V., Pechoucek, M., Vrba, P., and Hrdonka, V. (2003). FIPA Standards and Holonic Manufacturing. In Deen, S., editor, *Agent-based Manufacturing: Advances in the Holonic Approach*, pages 89–121. Springer Verlag.

[Markus et al., 1996] Markus, A., Vancza, T. K., and Monostori, L. (1996). A Market Approach to Holonic Manufacturing. *Annals of CIRP*, 45:433–436.

[Maturana et al., 1996] Maturana, F., Balasubramanian, S., and Norrie, D. (1996). A Multi-Agent Approach to Integrated Planning and Scheduling for Concurrent Engineering. In *Proceedings of the International Conference on Concurrent Engineering: Research and Applications*, pages 272–279, Toronto, Ontario.

[Maturana and Norrie, 1996] Maturana, F. and Norrie, D. (1996). Multi-Agent Mediator Architecture for Distributed Manufacturing. *Journal of Intelligent Manufacturing*, 7:257–270.

[Maturana et al., 2002] Maturana, F., Staron, R., Tichy, P., and Slechta, P. (2002). Using Dynamically Created Decision-Making Organisation (Holarchies) to Plan, Commit and Execute Control Tasks in a Chilled Water System. In *Proceedings of the 3rd International Workshop on Industrial Applications of Holonic and Multi-Agent Systems*, pages 613–619, Aix-en-Provence, France. IEEE Computer Society.

[McFarlane et al., 1995] McFarlane, D., Marett, B., Elsley, G., Jarvis, D., and Wilbers, P. (1995). Application of Holonic Methodologies to Problem Diagnosis in a Steel Rod Mill. In *IEEE International Conference on Systems, Man and Cybernetics*, Vancouver.

[Merriam-Webster, 2003] Merriam-Webster (2003). Merriam-Webster Online. http://www.m-w.com/mw/netdict.htm.

[Monostori et al., 1996] Monostori, L., Markus, A., Brussel, H. V., and Westkäuper, E. (1996). Machine Learning Approaches to Manufacturing. *CIRP Annals*, 45(2):675–712.

[Müller, 1996] Müller, J. (1996). *An Architecture for Dynamically Interacting Agents.* PhD thesis, Universität des Saarlandes, Saarbrücken.

[Murata, 1989] Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *IEEE*, 77(4):541–580.

[Neely, 1999] Neely, A. (1999). The Performance Measurement Revolution: Why Now and What Next? *International Journal of Operations and Production Management*, 19(2):205–228.

[Nussbaumer, 1991] Nussbaumer, H. (1991). *Téléinformatique IV, Collection Informátique.* Presses Polytechniques Romandes.

[Nwana, 1996] Nwana, H. (1996). Software Agents: An Overview. *Knowledge Engineering Review*, 11(3).

[Obitko, 2003] Obitko, M. (2003). Introduction to genetic algorithms with Java Applets. http://cs.felk.cvut.cz/∼xobitko/ga/.

[Odell et al., 2000] Odell, J., Parunak, H. V. D., and Bauer, B. (2000). Representing Agent Interaction Protocols in UML. In Ciancarini, P. and Wooldridge, M., editors, *Agent-Oriented Software Engineering*, volume 1957 of *Lecture Notes in Computers Science*. Springer-Verlag.

[Okino, 1993] Okino, N. (1993). Bionic Manufacturing System. In Peklenik, J., editor, *CIRP Flexible Manufacturing Systems Past-Present-Future*, pages 73–95.

[Onori, 1996] Onori, M. (1996). *The Robot Motion Module: A Task-oriented Robot Programming System for FAA Cells*. PhD thesis, Department of Manufacturing Systems, The Royal Institute of Technology, Sweden.

[Parunak, 1996] Parunak, H. V. D. (1996). Foundations of Distributed Artificial Intelligence. In O'Hare, G. and Jennings, N., editors, *Applications of Distributed Artificial Intelligence in Industry*, pages 139–164. John Wiley & Sons.

[Parunak, 1998] Parunak, H. V. D. (1998). What can Agents do in Industry, and Why? An Overview of Industrially-Oriented R&D at CEC. In Klusch, M. and Weiss, G., editors, *Cooperative Information Agents II - Learning, Mobility and Electronic Commerce for Information Discovery on the Internet*, volume 1435 of *Lecture Notes in Computers Science*, pages 1–18. Springer.

[Parunak et al., 1998] Parunak, H. V. D., Baker, A., and Clark, S. (1998). The AARIA Agent Architecture: An Example of Requirements-Driven Agent-Based System Design. In *Proceedings of First International Conference on Autonomous Agents*, pages 482–483.

[Parunak and Brueckner, 2001] Parunak, H. V. D. and Brueckner, S. (2001). Entropy and Self-organization in Multi-Agent Systems. In *Proceedings of the International Conference on Autonomous Agents*, Montreal, Canada.

[Parunak et al., 2001] Parunak, H. V. D., Brueckner, S., Sauter, J., and Posdamer, J. (2001). Mechanisms and Military Applications for Synthetic Pheromones. In *Proceedings of Workshop on Autonomy Oriented Computation, Agents*, pages 58–67, Montreal, Canada.

[Peeters et al., 1999] Peeters, P., Valckenaers, P., Wyns, J., and Brueckner, S. (1999). Manufacturing Control Algorithm and Architecture. In *Proceedings of the Second International Workshop on Intelligent Manufacturing Systems*, pages 877–888, Leuven, Belgium.

[Peng et al., 1998] Peng, Y., Finin, T., Labrou, Y., Chu, B., Long, J., Tolone, W., and Boughannam, A. (1998). A Multi-Agent System for Enterprise Integration. In *Proceedings of Third*

International Conference on the Practical Application of Intelligent Agents and Multi-Agents, London.

[Pham and Pham, 2001] Pham, D. and Pham, P. (2001). *Computational Intelligence for Manufacturing.* CRC Press.

[Quintas and Leitão, 1997] Quintas, A. and Leitão, P. (1997). A Manufacturing Cell Controller Architecture. In Sullivan, W. G. and Ahmad, M. M., editors, *Proceedings of the Flexible Automation and Intelligent Manufacturing Conference*, pages 483–493, Middlesbrough, England. Begell House Inc.

[Rabelo and Camarinha-Matos, 1994] Rabelo, R. and Camarinha-Matos, L. (1994). Multi-Agent based Dynamic Scheduling. *International Journal on Robotics and Computer Integrated Manufacturing*, II(4):303–310.

[Rabelo et al., 1999] Rabelo, R., Camarinha-Matos, L., and Afsarmanesh, H. (1999). Multi-Agent-Based Agile Scheduling. *Journal of Robotics and Autonomous Systems*, 27(1-2):15–28.

[Ranky, 1990] Ranky, P. (1990). *Flexible Manufacturing Cells and Systems.* CIMWare.

[RapidCIM, 2003] RapidCIM (2003). Rapidcim web. http://tamcam.tamu.edu/rapidcim/rapidcim.htm.

[Rembold et al., 1993] Rembold, U., Nnaji, B., and Storr, A. (1993). *Computer Integrated Manufacturing and Engineering.* Addison-Wesley.

[Reveliotis and Ferreira, 1996] Reveliotis, S. and Ferreira, P. (1996). Deadlock Avoidance Policies for Automated Manufacturing Cells. *IEEE Transactions on Robotics and Automation*, 12:845–857.

[Ritter et al., 2002] Ritter, A., Baum, W., Höpf, M., and Westkäuper, E. (2002). Agentification for Production Systems. In *Proceedings of European Joint Conferences on Theory and Practice of Software*, Grenoble, France.

[RobComm, 1999] RobComm (1999). *RobComm User's Guide, version 3.0/3.* ABB Flexible Automation.

[Rumbaugh et al., 1998] Rumbaugh, J., Jacobson, I., and Booch, G. (1998). *The Unified Modelling Language Reference Manual.* Addison-Wesley.

[Russel and Norvig, 1995] Russel, S. and Norvig, P. (1995). *Artificial Intelligence, A Modern Approach.* Prentice-Hall.

[Sadeh and Fox, 1989] Sadeh, N. and Fox, M. (1989). CORTES: An Exploration into Micro-Opportunistic Job-Shop Scheduling. In *Proceedings of Workshop on Manufacturing Production Scheduling*, Detroit.

[Saint-Germain et al., 2003] Saint-Germain, B., Valckenaers, P., Brussel, H. V., Hadeli, Bochmann, O., Zamfirescu, C., and Verstraete, P. (2003). Multi-agent Manufacturing Control: An Industrial Case Study. In *Proceedings of the 7th IFAC Workshop on Intelligent Manufacturing Systems*, pages 227–232, Budapest, Hungary.

[Sauter and Massotte, 2001] Sauter, T. and Massotte, P. (2001). Enhancement of Distributed Product Systems through Software Agents. In *Proceedings of 8th IEEE International Conference on Emerging Technologies and Factory Automation*, volume 1, pages 267–272, Antibes, France.

[Searle, 1969] Searle, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press.

[Sen and Weiss, 1999] Sen, S. and Weiss, G. (1999). Learning in Multi-Agent Systems. In Weiss, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 259–298. The MIT Press, Cambridge, MA, USA.

[Shen and Norrie, 1998] Shen, W. and Norrie, D. (1998). An Agent-based Approach Dynamic Manufacturing Scheduling. In *Workshop Notes of the Agent-based Manufacturing Workshop at Autonomous Agents.*

[Shen and Norrie, 1999] Shen, W. and Norrie, D. (1999). Agent-based Systems for Intelligent Manufacturing: A State-of-the-Art Survey. *Knowledge and Information Systems: an International Journal*, 1(2):129–156.

[Shen et al., 1998] Shen, W., Xue, D., and Norrie, D. (1998). An Agent-Based Manufacturing Enterprise Infrastructure for Distributed Integrated Intelligent Manufacturing Systems. In *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agents*, pages 23–25, London, UK.

[Silberschatz and Peterson, 1991] Silberschatz, A. and Peterson, G. (1991). *Operating Systems Concepts.* Addison-Wesley, Massachusetts.

[Silva and Valette, 1989] Silva, M. and Valette, R. (1989). Petri Nets and Flexible Manufacturing. In *Advances in Petri Nets*, volume 424 of *Lectures Note in Computer Science*, pages 374–417. Springer Verlag.

[Smith, 1980] Smith, R. (1980). Contract Net Protocol: High-Level Communication and Control in a Distributed Solver. *IEEE Transactions on Computers*, C-29(12):1104–1113.

[Sousa and Ramos, 1999] Sousa, P. and Ramos, C. (1999). A Distributed Architecture and Negotiation Protocol for Scheduling in Manufacturing Systems. *Computers in Industry*, 38(2):103–113.

[Sousa et al., 1999] Sousa, P., Silva, N., Heikkilä, T., Kollingbaum, M., and Valckenaers, P. (1999). Aspects of Co-operation in Distributed Manufacturing Systems. In *Proceedings of the Second International Workshop on Intelligent Manufacturing Systems*, pages 695–717, Leuven, Belgium.

[Sugimura et al., 1996] Sugimura, N., Hiroi, M., Moriwaki, T., and Hozumi, K. (1996). A Study on Holonic Scheduling for Manufacturing System of Composite Parts. In *Japan/USA Symposium on Flexible Manufacturing.*

[Suzuki and Murata, 1983] Suzuki, I. and Murata, T. (1983). A Method for Stepwise Refinements and Abstractions of Petri nets. *Journal of Computer and Systems Science*, 27:51–76.

[Swamidass, 2000] Swamidass, P. M., editor (2000). *Encyclopedia of Production and Manufacturing Management.* Kluwer Academic Publishers.

[Tanaya et al., 1997] Tanaya, P., Detand, J., and Kruth, J. (1997). A Holonic Machine Controller: a Study and Implementation of Holonic Behaviour to Current NC Controller. *Computers in Industry*, 33(2-3):323–333.

[Teunis et al., 1998] Teunis, G., Leitão, P., and Madden, M. (1998). A New Architecture for Flexible Shop Control Systems. In *Proceedings of Integration in Manufacturing Conference*, pages 699–709, Gotengorg, Sweden.

[Tharumarajah et al., 1996] Tharumarajah, A., Wells, A., and Nemes, L. (1996). Comparison of the Bionic, Fractal and Holonic Manufacturing Systems Concepts. *International Journal of Computer Integrated Manufacturing*, 9(3):217–226.

[Tönshoff et al., 2000] Tönshoff, H.-K., Seilonen, I., Teunis, G., and Leitão, P. (2000). A Mediator-based Approach for Decentralised Production Planning, Scheduling and Monitoring. In *Proceedings of 2nd CIRP International Seminar on Intelligent Computation in Manufacturing Engineering*, pages 113–118, Capri, Italy.

[Tönshoff and Winkler, 1996] Tönshoff, H.-K. and Winkler, M. (1996). Shop Control for Holonic Manufacturing Systems. *Manufacturing Systems*, 3(25):1–5.

[Torsun, 1995] Torsun, I. (1995). *Foundations of Intelligent Knowledge-Based Systems.* Academic Press.

[Tsourveloudis and Phillis, 1998] Tsourveloudis, N. and Phillis, Y. (1998). Manufacturing Flexibility Measurement: A Fuzzy Logic Framework. *IEEE Transactions on Robotics and Automation*, 14(4):513–524.

[Upton, 1992] Upton, D. (1992). Flexible Structure for Computer Controlled Manufacturing System. *Manufacturing Review*, 5(1):58–74.

[Vaario and Ueda, 1996] Vaario, J. and Ueda, K. (1996). Self-Organisation in Manufacturing Systems. In *Japan-USA Symposium on Flexible Automation*, pages 1481–1484, Boston, US.

[Vaario and Ueda, 1997] Vaario, J. and Ueda, K. (1997). Biological Concept of Self-organisation for Dynamic Shop Floor Configuration. In *Proceedings of Advanced Product Management Systems (APMS'97)*, pages 55–66.

[Valckenaers et al., 1999] Valckenaers, P., Heikkilä, T., Baumgaertel, H., McFarlane, D., and Courtois, J. (1999). Towards a Novel Manufacturing Control Principle. In *Proceedings of the Second International Workshop on Intelligent Manufacturing Systems*, pages 871–875, Leuven, Belgium.

[Vallete, 1979] Vallete, R. (1979). Analysis of Petri nets by Stepwise Refinements. *Journal of Computer and Systems Science*, 18:35–46.

[Viswanadham et al., 1990] Viswanadham, N., Narahari, Y., and Johnson, T. (1990). Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems Using Petri Nets Models. *IEEE Transactions on Robotics and Automation*, 6(6):713–722.

[Vrba, 2003] Vrba, P. (2003). JAVA-Based Agent Platform Evaluation. In Marík, V., Müller, J., and Pechoucek, M., editors, *Multi-Agent Systems and Applications III, 3rd International/Central and Eastern European Conference on Multi-Agent Systems*, volume 2691 of *Lecture Notes in Artificial Intelligence*, pages 47–58. Springer-Verlag.

[Vyatkin and Hanisch, 2003] Vyatkin, V. and Hanisch, H. (2003). Verification of Distributed Control Systems in Intelligent Manufacturing. *Journal of Intelligent Manufacturing*, 14(1):123–136.

[Vyatkin et al., 2001] Vyatkin, V., Hanisch, H.-M., and Ivanov, G. (2001). Application of Formal Methods for Deep Testing of Controllers in Holonic Systems. In *Proceedings of the First*

IEEE International Conference on Information Technology in Mechatronics (ITM2001), pages 53–58, Istanbul, Turkey.

[Wang and Norrie, 2001] Wang, L. and Norrie, D. (2001). Process Planning and Control in Holonic Manufacturing Environment. *Journal of Applied Systems Studies, Special Issue on Industrial Applications of Holonic and Multi-Agents Systems*, 2(1).

[Warneke, 1993] Warneke, H. (1993). *The Fractal Company.* Springer-Verlag.

[Winkler and Mey, 1994] Winkler, M. and Mey, M. (1994). Holonic Manufacturing Systems. *European Production Engineering.*

[Womack et al., 1990] Womack, J., Jones, D. T., and Roos, D. (1990). *The Machine that Changed the World.* MIT Press, Cambridge.

[Wooldridge, 2002] Wooldridge, M. (2002). *An Introduction to Multi-Agent Systems.* John Wiley & Sons.

[Wooldridge and Ciancarini, 2000] Wooldridge, M. and Ciancarini, P. (2000). Agent-Oriented Software Engineering: The State of the Art. In Ciancarini, P. and Wooldridge, M., editors, *Agent-Oriented Software Engineering*, volume 1957 of *Lecture Notes in Computers Science.* Springer-Verlag.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2):115–152.

[Wyns, 1999] Wyns, J. (1999). *Reference Architecture for Holonic Manufacturing Systems - the Key to Support Evolution and Reconfiguration.* PhD thesis, Katholieke Universiteit Leuven, Belgium.

[Yoon and Lee, 2000] Yoon, H. and Lee, D. (2000). Deadlock-Free Scheduling for Automated Manufacturing Cells. In *Proceedings of Sixth International Conference on Control, Automation, Robotics and Vision.*

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy Sets. *Information and Control*, 8:338–353.

[Zhou et al., 1999] Zhou, B., Wang, L., and Norrie, D. (1999). Design of Distributed Real-time Control Agents for Intelligent Manufacturing Systems. In *Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems*, pages 237–244, Leuven, Belgium.

Appendix A

Qualitative Analysis of Petri net Models

This appendix intends to provide some basic concepts about the validation of the Petri net models using the qualitative analysis.

A.1 Basic Concepts about Qualitative Analysis

The qualitative analysis allows verifying the structural and behavioural properties of the model, extracting conclusions about the functionality of the system, such as the existence of deadlocks, the bounded capacity of resources, and the existence of structural and behavioural conflicts in the system [Feldmann et al., 1996]. Several properties can be verified during the qualitative analysis of a Petri net model [Desrochers and Al-Jaar, 1995]:

- **Boundedness**: A Petri net is k-bounded if each place in the net gets at most k tokens for all markings in the reachability set. This guarantees that the number of resources (number of tokens at the places, which can be number of machines, places in the buffers, etc.) is limited. If the model is k-bounded and $k = 1$, then the Petri net is safe.

- **Reversibility**: A Petri net is reversible if the initial marking is reachable from all reachable markings, guaranteeing that the model can reinitialise from itself.

- **Liveness**: A Petri net is live if for any marking in the reachability set, it is possible to fire any transition in the net. This property is very important since the liveness guarantees the absence of deadlocks. In other words, a reachable marking m is a deadlock if no transition is enabled in m, which allows changing to another reachable marking. A Petri net is deadlock free if no reachable marking is a deadlock.

- **Conservativeness**: A Petri net is conservative if it is not created or destroyed new tokens in the net. As example, this guarantees that the parts (material under processing aiming the final product) are not destroyed from the system.

The available methods for the analysis of behavioural properties are the following: reachability tree and linear algebra method [Desrochers and Al-Jaar, 1995].

The first method is a graph built upon the reachability tree of the Petri net, and shows the evolution of the tokens over the net model. The nodes of the tree represent a marking in the net that corresponds to a state of the system. The arcs represents the transitions that can occur from each node. The building of reachability tree is done from the definition of an initial marking, m_0, which implies that this method is dependent on the initial marking of the Petri net. For that marking it is analysed which transitions lead to a new marking m. If this marking still not exist, it is attributed a name to it, m_k, different from the existing ones. For each new marking in the tree it is repeated the process done for m_0. In case of a marking, which no transition trigger a new marking, the situation is designated by deadlock.

From the analysis of the reachability tree it is possible to verify the behavioural properties, such as the existence of deadlocks and the boundedness. However, for complex Petri nets the construction of the reachability tree explodes, making impracticable the structural analysis using this method.

The second method is based in mathematical linear algebra background using the incidence matrix. The incidence matrix, W, represents the Petri net structure in an algebraic form, describing how the nodes in the Petri net are interconnected, and in this way describing the evolution of the tokens by the several places, according the trigger of a sequence of transitions. Formally, the incidence matrix can be defined as a $(n \times m)$ matrix of integers,

$$W = [a_{ij}] \tag{A.1}$$

where a_{ij} represents the number of marks removed from the input places (negative integers), or the number of marks stored in the output places (positive integers). The elements of the incidence matrix can be calculated by,

$$W(p,t) = I(p,t) - O(p,t)$$

This method is based in the following equation,

$$m_k = m_0 + W.T \tag{A.2}$$

which allows to determine a new marking in the model (m_k) from the knowledge of the incidence matrix (W), the initial marking (m_0) and a sequence of triggered transitions (T). In order to support the analysis of the behavioural properties it is defined the concepts of P-invariants and T-invariants, which uses the incidence matrix.

The P-invariant is a $(n \times 1)$ non-negative integer vector x satisfying:

$$x^T.W = 0 \tag{A.3}$$

Considering $r = rank(W)$, there are $(n - r)$ minimal P-invariants and a Petri net is covered by P-invariants if all its places belong to some P-invariants. A siphon is a set of places such that every transition that outputs to one of these places also inputs from one of these places. Thus, if all the places in a siphon have no tokens, then the corresponding set of places will never have a token.

Similarity, a T-invariant is a $(m \times 1)$ non-negative integer vector y satisfying:

$$W.y = 0 \tag{A.4}$$

Using the P-invariants it is possible to verify the liveness and boundedness of a Petri net: a Petri net is live and bounded if it is covered by P-invariants, all the P-invariants are marked with tokens, and none of the siphons is ever cleared of tokens. Additionally, using the P-invariants and T-invariants it is possible to determine the properties of the Petri net [Desrochers and Al-Jaar, 1995, Colombo, 1998]:

- A Petri net is conservative if there exists an $(n \times 1)$ vector x such that $x^T.W = 0$;
- A Petri net is repetitive if there exists a $(m \times 1)$ vector y such that $W.y \geq 0$;
- A Petri net is not reversible if $W.y = 0$ admits only the trivial solution. Even a nontrivial solution only guarantees partial reversibility.

Like the reachability tree method, the analysis of Petri net using linear algebra methods over the incidence matrix has its disadvantages, such as the solution for the equation (A.2) is only a necessary condition to verify the reachability, and the determination of the liveness requires a hard search for the solution.

A.2 Qualitative Analysis of the ADACOR Holon Petri net Models

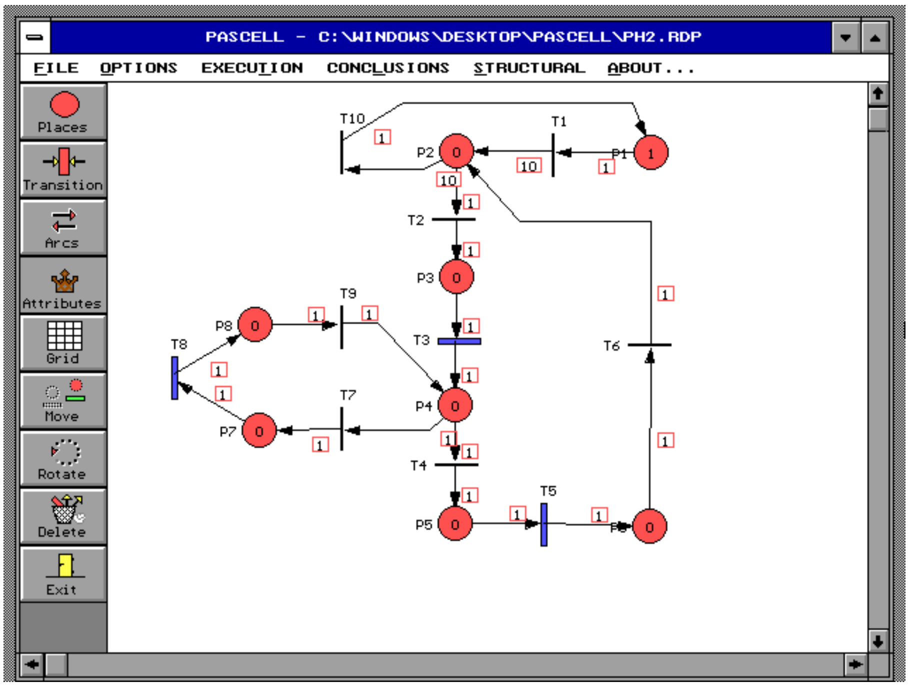
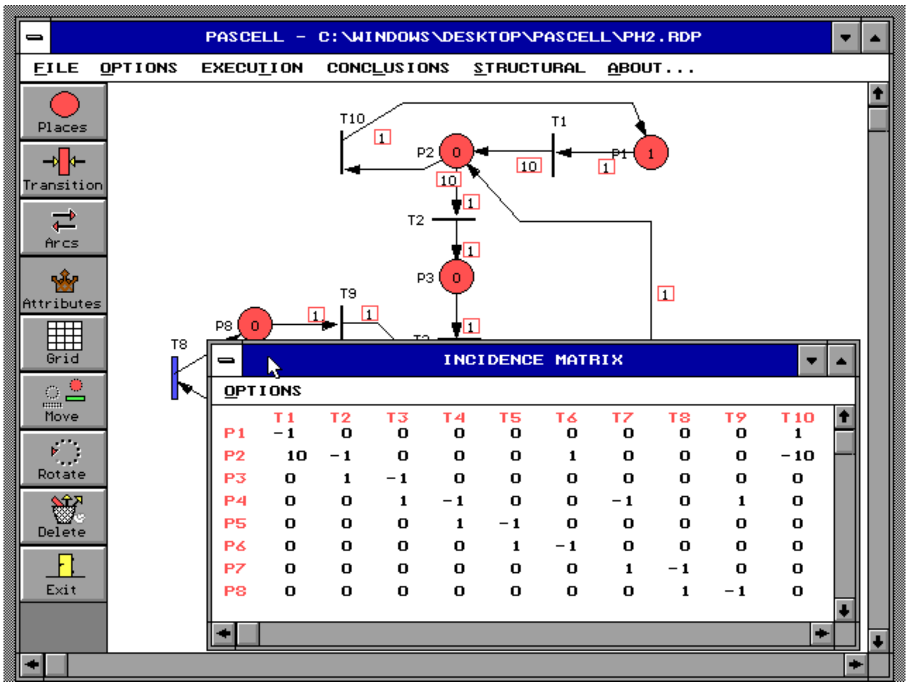
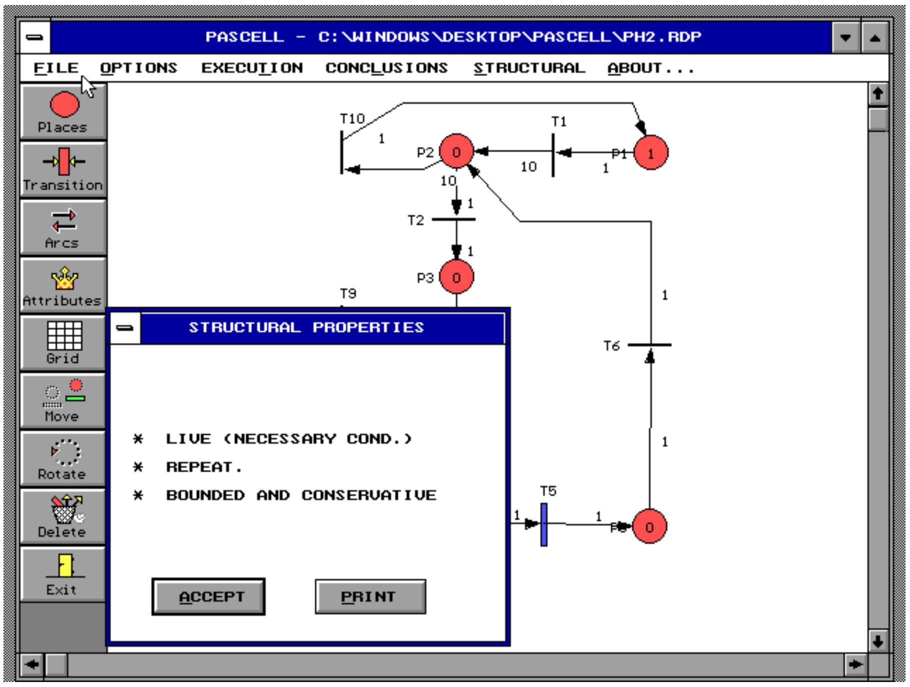

PASCELL - C:\WINDOWS\DESKTOP\PASCELL\PH2.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
T10
T1
P1
P2
T2
P3
T3
T9
P8
T8
T7
P7
P4
T4
P5
T5
P6
T6
10
1

Figure A.1: Edition of the Product Holon Model


PASCELL - C:\WINDOWS\DESKTOP\PASCELL\PH2.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
INCIDENCE MATRIX
OPTIONS

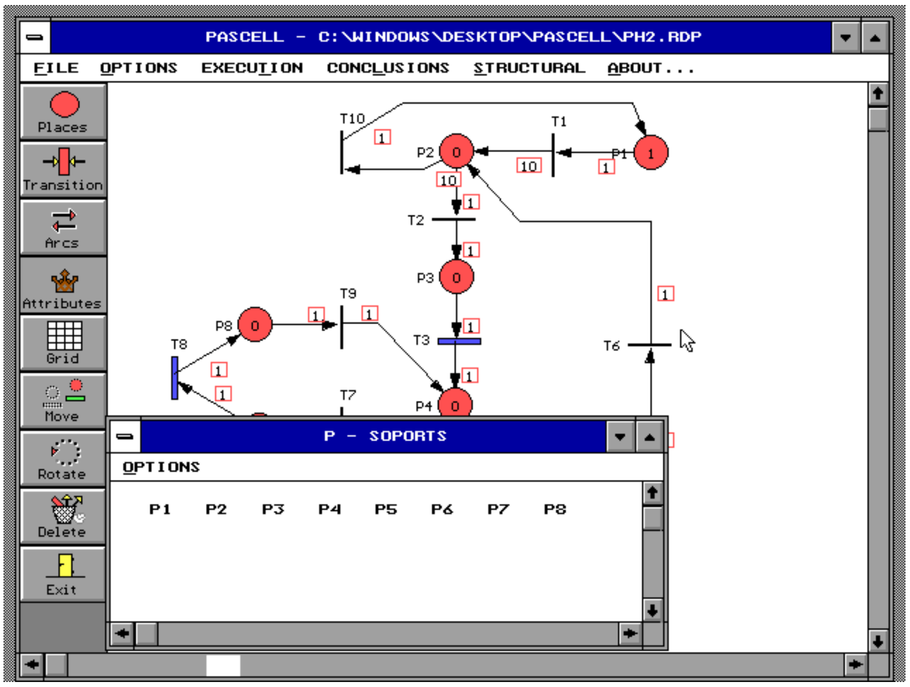
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
P1	−1	0	0	0	0	0	0	0	0	1
P2	10	−1	0	0	0	1	0	0	0	−10
P3	0	1	−1	0	0	0	0	0	0	0
P4	0	0	1	−1	0	0	−1	0	1	0
P5	0	0	0	1	−1	0	0	0	0	0
P6	0	0	0	0	1	−1	0	0	0	0
P7	0	0	0	0	0	0	1	−1	0	0
P8	0	0	0	0	0	0	0	1	−1	0

Figure A.2: Incidence Matrix for the Product Holon Model


PASCELL - C:\WINDOWS\DESKTOP\PASCELL\PH2.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
STRUCTURAL PROPERTIES
ACCEPT
PRINT

* LIVE (NECESSARY COND.)
* REPEAT.
* BOUNDED AND CONSERVATIVE

Figure A.3: Structural Analysis of the Product Holon Model


PASCELL - C:\WINDOWS\DESKTOP\PASCELL\PH2.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
P - SOPORTS
OPTIONS

P1 P2 P3 P4 P5 P6 P7 P8

Figure A.4: P-invariants for the Incidence Matrix of the Product Holon Model

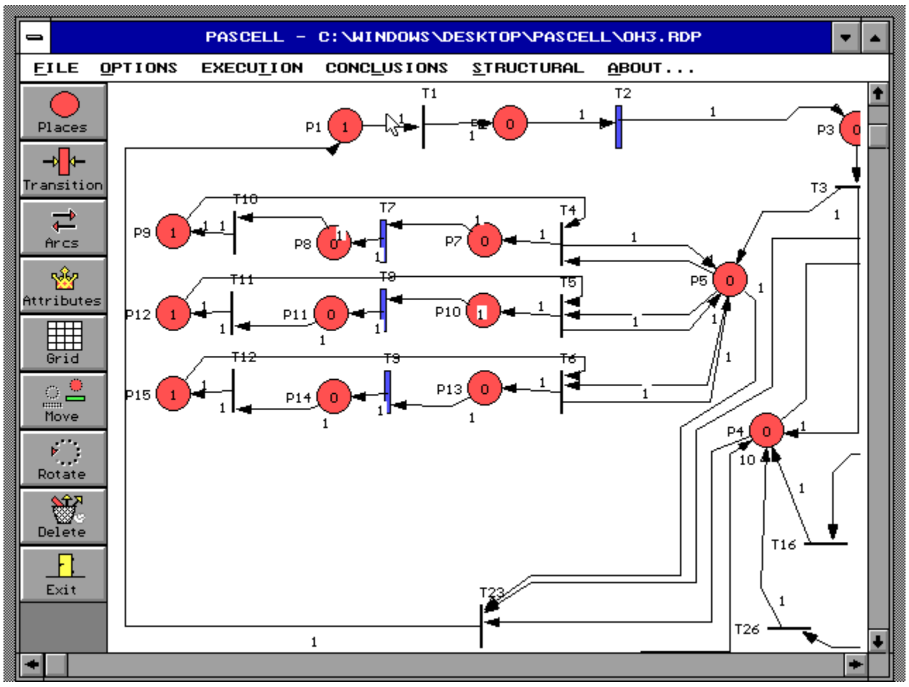
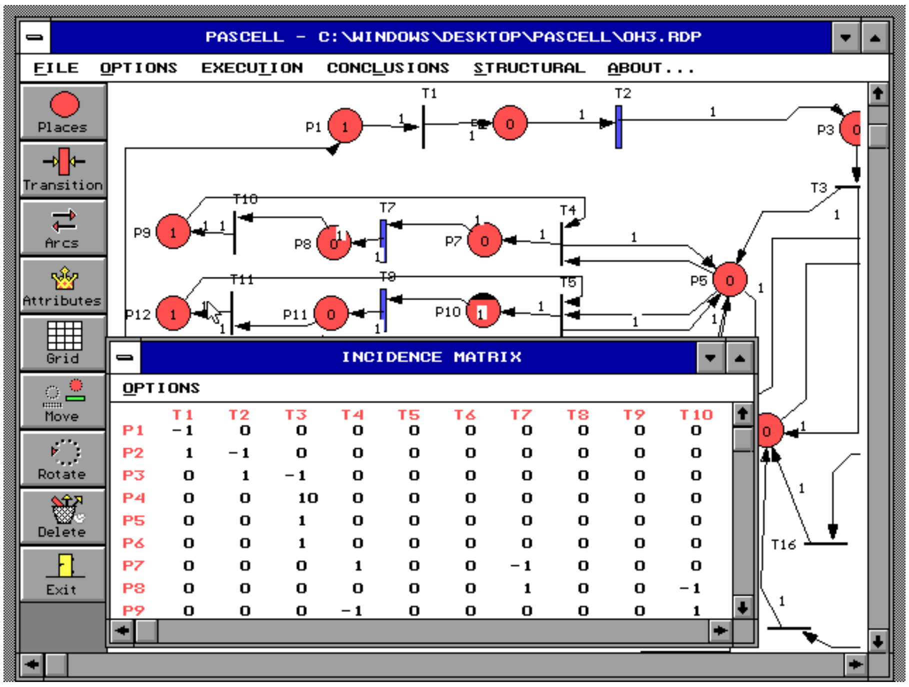
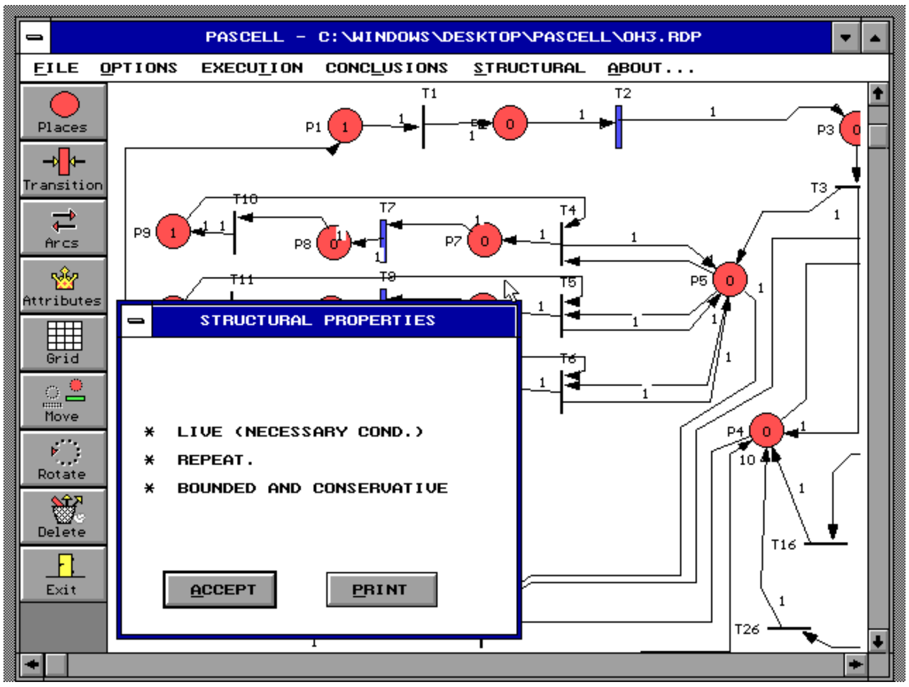


Figure A.5: Edition of the Operational Holon Model


PASCELL - C:\WINDOWS\DESKTOP\PASCELL\OH3.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
INCIDENCE MATRIX
OPTIONS

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
P1	-1	0	0	0	0	0	0	0	0	0
P2	1	-1	0	0	0	0	0	0	0	0
P3	0	1	-1	0	0	0	0	0	0	0
P4	0	0	10	0	0	0	0	0	0	0
P5	0	0	1	0	0	0	0	0	0	0
P6	0	0	1	0	0	0	0	0	0	0
P7	0	0	0	1	0	0	-1	0	0	0
P8	0	0	0	0	0	0	1	0	0	-1
P9	0	0	0	-1	0	0	0	0	0	1

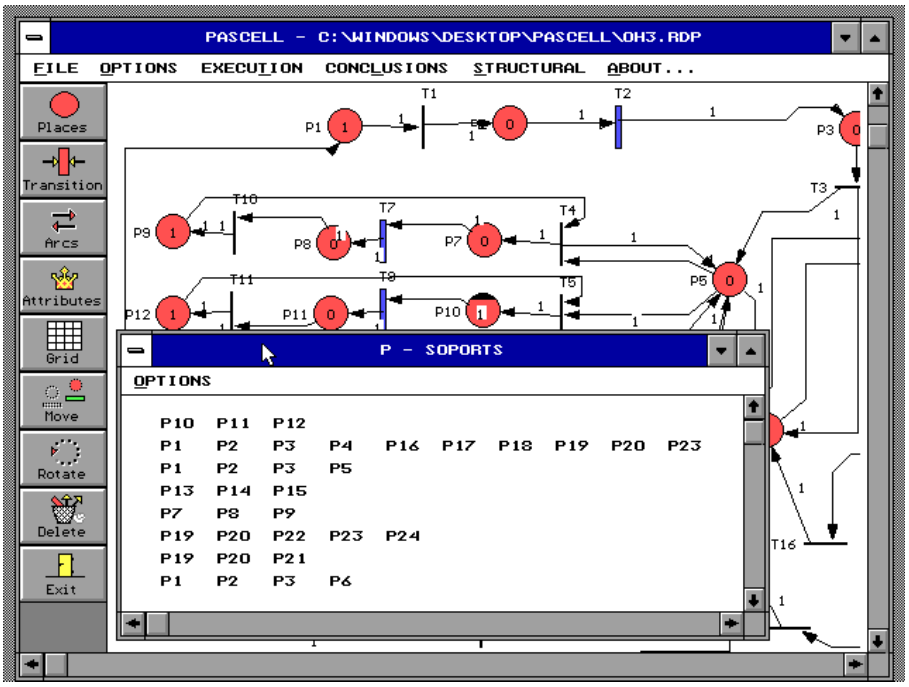
Figure A.6: Incidence Matrix for the Operational Holon Model


PASCELL - C:\WINDOWS\DESKTOP\PASCELL\OH3.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
ACCEPT
PRINT

```
STRUCTURAL PROPERTIES

*  LIVE (NECESSARY COND.)
*  REPEAT.
*  BOUNDED AND CONSERVATIVE
```

Figure A.7: Structural Analysis of the Operational Holon Model


PASCELL - C:\WINDOWS\DESKTOP\PASCELL\OH3.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
OPTIONS

```
P - SOPORTS

P10   P11   P12
P1    P2    P3    P4    P16   P17   P18   P19   P20   P23
P1    P2    P3    P5
P13   P14   P15
P7    P8    P9
P19   P20   P22   P23   P24
P19   P20   P21
P1    P2    P3    P6
```

Figure A.8: P-invariants for the Incidence Matrix of the Operational Holon Model

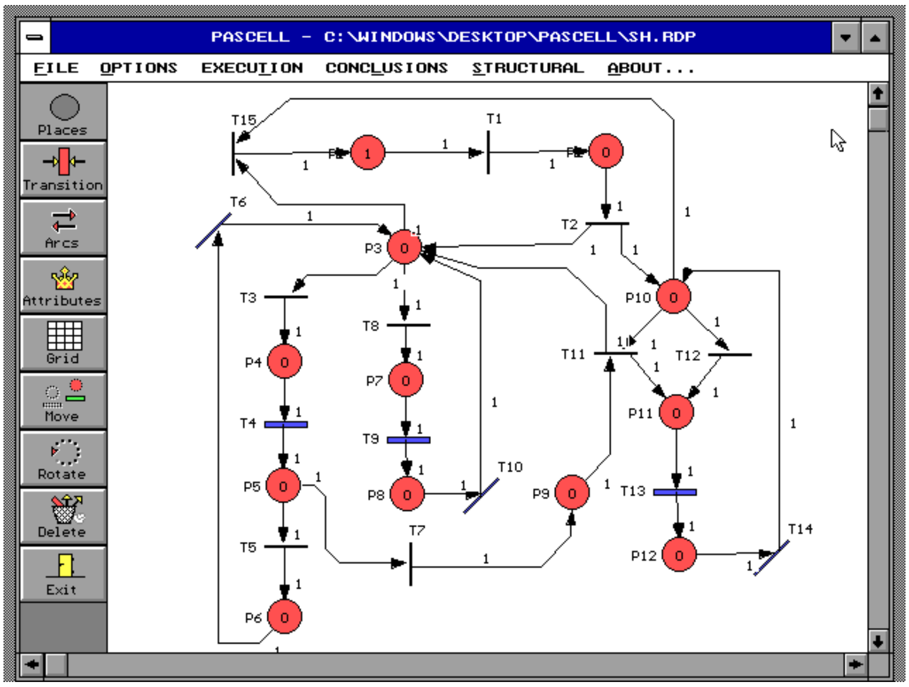
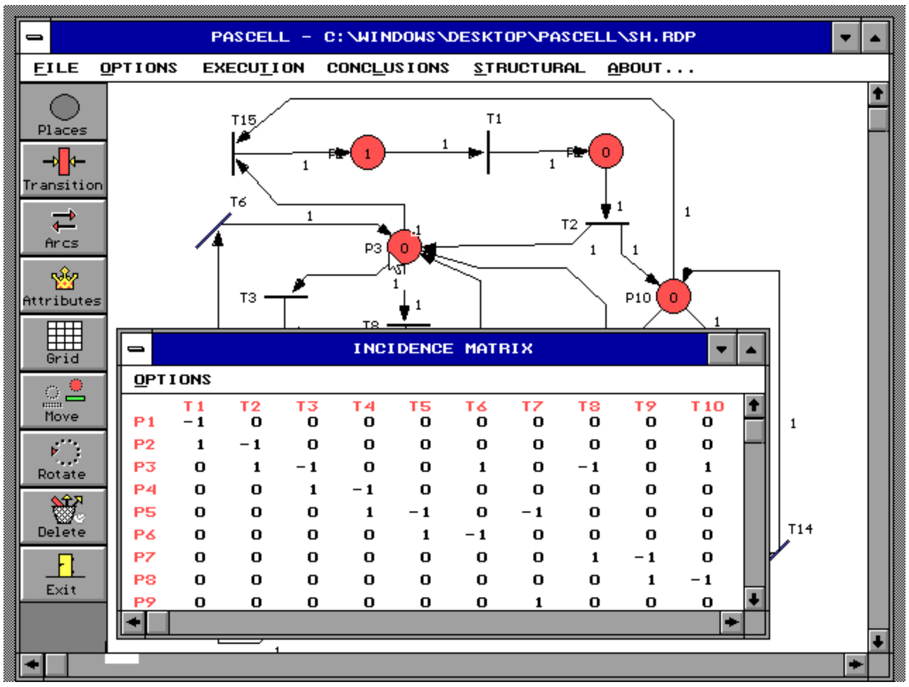
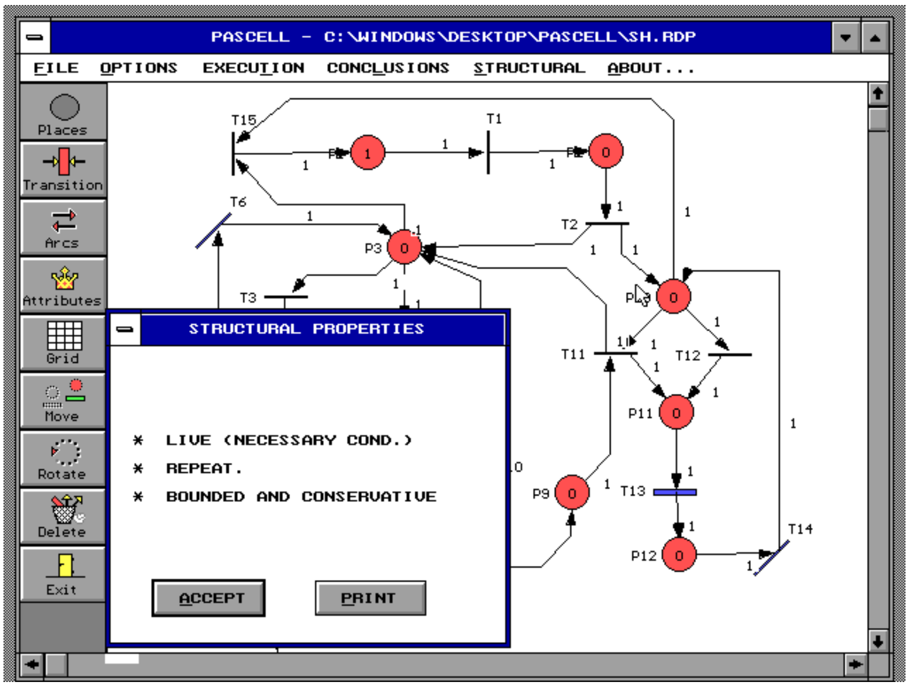


Figure A.9: Edition of the Supervisor Holon Model


PASCELL - C:\WINDOWS\DESKTOP\PASCELL\SH.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
INCIDENCE MATRIX
OPTIONS

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
P1	-1	0	0	0	0	0	0	0	0	0
P2	1	-1	0	0	0	0	0	0	0	0
P3	0	1	-1	0	0	1	0	-1	0	1
P4	0	0	1	-1	0	0	0	0	0	0
P5	0	0	0	1	-1	0	-1	0	0	0
P6	0	0	0	0	1	-1	0	0	0	0
P7	0	0	0	0	0	0	0	1	-1	0
P8	0	0	0	0	0	0	0	0	1	-1
P9	0	0	0	0	0	0	1	0	0	0

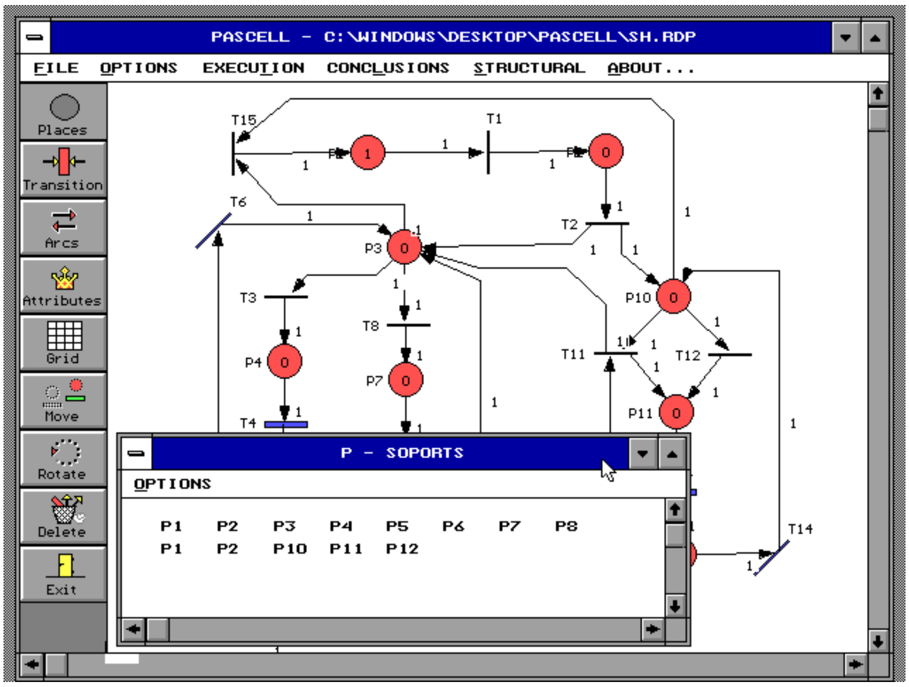
Figure A.10: Incidence Matrix for the Supervisor Holon Model



STRUCTURAL PROPERTIES

```
*  LIVE (NECESSARY COND.)
*  REPEAT.
*  BOUNDED AND CONSERVATIVE
```

Figure A.11: Structural Analysis of the Supervisor Holon Model


PASCELL - C:\WINDOWS\DESKTOP\PASCELL\SH.RDP
FILE OPTIONS EXECUTION CONCLUSIONS STRUCTURAL ABOUT...
Places
Transition
Arcs
Attributes
Grid
Move
Rotate
Delete
Exit
T15
T1
T6
T2
T3
T8
T4
T11
T12
T14
P3
P4
P7
P10
P11
OPTIONS

P - SOPORTS

```
P1   P2   P3   P4   P5   P6   P7   P8
P1   P2   P10  P11  P12
```

Figure A.12: P-invariants for the Incidence Matrix of the Supervisor Holon Model

Appendix B

Implementation of Scheduling Mechanisms

B.1 Centralised Scheduling

In our implementation, the centralised scheduling mechanism, dealing with the multiple machines and multiple jobs problem, uses a simple but non-optimised algorithm that guarantees rapid and reliable scheduling. The structure of the scheduling mechanism, executed by the method generateSchedule(), is illustrated in the Figure B.1.

Initially, the scheduling mechanism executes the findAvailability4EachOperation method, trying to find availability to execute each operation through the actual capacity of the supervisor holon coordination domain. Since each operation has requirements, $B_{ik} = \{B_{ikw} | w \in I\}$, associated to the type of machine that can perform the task, it is possible to verify if any available machine fulfils those requirements, returning a list of machines that can perform each operation, $\mathcal{R}_{ik}$ (list of possible resources that can execute the operation k that belongs to the order i). The list $\mathcal{R}_{ik}$ is achieved, using rule-oriented programming, matching the requirements of the operation with the skills of each resource $S_j = \{S_{jw} | w \in I\}$. The list $\mathcal{R}_{ik}$ allows choosing between alternative scheduling plans. The execution of this method can be avoided if the product holon pass to the task holon alternative process plans.

After having the list of possible resources to execute each operation, it is invoked the fillWaitingListOfMachines method, that fulls the waiting list of each machine with the possible operations available to start at the beginning, i.e. at $currentTime = 0$. After that, the schedule mechanism enters in a *do...while* cycle until all operations be scheduled.

The allocateWO2Machine method spans all available machines that are in the idle state at the considered $currentTime$, analysing its waiting list and trying to allocate an operation to

```
public List generateSchedule(){
    int currentTime=0;
    boolean allWOScheduled=false;
    findAvailability4EachOperation();
    fillWaitingListOfMachines();
    do {
        for(int i=0; i < noMachines); i++) {
            Resource mach = Resource();
            mach=(Resource)listOfMachines.get(i);
            if (mach.getMachineStatus == idle) {
                 allocateWO2Machine();
            }
            if (mach.getMachineStatus != idle) {
                 if (mach.getEndOfExecution <= currentTime) {
                       mach.addWO2ResourceSchedule(wo);
                  }
            }
            findNextTimeStep();
            clearWaitingLists();
            fillWaitingListOfAvailableMachines();
            allWOScheduled=searchScheduleStatus();
        }
    } while (!allWOScheduled)
    return listSchedule;
}
```

Figure B.1: Structure of the Scheduling Algorithm

the machine. Initially, it is selected one operation from the waiting list of the machine using a dispatching rule, such as the EDD or SPT. After selecting the operation, it is necessary to remove the selected operation from the waiting list of other machines, in order to be not considered when those machines be analysed to allocate operations. This mechanism guarantees that precedent operations belonging to the same production order are allocated to the same machine, since the operation requirements allow that.

The findNextTimeStep method determines the next iterative round, searching the remaining time of the first machine to finish the execution of actual operation. This value is added to the actual *currentTime* and is removed from the time2FinishExecution parameter of each machine in execution. The fillWaitingListOfAvailableMachines method fills the waiting list of the available machines at *currentTime* value, with the available operations to start at this moment.

B.2 Distributed Scheduling

The implementation of the distributed scheduling requires the development of mechanisms to elaborate the price to execute a work order and to evaluate the proposals to select the best one.

B.3 Elaboration of the Price

The operational holon elaborates the price to execute a work order, to be included in the proposal to be sent to the task holon that announced the work order.

The price function determines the price based in two main components: the fixed costs and the profit margin. The fixed costs are the amount of the cost that is required to execute the operation and the profit margin is the amount that can be adapted according to the market laws.

The price, p_{jik}, to be proposed by the operational holon j to execute the work order ik, is calculated using the following function that comprises the fixed costs and the profit margin:

$$p_{jik} = C_s + C_t + C_p * d_{ik} + C_b * (2 - e^{-\sigma * \beta} * (1 - \gamma))$$

where C_s is the cost associated to the setup execution, calculated multiplying the set-up time by the cost to have the machine stopped, C_t is the cost associated to the acquisition of new tools for the execution of the work order, and C_p is related to the cost associated to maintain that machine working (electricity power, compressed air, maintenance, etc). The parameter C_b reflects the investment done to buy the machine.

In order to have a dynamic price, the profit margin is regulated by the market laws, increasing or decreasing the final price in function of the actual load of the resource and in the actual bid acceptance rate. The parameter σ is the price saturation coefficient associated to each resource, the parameter β is the quotient between the actual load and the capacity for each resource, and γ is the acceptance rate ($0 \leq \gamma \leq 1$).

As an example, considering C_s=100, C_t=100, σ=4, C_p=1, C_b=5000 and d_{ik}=300 seconds, it is obtained the 3D graphic represented in Figure B.2, that represents the variation of price in function of the actual resource capacity and the acceptance rate. This graphic shows that the price increases with the increase of the actual load or the acceptance rate. In an opposite way, when the operational holon has low capacity allocated or low acceptance rate, it decreases the price in order to get more work orders.

In case of transporter resources, such as automated guided vehicles, the cost is calculated using the following formula:

$$p_{jik} = C_{transp} * (|L_{actual} - L_{source}| + |L_{dest} - L_{source}|) + C_b * (2 - e^{-\sigma * \beta} * (1 - \gamma))$$

where L_{source} is the source localisation of the work station, L_{dest} is the destination of the transport, and L_{actual} is the actual localisation of the transport resource. In this case, the β parameter takes in consideration the actual autonomy of the resource.

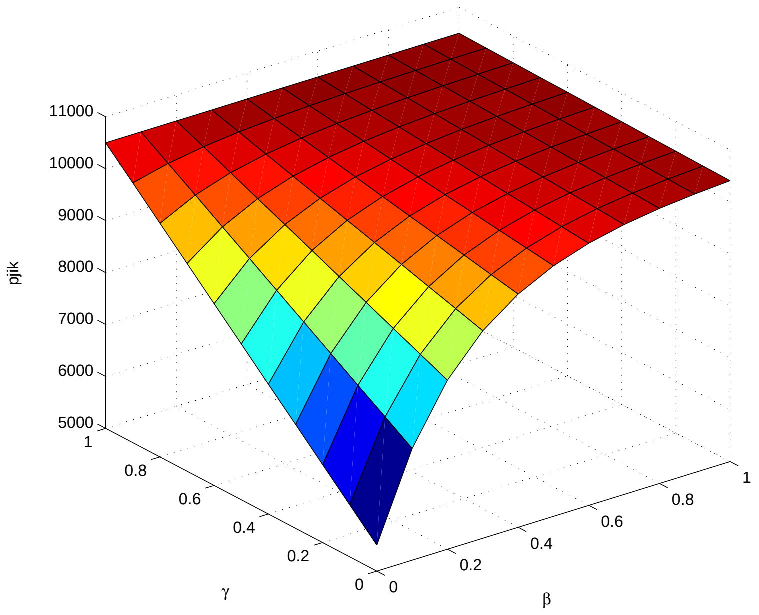


Figure B.2: Price Variability according to the Manufacturing Load and Acceptance Rate

The operational holon uses the learned knowledge of previous bids, by introducing the γ parameter to adjust the final price to be proposed to the task holons: reducing the price if the acceptance rate of the bids is low or increasing the price in opposite case.

B.4 Evaluation of the Proposals

The decision to select the best proposal, elaborated by the task holon, combines the analysis of the proposed price, the location of the resource, the proposed due date and the confidence about the holon, sorting the proposals according to the following heuristic function:

$$bidRate = w_p * p_{jik} + w_{dd} * Cdd_{jik} + w_l * Cl_{jik} + w_q * C_q + w_c * (1 - C_r)$$

where w_p is the price weight, w_l is the location weight, w_{dd} is the due date weight, w_q is a dynamic value that allow to evaluate different types of proposals (for example total or partial quantities) and w_c is the confidence weight. The parameter p_{jik} is the proposed price, Cdd_{jik} is the cost related to the due date, Cl_{jik} is the cost related to the location of the resource, used to introduce the cost associated to transport the part between the actual location and the location of the resource that will execute the next work order, C_q is the cost associated to the partial quantity (if applicable), and C_r is the confidence degree of the proponent operational holon.

In order to achieve the best proposal it is necessary to minimise the function *bidRate*. The coefficients w_m reflect the importance of each cost for the final value of the function and their values are dependent of the decision entity.

The confidence degree that reflects the trust that the task holon has in the holon, is calculated based on the knowledge learned in previous interactions, using the following expression:

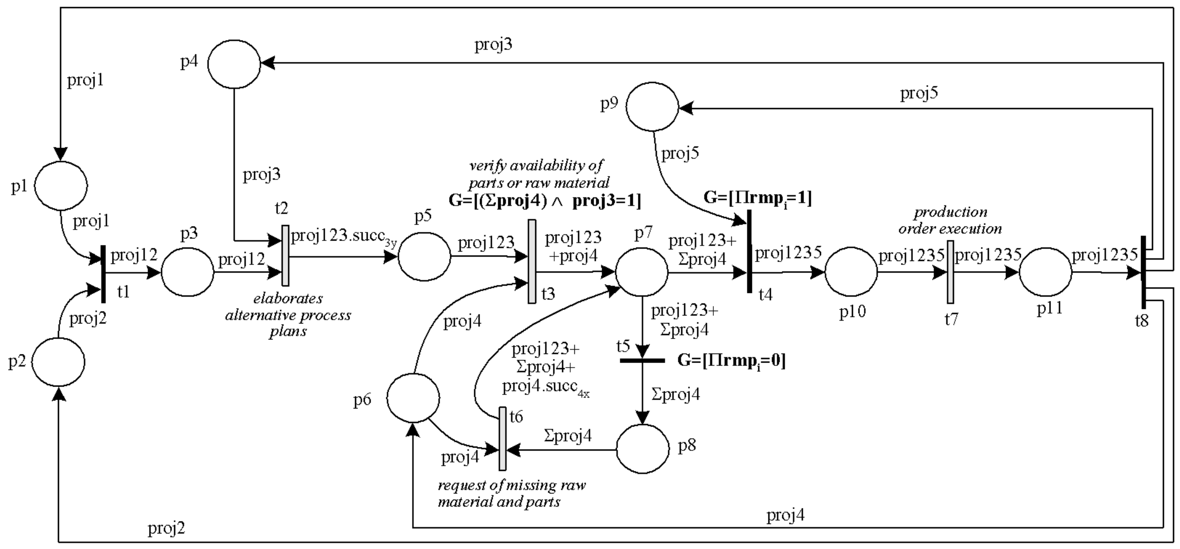
$$C_r = \frac{nWO_s}{nWO_s + nWO_{dd} + nWO_f}$$

where the nWO_s parameter is the number of work orders executed by the resource with success, the nWO_{dd} represents the number of work orders that the resource did not fulfil the due date and the nWO_f represents the number of work orders allocated to the operational holon that were cancelled due to failures. The confidence in a resource is as higher as higher is the C_r value.

Appendix C

Formal Modelling of the Product Holon using Coloured Petri Nets

This appendix shows the application of Coloured Petri net (CPN) to model the ADACOR product holon. This model, illustrated in the Figure C.1, allows to represent simultaneously all products available in the factory and their instances [Leitão et al., 2003d].



$M_0(p1)=\{\Sigma_{[i:1\text{-}n]}ph_i\}$; $M_0(p2)=\{\Sigma_{[i:1\text{-}m]}or_i\}$; $M_0(p4)=\{\Sigma_{[i:1\text{-}l]}wp_i\}$; $M_0(p6)=\{\kappa.\Sigma_{[i:1\text{-}q]}rmp_i\}$, $\kappa \in N$; $M_0(p9)=\{\Sigma_{[i:1\text{-}p]}th_i\}$

Figure C.1: Coloured Petri Net Model for the Product Holon

In this case, each place of the model can contain a set of coloured tokens carrying a data value. The arcs have an attached function (expression), which describes how the state of the CPN changes when the transitions are fired. Guards are associated to the transitions and represent restrictions to the type of data value, i.e., coloured marks, that a transition can move during its firing. When a transition fires, the tokens must take values that match the arc expressions, and they must belong to a type that match also the guard associated to the transition.

C.1 Colours Definition

The initial phase in the modelling of the product holon class, using CPN, is the definition of the colours related to the product behaviour. During the holon life-cycle, these colours will be managed by the functions associated with the structure of the holons present in the holonic control system.

The basic colours defined in this work for the product holon model are:

- $PH = \{ph_1, ph_2, ..., ph_n\}$, which are all possible product holons in the system. In case of 1000000 product holons present in the system, the n value is related to 1000000;
- $Or = \{or_1, or_2, ..., or_m\}$, which are all possible production orders in the system;
- $WP = \{wp_1, wp_2, ..., wp_l\}$, which are all possible work plans in the system;
- $RMP = \{rmp_1, rmp_2, ..., rmp_q\}$, which are all possible raw materials and parts in the system;
- $TH = \{th_1, th_2, ..., th_p\}$, which are all possible task holons in the system.

Some complex colours are also defined from the product of two or more basic colour domains:

- $\Omega_{12} = PH \times Or =< ph_i, or_j >$, which represents a production order or_j for the product represented by the product holon ph_i (since each product holon can have several instances running at the factory plant).
- $\Omega_{123} = PH \times Or \times WP =< ph_i, or_j, wp_z >$, which represents the work plan wp_z defined to the execution of the production order or_j associated to the product holon ph_i.
- $\Omega_{1234} = PH \times Or \times WP \times RMP =< ph_i, or_j, wp_z, rmp_h >$, which represents the raw materials or parts that will be used in the execution of the product order or_j associated to the product holon ph_i, according the work plan wp_z.
- $\Omega_{12345} = PH \times Or \times WP \times RMP \times TH =< ph_i, or_j, wp_z, rmp_h, th_r >$, which is the universal colour in the model, i.e. the Cartesian product of all basic colour domains.

C.2 Functions Definition

In a CPN model, each input and output arc have associated a function that manipulates the colours components, by removing appropriated components from the input elements and putting appropriated components in the output elements. In the product holon model, it was defined the following functions:

- proj1.$\omega_k =< ph_i >$, is the projection function that filters the ph_i colour.
- proj2.$\omega_k =< or_j >$, is the projection function that filters the or_j colour, creating a new instance of a production order.
- proj3.$\omega_k =< wp_z >$, is the projection function that filters the wp_z colour.
- proj4.$\omega_k =< rmp_h >$, is the projection function that filters the rmp_h colour.
- proj5.$\omega_k =< th_r >$, is the projection function that filters the th_r colour.
- proj12.$\omega_k =< ph_i, or_j >$, filters composed sub-tuples in order to get refined information, in this case the pair product holon (i) and production order (j).
- proj123.$\omega_k =< ph_i, or_j, wp_z >$, filters ω_k, returning the pair product holon (i), production order (j) and work plan (z).
- proj1235.$\omega_k =< ph_i, or_j, wp_z, th_r >$, filters ω_k, returning the pair product holon (i), production order (j), work plan (z) and task holon (r).
- succ$_{3y}$.$\omega_{ak} =< ph_i, or_j, wp_{z+y} >$, being wp_z one possible work plan of the set WP, the function succ$_{3y}$ selects the "y" successor of the wp_z, according to the $proj1$, $proj2$ and $proj3$ functions, i.e. that work plan that matches with the current product holon-production order pair.
- succ$_{4x}$. $< rmp_h >=< rmp_{h+x} >$, returns the x successor of the rmp component, i.e. the next raw material or part component in the product structure.

A transition that represents the selection of two distinct actions or the result of a decision for an component has so many firing modes as holons of these types were defined. Only by filtering it will be possible to know which action will be performed or which component is taking the action. As an example, to know if all raw material and parts, necessary to execute the product, are available, it is used a guard function that selects the flow of the tokens. The guard G=[Π rmp$_i$=1] has the True Boolean value if all instances of RMP colour domain are True, i.e. all raw material or parts are available to produce a product. Otherwise, this guard function is false, but the guard G=[Π rmp$_i$=0] has the True Boolean value, re-directing the flow of the tokens to the need to request the missing raw materials or parts.

Analysing in more detail the product holon model, it is possible to verify that it contains recursivity. In the case that a sub-product is requested through the *request of missing raw material and parts* activity, this model is recursively invoked, implying a production order (belonging to the set $\mathcal{O}r$), and the sub-product will be considered by a product holon (belonging to the set $\mathcal{PH}$).

C.3 Formal Validation of the Product Holon CPN Model

The formal validation of the CPN model for the product holon can also be performed.

Table C.1: Example of Marking

Place	**Marking**
p1	$\{< ph_1 >, < ph_2 >, < ph_4 >, < ph_{(i-1)} >, < ph_{(i+1)} >, ..., < ph_n > \}$
p2	$\{< or_1 >, < or_3 >, ..., < or_{(j-1)} >, < or_{(j+1)} >, ..., < or_m >\}$
p3	
p4	$\{(\Sigma[i : 1 - l]wp_i) - wp_k - wp_s\}$
p5	$\{< ph_i, or_2, wp_k >\}$
p6	$\kappa.\{(\Sigma[i : 1 - q]rmp_i)\}$
p7	
p8	
p9	$\{(\Sigma[i : 1 - p]th_i) - th_p\}$
p10	$\{< ph_3, or_j, wp_s, th_p >\}$
p11	

Taking into consideration the marking example presented in Table C.1, which represents a state of the net, the following are some of the specifications that can be validated:

- There are two products with their corresponding production orders that are being currently processed:
  - The product holon ph_i, which is in charge of the production order or_2, has already assigned/elaborated a work plan (wp_k) to the product and is ready to start the verification of the availability of parts and raw material that are necessary for the production process.

- The product holon ph_3, which is in charge of the production order or_j, has already finished the verification of available parts and raw material and it has also synchronised its actions with a task holon (th_p). The last is ready to start with the control of the execution of the production order using the work plan wp_s.

- The system is ready to initiate the process of $(n-2)$ new products, to process $(m-2)$ new orders, to assign $(l-2)$ work plans, and to synchronise the activities with $(q-1)$ task holons.

The characteristics addressed above and other structural and behavioural specifications of the modelled system can be formal validated using the information contained in the incidence matrix of Figure C.2.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
p_1	-proj1							proj1
p_2	-proj2							proj2
p_3	proj12	-proj12						
p_4		-proj3						proj3
p_5		proj123. succ_{3y}	-proj123					
p_6			-proj4					proj4
p_7			proj123 +proj4	-(proj123 +proj4)	-(proj123 +proj4)	proj123+proj4 +proj4.succ_{4x}		
p_8					proj4	-proj4		
p_9				-proj5				proj5
p_{10}				proj1235			-proj1235	
p_{11}							proj1235	-proj1235

Figure C.2: Incidence Matrix

Applying well-known concepts from the functional analysis and linear algebra, the matrix of Figure C.2 will be diagonalised. The procedure that conduces to the diagonalisation of the incidence matrix allows obtaining the set of right- and left-anulators, which are related to sub-sets of places and sub-sets of transitions (and their firing modes) of the CPN model. In the literature, these sub-sets are called place and transition flows (please consult the references [Colombo et al., 2002, Haddad and Couvreur, 1988, Jensen, 1992] and the references therein to know more details about the subject).

Table C.2: Place-flows of Incidence Matrix

	Place-flow 1	**Place-flow 2**
p1	1	0
p2	0	0
p3	proj1	0
p4	0	0
p5	proj1	0
p6	0	0
p7	proj1	0
p8	proj1	0
p9	0	1
p10	proj1	proj5
p11	proj1	proj5

As an example, two place-flows of the incidence matrix are shown in Table C.2. These sets of place-flows generate a set of place-invariants (PI). For example:

PI1 : $M(p_1) + < proj1 > .[M(p_3) + M(p_5) + M(p_7) + M(p_8) + M(p_{10}) + M(p_{11})] =$
$= M_0(p_1) + < proj1 > .[M_0(p_3) + M_0(p_5) + M_0(p_7) + M_0(p_8) + M_0(p_{10}) +$
$+M_0(p_{11})] = \sum_{i=1}^{n} < ph_i >$

PI2 : $M(p_9) + < proj5 > .[M(p_{10}) + M(p_{11})] = M_0(p_9) + < proj5 > .[M_0(p_{10}) +$
$+M_0(p_{11})] = \sum_{i=1}^{l} < th_i >$

The processing of these invariants allows the validation of different specifications, for example those illustrated in the following propositions.

Proposition C.1 *At any one time, the maximum number of product holons that can be set into operation by ADACOR is limited to n, i.e., the initial marking of place* p_1.

Proof: From PI1, if this is multiplied by $< ph_i >$, then $\forall \omega * =< ph_i, or_j, wp_l, rmp_k, th_s > \in \Omega *$

$$< ph_i > .\{M(p_1) + < proj1 > .[M(p_3) + M(p_5) + M(p_7) + M(p_8) + M(p_{10}) + M(p_{11})]\} =$$
$$=< ph_i > .\{M_0(p_1) + < proj1 > .[M_0(p3) + M_0(p_5) + M_0(p_7) + M_0(p_8) + M_0(p_{10}) +$$
$$+M_0(p_{11})]\} =< ph_i > .\sum_{i=1}^{n} < ph_i > = n$$

Thus, the maximum number of product holons in the system is n. With this equality it is possible to conclude that the number of product holons in activity is

$$< ph_i > . < proj1 > .[M(p_3) + M(p_5) + M(p_7) + M(p_8) + M(p_{10}) + M(p_{11})] \leq n$$

On the other hand, the number of product holons idle is

$$< phi > .M(p_1) \leq n$$

Note that the marking of place p_1 represents products waiting to start their execution. The proof is straightforwardly obtained from both un-equalities.

Corollary C.1 *At any one time, a product holon can only be assigned to the process of a product.*

For the exemplary state of the CPN model for the product holon, the application of the results of proposition 4.1 when the product holon $< ph_3 >$ is assigned, renders the following main conclusion:

$$\begin{aligned} & < ph_3 > . < proj1 > .[M(p_3) + M(p_5) + M(p_7) + M(p_8) + M(p_{10}) + M(p_{11})] = \\ = \; & < ph_3 > . < proj1 > .[0+ < ph_i, or_2, wp_k > +k.\sum_{i=1}^{q} rmp_i + 0+ < ph_3, or_j, wp_s, th_p > +0] \\ = \; & < ph_3 > .[0+ < ph_i > +0 + 0+ < ph_3 > +0] = 1 \end{aligned}$$

and

$$\begin{aligned} & < ph_3 > .M(p_1) =< ph_3 > .\{< ph_1 >, < ph_2 >, ..., < ph_4 >, < ph_{(i-1)} >, \\ & < ph_{(i+1)} >, ..., < ph_n >\} = 0 \end{aligned}$$

This means, that at any one time, a product holon can only be assigned to the process of a product.

Corollary C.2 *A product holon can be assigned to produce at any time different production orders.*

A colour tone can be present with a given multiplicity, i.e., $u. < ph_i >$. This means that additionally, a product holon can at any time handle the execution of several different production orders, in the maximum of m (cardinality of the set $\mathcal{O}$r). For example, a production order $or(2)$ to produce the product A with due date D1, ..., and a production order $or(12)$ to produce the product A with due date D4.

Proposition C.2 *At any one time, the maximum number of task holons that can be set into operation by ADACOR is limited to p, i.e., the initial marking of place* p_9*. This value is equal to the sum of all production orders launched to the shop floor to be executed (those in places* p_{10} *and* p_{11}*), plus the set of task holons that are already idle.*

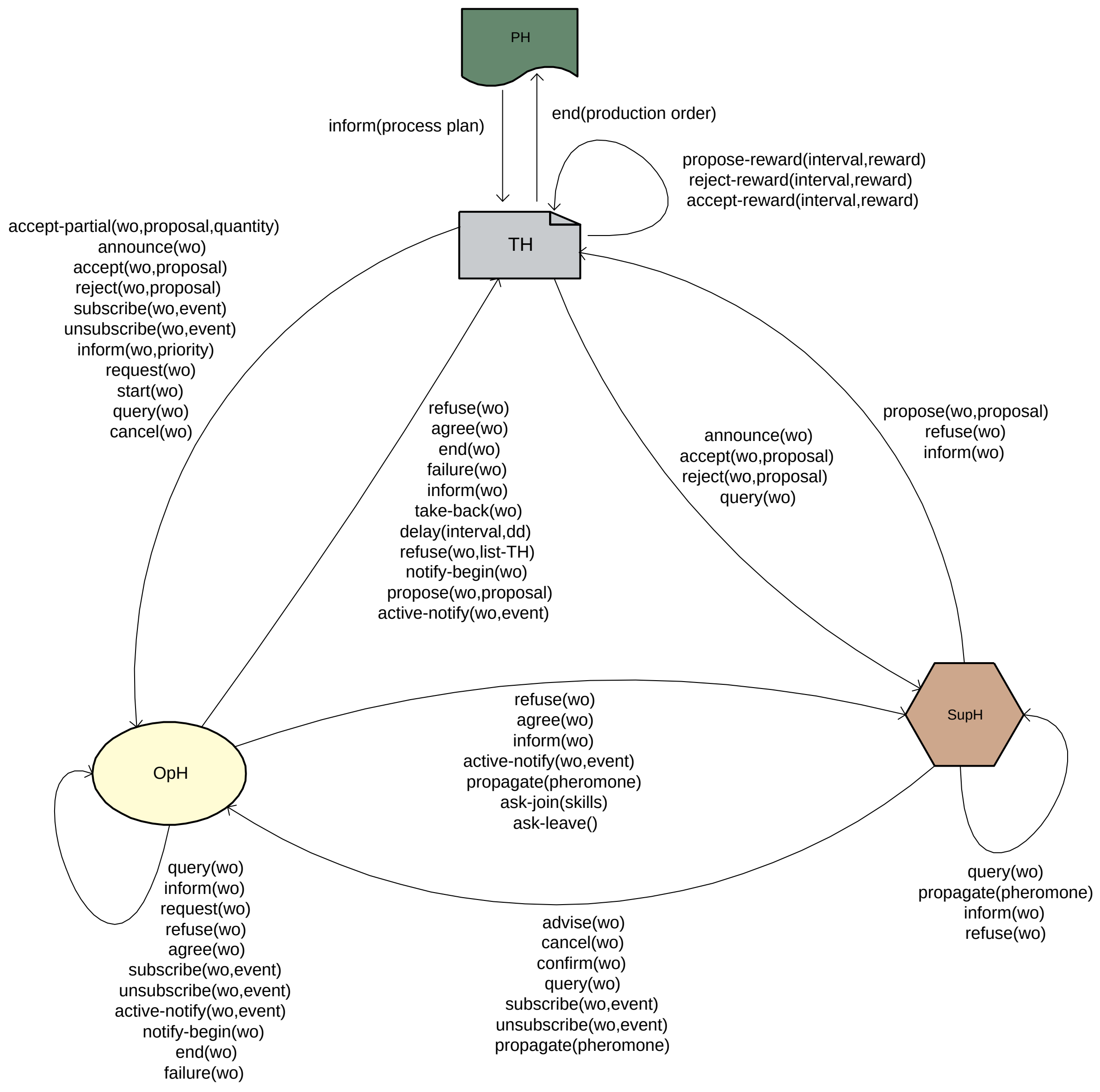
Proof: Immediate from the structure of place-invariant relationship PI2.

Corollary C.3 *The mutual exclusion relationship between the markings of* p_9*,* p_{10} *and* p_{11}*, is closed related to the firing sequence associated to the set of transitions addressed above.*

Transition t_4 models activities related to the decision if all raw materials and parts are physically available to produce a product. The transition t_7 models the production order execution. This transition initially launches the task holon (or starts or gives the indication to the task holon), passing the production order information and a set of process plans. After that, the control is passed to the task holon (firing the colour of the transition t_7), which is responsible for the execution of the production order. When the production order is finished, the task holon returns the control to the product holon passing back information about the production order execution, by means of the firing of transition t_8.

Appendix D

ADACOR Architecture Messages


PH
inform(process plan)
end(production order)
propose-reward(interval,reward)
reject-reward(interval,reward)
accept-reward(interval,reward)
TH
accept-partial(wo,proposal,quantity)
announce(wo)
accept(wo,proposal)
reject(wo,proposal)
subscribe(wo,event)
unsubscribe(wo,event)
inform(wo,priority)
request(wo)
start(wo)
query(wo)
cancel(wo)
refuse(wo)
agree(wo)
end(wo)
failure(wo)
inform(wo)
take-back(wo)
delay(interval,dd)
refuse(wo,list-TH)
notify-begin(wo)
propose(wo,proposal)
active-notify(wo,event)
announce(wo)
accept(wo,proposal)
reject(wo,proposal)
query(wo)
propose(wo,proposal)
refuse(wo)
inform(wo)
SupH
OpH
refuse(wo)
agree(wo)
inform(wo)
active-notify(wo,event)
propagate(pheromone)
ask-join(skills)
ask-leave()
query(wo)
propagate(pheromone)
inform(wo)
refuse(wo)
query(wo)
inform(wo)
request(wo)
refuse(wo)
agree(wo)
subscribe(wo,event)
unsubscribe(wo,event)
active-notify(wo,event)
notify-begin(wo)
end(wo)
failure(wo)
advise(wo)
cancel(wo)
confirm(wo)
query(wo)
subscribe(wo,event)
unsubscribe(wo,event)
propagate(pheromone)