INNOVATIVE TECHNOLOGY SERIES

INFORMATION SYSTEMS AND NETWORKS

Intelligent Agent-based Operations Management

edited by

Sophie d'Amours & Alain Guinet

HPS HERMES PENTON SCIENCE

an imprint of KOGAN PAGE SCIENCE

Intelligent Agent-based Operations Management

This page intentionally left blank

INNOVATIVE TECHNOLOGY SERIES
INFORMATION SYSTEMS AND NETWORKS

Intelligent Agent-based Operations Management

edited by
Sophie d'Amours & Alain Guinet

KOGAN PAGE SCIENCE

London and Sterling, VA

First published in Great Britain and the United States in 2003 by Kogan Page Science, an imprint of Kogan Page Limited

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms and licences issued by the CLA. Enquiries concerning reproduction outside these terms should be sent to the publishers at the undermentioned addresses:

120 Pentonville Road
London N1 9JN
UK
www.koganpagescience.com

22883 Quicksilver Drive
Sterling VA 20166-2012
USA

© Hermes Science Publishing Limited, 2003
© Kogan Page Limited, 2003

The right of Sophie d'Amours and Alain Guinet to be identified as the editors of this work has been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

ISBN 1 9039 9643 0

British Library Cataloguing-in-Publication Data

A CIP record for this book is available from the British Library.

Library of Congress Cataloging-in-Publication Data

Intelligent agent-based operations management / edited by Sophie
d'Amours and Alain Guinet.
p. cm. -- (Innovative technology series)
ISBN 1-903996-43-0
1. Production management. 2. Intelligent agents (Computer software)
I. d'Amours, Sophie. II. Guinet, Alain. III. Innovative technology
series: information systems and networks
TS155.I5773 2003
658.5--dc21
2003003757

Typeset by Kogan Page
Printed and bound in Great Britain by Biddles Ltd, Guildford and King's Lynn
www.biddles.co.uk

Contents

Foreword
Sophie d'Amours and Alain Guinet vii

1. **A Procedure for Building Product Models**
Lars Hvam, Jesper Riis, Martin Malis and Benjamin Hansen 1

2. **Identification of Scheduling Problems: The DeSAP Interface within the e-OCEA Environment**
Claudine Tacquard and Franck Thibaut 27

3. **Product Generic Modelling for Configuration: Requirement Analysis and Modelling Elements**
Michel Aldanondo, Khaled Hadj-Hamou, Guillaume Moynard and Jacques Lamothe 49

4. **Production Management Systems**
Farid Ameziane and Stéphane Lasserre 71

5. **Agent-based Agile Manufacturing System Scheduling**
David He and Astghik Babayan 87

6. **New Product Development within a Concurrent Engineering Environment: Knowledge and Software Tools**
Jean-Louis Selves, Eric Sanchis and Zhaoyang Pan 109

7. **An IEC 61499-based Model for Reconfiguration of Real-time Distributed Control Systems**
R.W. Brennan, M. Fletcher and D.H. Norrie 127

8. **Intelligent Agents for Production Systems**
Pierre Massotte, Jihad Reaidy, Yingjiu Liu and Daniel Diep 147

Index 165

This page intentionally left blank

Foreword

The publication is dedicated to multi-agent systems for product, process and organisation modelling. Eight papers dealing with these topics are included.

Regarding product modelling, the paper "New product development within a concurrent engineering environment: knowledge and software tools" by Jean-Louis Selves, Eric Sanchis and Zhaoyang Pan, proposes a concurrent engineering approach instead of the traditional sequential approach in the framework of new products development. Their approach allows the project team to respond more quickly to changing market conditions and is supported by software tools based on software agents. Lars Hvam, Jesper Riis, Martin Malis and Benjamin Hansen emphasise the need to propose new approaches. Their article "A procedure for building product models" focuses on the opportunities for supporting the product specification process with new tools. Their idea is to formalise knowledge, information of the products and their life cycle properties, and to express the knowledge in intelligent systems. In "Product generic modelling for configuration: requirement analysis and modelling elements", Michel Aldanondo, Khaled Hadj-Hamou, Guillaume Moynard and Jacques Lamothe identify and classify configuration modelling requirements for customisable industrial products. It analyses how generic modelling and configuration assistance can fulfil the requirements. Regarding process modelling, in "Production management systems" by Farid Ameziane and Stéphane Lasserre, the authors analyse the contribution of Concurrent Engineering and Knowledge capitalisation in the process modelling of building construction industry. Their work focuses on information and knowledge management.

Regarding organisation modelling, four papers emphasise the contribution of multi-agent approaches. First, the paper "Intelligent agents for production systems" by Pierre Massotte, Jihad Reaidy, Yingjiu Liu and Daniel Diep, describes a new approach devoted to the management and control of distributed manufacturing systems and based on interactions between intelligent agents. These agents are able to perform automatic reconfigurations of a supply chain. In the same field, R.W. Brennan, M. Fletcher and D.H. Norrie describe a general approach for dynamic and intelligent reconfiguration of real-time distributed control systems, in their paper "An IEC 61499-based model for reconfiguration of real-time distributed control systems". Their approach takes advantage of multi-agent systems. Next, the paper "Identification of scheduling problems" by Claudine Tacquard and Franck Thibaut, proposes a decision support system to model flexible manufacturing systems and identify the associated scheduling problems. Appropriate techniques and tools are proposed to the user. Finally, in "Agent-based agile manufacturing system scheduling" by David He and Astghik Babayan, the contribution of agent based approaches to improve scheduling flexibility and robustness is studied. The authors propose a methodology for the development of negotiation mechanism between agents.

Sophie d'Amours
Alain Guinet

Chapter 1

A Procedure for Building Product Models

Lars Hvam, Jesper Riis, Martin Malis and Benjamin Hansen
Department of Manufacturing Engineering and Management, Technical University of Denmark, Denmark

1. Introduction

A product model supports the the *specification process* for products in sales, design and methods engineering. The specification process denotes the part of the engineering system in which the specifications for the customised product variants are created, as illustrated in Figure 1.

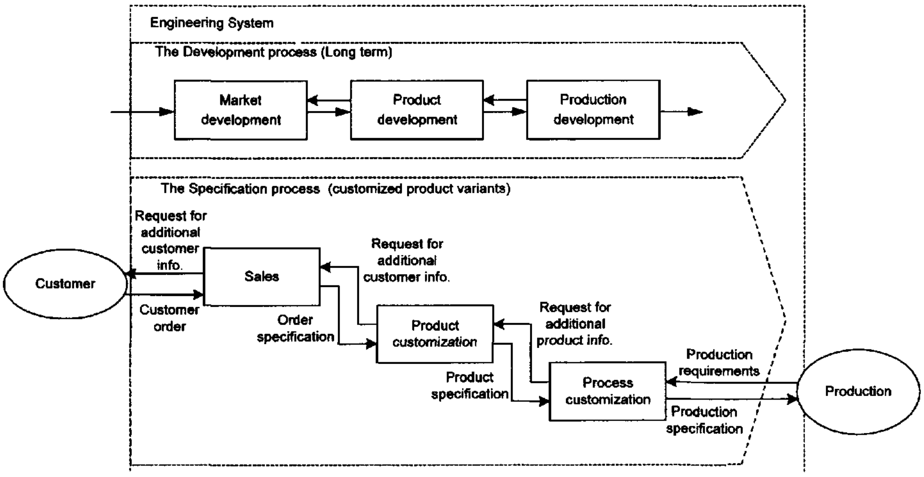


Figure 1. *The engineering system*

The activities in the specification process includes an analysis of the customer's needs, design and specification of a product which fulfill the customer's needs and specification of e.g. product manufacture, transportation, erection on site and service (specification of the product's life cycle properties). The activities in the specification process are characterised by having a relatively well-defined space of (maybe complex) solutions as a contrast to product development, which is a more creative process.

Typical goals for the specification process are the ability to find an optimal solution according to the customer's needs, high quality of the specifications, short lead time and a high productivity of the work carried out. Typical critical goals for the development process are to derive new original concepts of product design with improved functionality and life cycle properties, and to reduce time to market for the new product designs. The diversification of tasks and goals in the specification and development processes leads to a separation of the two processes as suggested in Figure 1 above.

This article focuses on the opportunities provided in supporting the specification process with IT. The idea is to formalise knowledge and information of the products and their life cycle properties and to express the knowledge in IT-systems – so-called *product models.*

Knowledge integrated product- and product related models are defined as:

"A knowledge base which contains part of or all of the knowledge and information associated with the product in different phases of the product's life cycle, e.g. sales, design, production, assembly, service and reuse", Hvam (2000b).

"Product related models contain knowledge and information about the systems related to the product's life cycle, while the product model contains knowledge and information about the product's structure and functional properties", Krause (1988).

Knowledge integrated or intelligent product models mean that the models contain knowledge and information about the products, and based on this are able to derive new specifications for product instances and their life cycle properties. The principle of using product models to support the specification process is to make the product knowledge of the engineer explicit to the rest of the organisation.

Product models implemented in IT-systems, such as sales configuration systems, have been applied in industry during the last 10 to 15 years for relatively simple products such as the configuration of computers and other electronic equipment.

There are today a number of examples of product and product related models which, for instance, are used to support sales, design of product variants and production preparation. Experiences from a considerable number of Danish companies shows that these models are often constructed without the use of a strict procedure or modelling techniques. The result is often that the systems are unstructured and undocumented, and they are therefore difficult or impossible to maintain or further develop. Consequently there is a need to develop a procedure and associated modelling techniques which can ensure that the constructed product and product related models are properly structured and documented.

Furthermore, experience shows that the product and product related models are not always designed to fit the business processes that they are meant to support. Finally, an important precondition for building product models is that the products are designed and structured in a way that makes it possible to define a general master of the product, from which the customer-specific products can be derived.

In order to cope with these challenges, a procedure for building product models should include: An analysis and redesign of the specification processes in focus, an analysis and eventually redesign/ restructuring of the products to be modelled, and finally, a structured "language" - or modelling technique - which makes it possible to document the product and product related models in a structured way.

2. A procedure for building product models

Figure 2 shows a procedure for building product and product related models. The procedure contains 7 phases. The starting point for the work is an analysis and redesign of the business processes, which will be affected by the product and product related models (phase 1). In phase 2 the products are analysed and described in a so-called *product master*. Phase 3 includes the final design of the product and product related models using object oriented modelling techniques. Phases 4 to 7 deals with design, programming, implementation and maintenance of the product models. Phases 3 to 7 follow by and large the general object oriented project life cycle.

There may be some overlap and iterations between the individual phases.

Phase	Description
1	**Process Analysis** Analysis of the existing specification process (AS-IS), statement of the functional requirements of the process. Design of the future specification process (TO BE). Overall definition of the product – and product related models to support the process. **Tools:** IDEF0, flow charts, Activity Chain, Model, key numbers, problem matrix, list of functional describing characteristics and gap analysis.
2	**Product Analysis** Analysing products and eventually life cycle systems. Redesigning/ restructuring of products. Structuring and formalising knowledge about the products and related life cycle systems in a product master. **Tools:** List of features and product master.
3	**Object Oriented Analysis (OOA)** Creation of object classes and structures. Description of object classes on CRC-cards. Definition of user interface. Other requirements to the IT solution. **Tools:** Use cases, screen layouts, class diagrams and CRC-cards.
4	**Object Oriented Design** Defining and further developing the OOA-model for a specific programming tool.
5	**Programming** Programming the system. Own development or use of a standard software.
6	**Implementation** Implementation of the product- and product related models in the organisation. Training users of the system, and further training of the people responsible for maintaining the product- and product related models
7	**Maintenance** Maintenance and further development of the product and product related models.

Figure 2. *A procedure for building product models. The contents of the phases are described in the following sections*

2.1. *Phase 1 – Process Analysis*

Initially an AS-IS description of the current processes is made. This description should expose the structure relating activities, people, IT systems, shifts of responsibility etc. Key figures for characterising quality, resource consumption and throughput times may support the description. Analysis tools to be used may be IDEF0 [ICAM, 1981], The Activity Chain Model [Barfod, 1997], or different kinds of flow charts.

In the other part of the process analysis of future requirements set by the surroundings are analysed and determined making it possible to evaluate the gap between the current process performance and the performance required. To support the requirement analysis a list consisting of functional characteristics is used. These are among others:

- Kinds of input and output specifications
- Throughput time
- Resource consumption
- Quality of specifications
- Insight into consequences
- Flexibility of the process
- Frequency of similar specification activities
- Accessibility of knowledge
- etc.

The functional characteristics are further outlined in [Hvam and Hansen, 1999 and Hvam et al 2000]. Based upon the functional requirements of the specification process and the characteristics of the existing specification process, a gap analysis is made in order to identify the major gap between the current performance of the business process and the requirements to the process. This leads to identify the potential improvements to realise using product- and product related models.

Based on the process analysis a concept for a future ideal business process is designed. This ideal concept is made in order to be more creative and not so restricted by "historic" procedures in the company, similar to the BPR approach [Hammer, 1990]. With the ideal concept in mind a more realistic process design (TO-BE) is made. This TO-BE description will consist of structural elements such as:

- Process structure
- Humans

- Organisation
- IT-systems

In relation to the definition of the future specification process, the product- and product related models to support the specification process are defined overall. Finally the purpose, view and context of the models to be build are defined.

2.2. *Phase 2 – Product Analysis*

In this phase the product to be modelled is analysed in order to gain an overview of the product families and their structure. The analysis covers the product's function and structure, the properties of the product, the variations of the product and the related systems in the product's life cycle. Figure 3 shows a general architecture for describing products including the above mentioned.

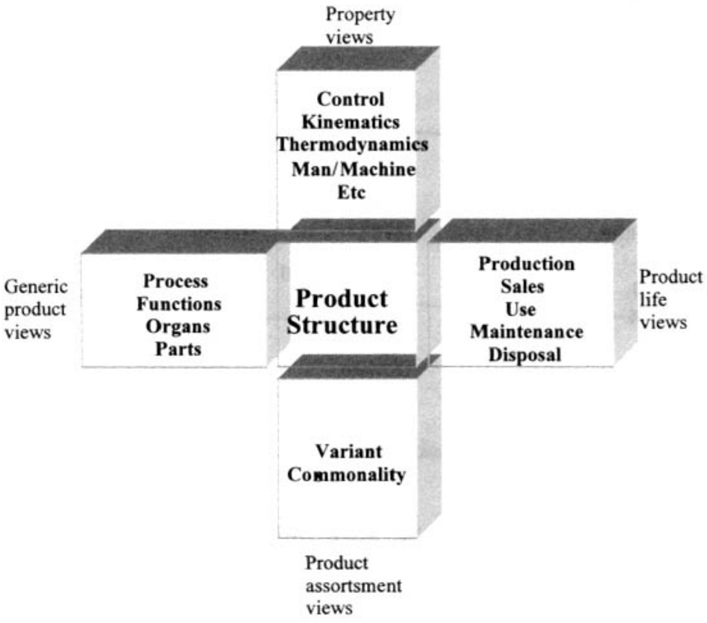


Figure 3. *An architecture for describing products [Hubka, 1988], [Mortensen, 2000]*

Normally, product and product related models contain only a minor part of the proposed views in the architecture. The specific views to include in the models are defined based on the overall content of the product and product related models outlined in phase 1.

Before the object-oriented analysis (OOA) is carried out, an overview over the product assortment and other views is set up by use of a so-called product master. Figure 4 below shows a part of a product master for a bookcase.

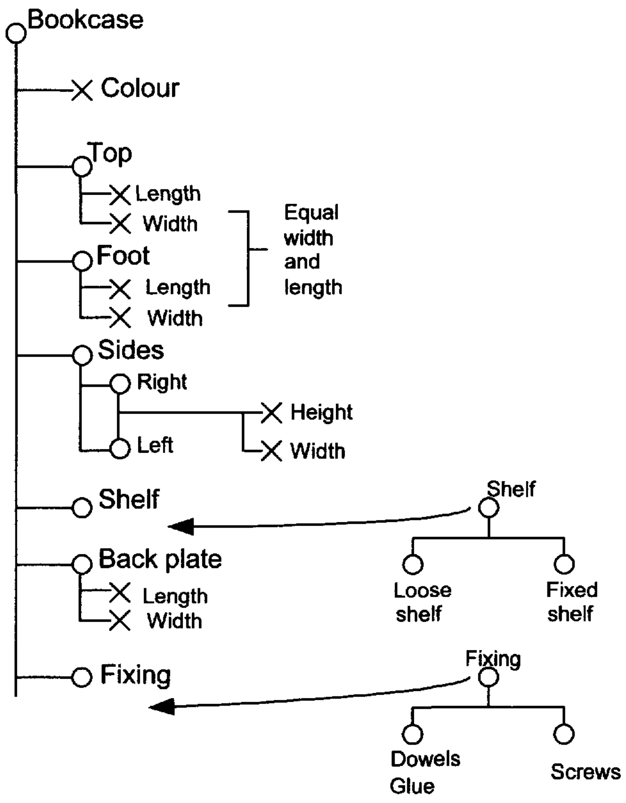


Figure 4. *Product master for a bookcase*

The product master is build up from *part-of* structures (corresponding to aggregation in OOA) and *kind-of* structures (corresponding to generalisation in OOA). A circle indicates a part in the bookcase, while a cross indicates a characteristic ('attribute' in object oriented modelling language). Relations/ constraints between the parts, or the characteristics of the parts, are indicated by a line between the two parts/ characteristics, and the relation is described.

2.3. *Phase 3 – Object Oriented Analysis (OOA)*

OOA is a method used for analysing a given problem domain and the field of application in which the IT system will be used. The purpose of OOA is to analyse the problem domain and the field of application in such a way that relevant knowledge can be modelled in an IT system. The problem domain is the part of reality outside the system that needs to be administrated, surveyed or controlled. The field of application is the organisation (person, department) that is going to use the system to administer, survey or control the problem domain.

2.3.1. *Modelling the problem domain*

The OOA model can be created through the activities described in Figure 5, which describes the OOA as consisting of five phases or layers. These layers can be seen as different viewpoints, which together make up the OOA model. Normally, the five activities are carried out through a top down approach, but there are no restrictions in that sense. Typically, the OOA model will be the result of a number of iterations of the analysis process.

The *subject layer* contains a sub-division of the complete domain which is to be modelled in different subject areas. In relation to the use of product models, a subject area can for example be a product model or a factory model [Krause, 1988].

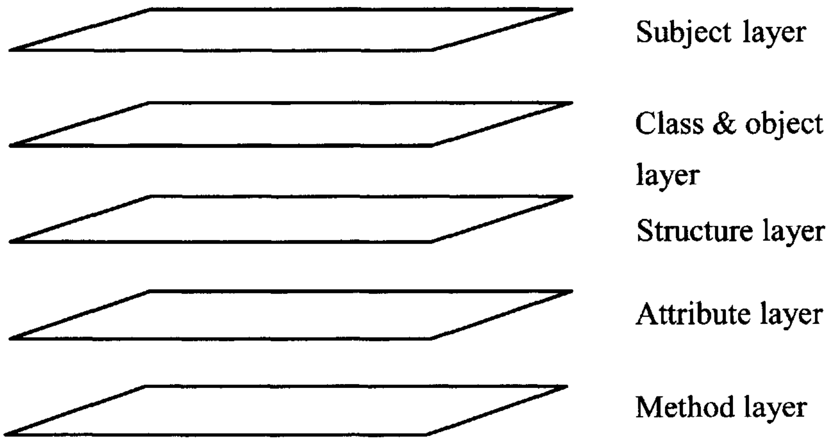


Figure 5. *The five layers of OOA modelling [Coad, 1991]*

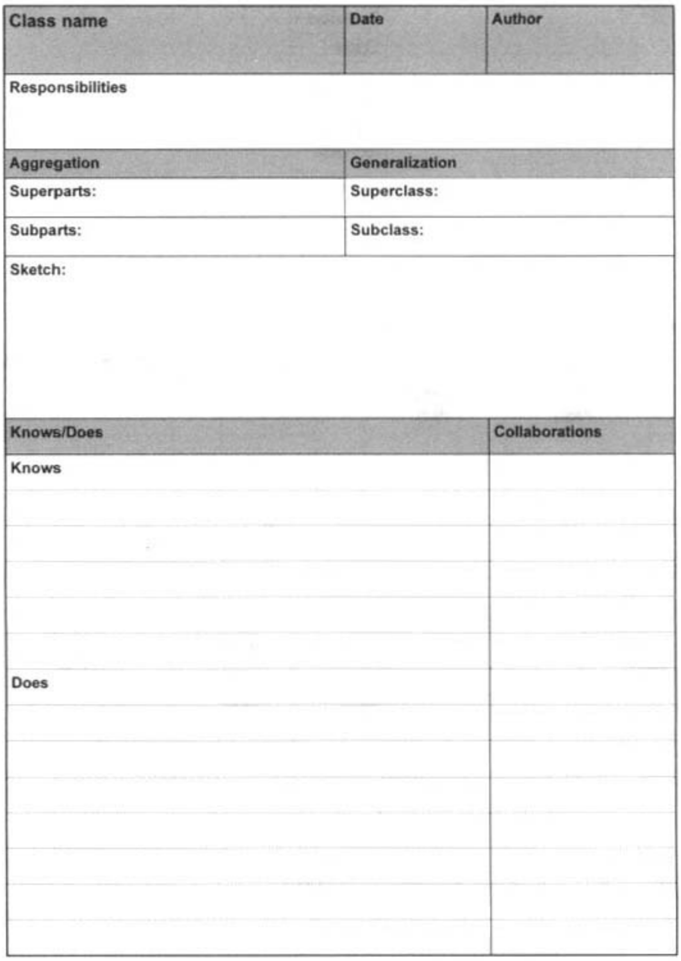
The *class and object layer* contains a list of the classes and objects which have been identified in the individual subject areas.

The *structure layer* contains the relationships between the objects, i.e. a specification of generalisation and aggregation.

The *attribute layer* contains a specification of the information associated with the individual objects, i.e what the objects know about themselves.

The *method layer* contains a description of the individual objects methods (procedures), i.e. what the objects can perform.

Classes and structures are identified based on the product master from phase 2. The static structure is mirrored in the layers of theme, classes and objects, structure and attributes, while the more dynamic aspects in the model are mainly related to the method layer. The result of the OOA can be illustrated in a class diagram [Booch et al, 1999; Bennett et al., 1999] and on CRC cards as shown in Figure 6 below. A CRC card describes the details of the classes and contain a description of the mission of the class, the attributes and methods (constraints), the relations to other classes and often also a sketch of the product/ part in focus [Bellin et al., 1997; Hvam and Riis, 1999].



Class name	Date	Author
Responsibilities		
Aggregation	Generalization	
Superparts:	Superclass:	
Subparts:	Subclass:	
Sketch:		
Knows/Does		Collaborations
Knows		
Does		

Figure 6. *CRC card for product modelling*

The notation used in the class diagram is illustrated in Figure 7, which shows a class diagram for the bookcase based on the product master shown in Figure 4. The notation is a part of the *unified modelling language* (UML), which has been chosen since it is the preferred standard world wide and is used in many development tools.

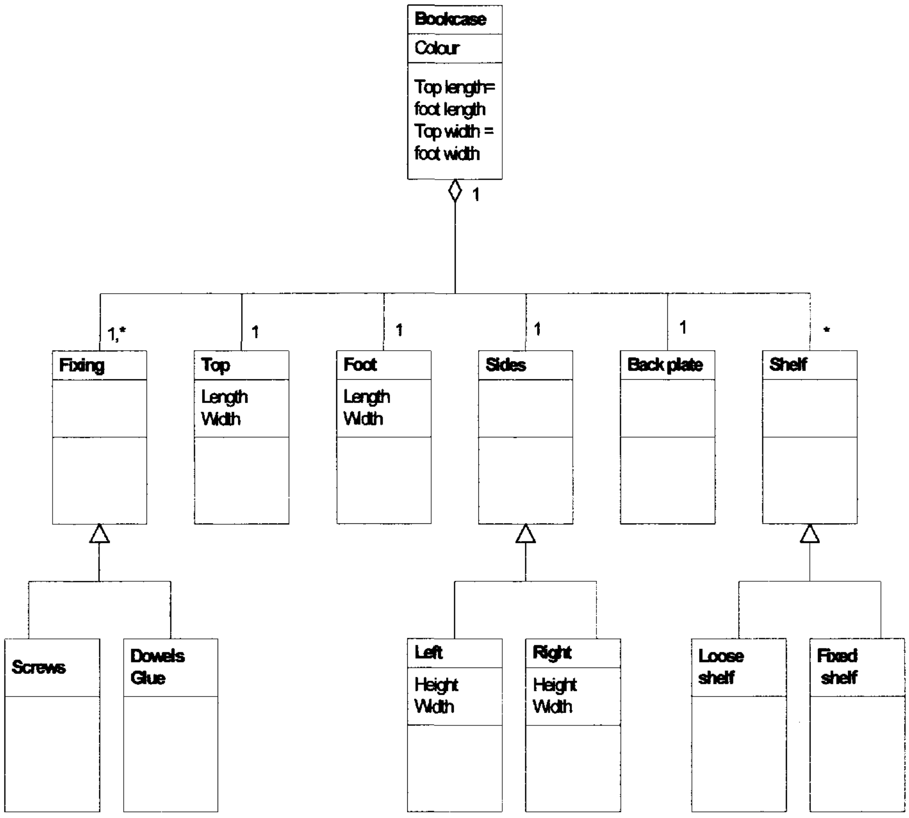


Figure 7. *Class diagram for a bookcase (UML)*

2.3.2. *Modelling the field of application*

The second part of the OOA consists of an analysis of the field of application. Here the interaction between man and machine is analysed in order to determine the functionality of the system, the user interface, software integration to other IT-systems etc. Other elements that need to be determined are also requirements for response time, flexibility and so on. The result of this is a description of the user interface and a requirement specification for the product and product related models.

2.4. *Phase 4 – Object-Oriented Design*

When a system is being built up, the perspective changes from being domain oriented (what and which task?) to being implementation oriented (how?).

According to [Coad, 1991] four perspectives are used during development of the OOD model:

- User interface, which determines the user's communication with the system.
- Problem domain, in which the OOA model is corrected in accordance with design-specific criteria.
- Data management, where the structure of the stored data and methods for control of data are modelled.
- Task management, which is used in cases where the system has to perform several tasks simultaneously (multitasking).

Object-oriented design contributes, like the other phases of the object-oriented project life cycle, to a structured procedure, and thus makes it possible to control the entire project more closely.

If a standard configuration tool is used (Baan Configurator, Oracle SellingPoint, etc.) most of the design parameters are frozen.

2.5. *Phase 5 – Programming*

The selection between a standard software system or own development depends on the company's resources such as economy, programming skills, current IT systems etc. If the company decides to develop its own system, object oriented programming languages such as C++, Java, or Smalltalk can be used.

In the last few years a large amount of work has been done to create various standard configuration solutions. The major suppliers of ERP and front office systems are now joining the market. The list below illustrates the more famous actors in the market:

Front Office/ERP:	**Front Office:**
Baan (now Invensys CRM), Oracle, Cincom, Sap, Intentia, i2 Technologies, J. D. Edwards, Peoplesoft etc.	Siebel, Trilogy, Calico, Firepond, Selectia, BT Squared, Clarify, SalesLogix etc.

A standard system makes the domain experts more independent of software programmers since the actual programming is relatively simple and can be done without extensive programming skills. However the integration of a standard system with other systems would normally necessitate the work from programmers.

Use of a standard system could provide advantages such as: easier development and maintenance, supplier support, safe IT costs, higher system- and education quality and the possibility of a test period before purchasing. On the other hand a standard system can be quite expensive and may lead to some disadvantages such as supplier dependence, integration/fitting difficulties, and changed work terms.

2.6. *Phase 6 – Implementation*

Implementation, user acceptance, maintenance and follow-up are very critical factors. The system must stand trial here. User acceptance is completely crucial if the system is to be a success; if the users are not satisfied, the system will not survive long. One way of getting the users' acceptance of the system is to involve the users already in the analysis phase. This can be done by developing an early prototype of the system, which the users can comment on. Also training and current information of the users will facilitate the users' acceptance of the system.

2.7. *Phase 7 – Maintenance*

Product and product related models can be viewed as "living organisms". The models will soon lose their value if they are not further developed and maintained. The object oriented structure and documentation (class diagrams, CRC cards etc.) of the product and product related models makes it considerably easier to maintain and develop the models further.

The application of product modelling introduces a new way of doing business. New tasks are introduced in the organisation. E.g. a salesman will have to use the product and product related models in order to configure a product, and a product designer will have to build up and maintain the information and rules describing the products. This calls for commitment and ongoing motivation from top management. Besides this both users and model managers need education and training in using and maintaining the product and product related models.

3. Empirical study

The proposed procedure has been tested at a Danish wind turbine manufacturing company, NEG Micon A/S, from August 1999 to March 2000 [Mertz and Vølund, 2000]. The aim of the project at NEG-Micon was to construct a product model (sales configurator) for the sales process. The work is described in the following.

3.1. *Case Study Phase 1 – Process Analysis*

A model of the sales- and specification process is shown in Figure 8. It illustrates the main activities from customer request to final documentation.

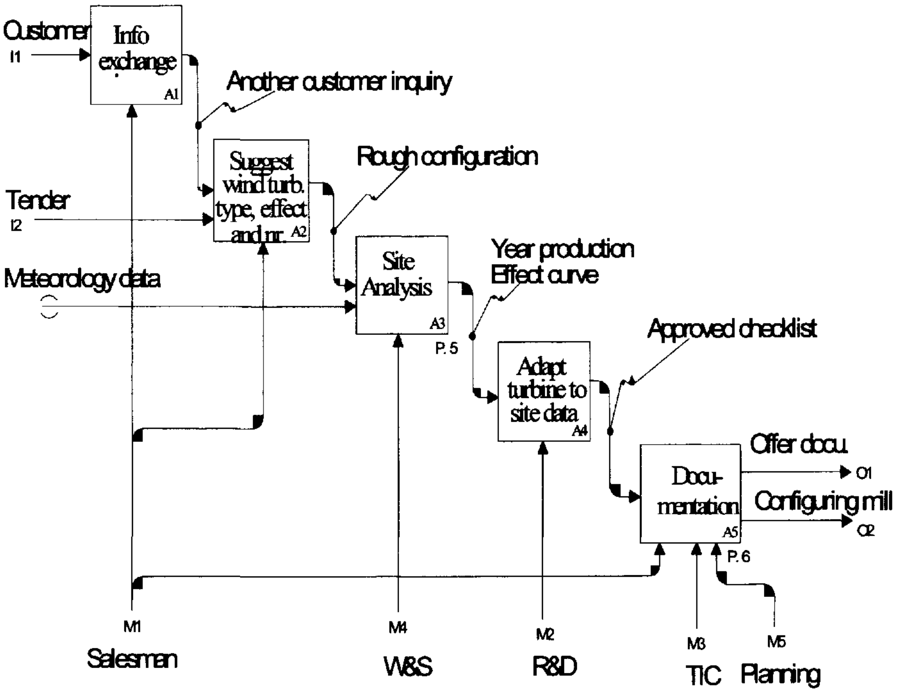


Figure 8. *IDEF0 diagram of the sales and specification process*

The analysis showed several weaknesses. It was for example found that approx. 55% of the specifications lacked quality. Based on the AS-IS description and the functional requirements gaps have been found in several areas:

- Quality of the specifications
- Resources spent for making the specifications
- Knowledge sharing between Product Design and Sales/ Manufacturing
- Lead-time for making the specifications

Furthermore the process should be able to:

- Handle a larger number of variants

- Handle continuous product changes in a better way
- Include approvals and regulatory requirements

Based on the above analysis of the requirements of the business process a future business process was set up. Based on this a simple diagram of an ideal configuration system was sketched and discussed with the representatives of the company. Three limitations (functionality, price of the system and time of development) led to the adjusted model. Business processes, which were difficult to support with product and product related models were sorted out. The product model (The Wind Turbine Conguration System) and related systems are illustrated in Figure 9.

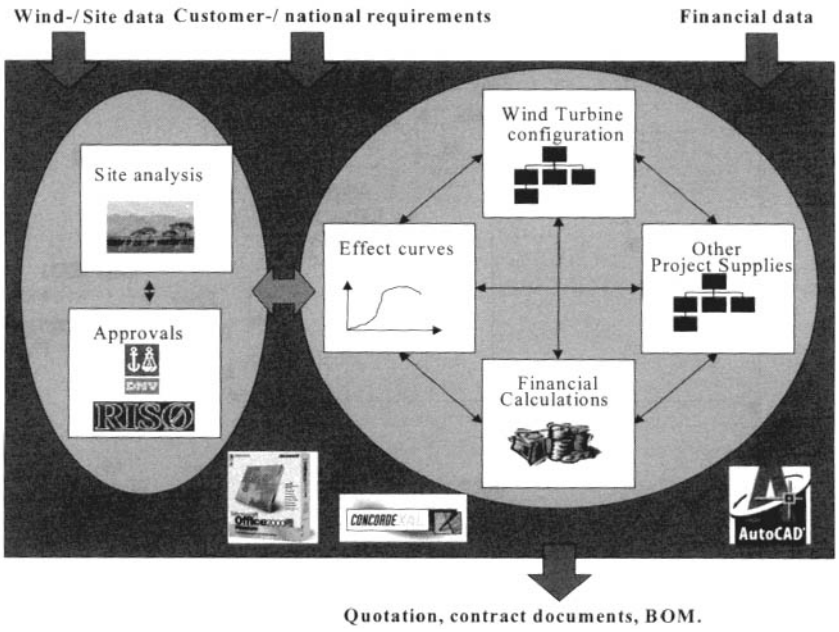


Figure 9. *The Wind Turbine Configuration System and its relation to other systems*

As a description tool for defining the interaction between the individual parts of the product and product related models and the people working in the specification process a Use Case has been made. The Use Case (Figure 10) shows the interactions between the employees at NEG-Micon involved in the specification process and the proposed product and product related models.

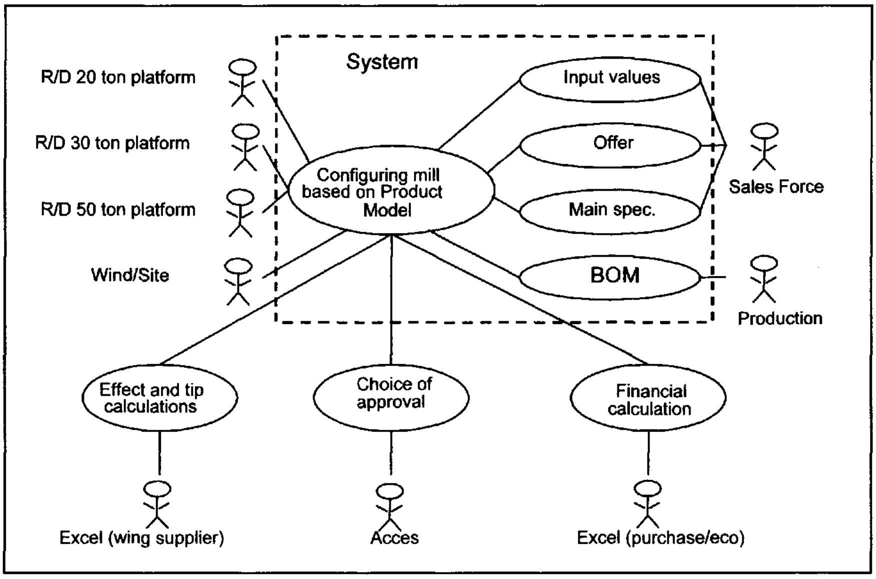


Figure 10. *Use Case at NEG Micon*

Based on the above listed analysis the purpose, view and context of the product and product related models have been defined as:

Purpose: The purpose is to set up a system, which is able to specify bills of materials at an overall level for 80% of the windmills sold.

View: The view is primarily the view of the sales personnel, secondary the view of the product designers who will be responsible for maintaining the rules in the system.

Context: The context of the models is specification of bills of materials at an overall level for windmills based on the 20 tons platform.

Standard software (Baan Configuration version 98.2) was chosen for the programming.

3.2. *Case Study Phase 2 – Product Analysis*

Figure 11 shows a part of the windmill (Nacelle and rotor).

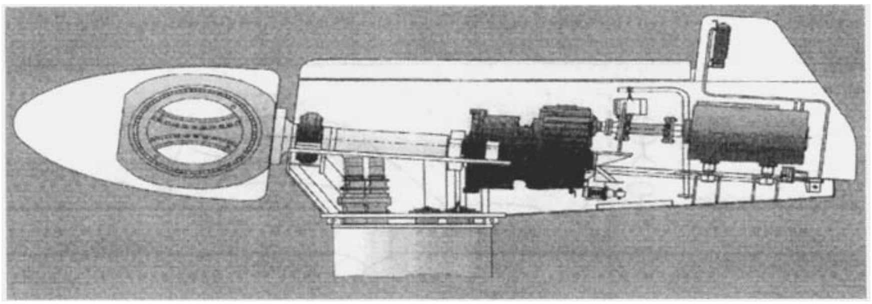


Figure 11. *Nacelle and rotor*

In order to get an overview of the windmill the product was analysed and a product master was built up. The overall content and the details to include in the product master is decided, based on the analysis of the business processes in phase 1. Figure 12 illustrates a part of the product master for windmills, where different parts of the wind turbine are described.

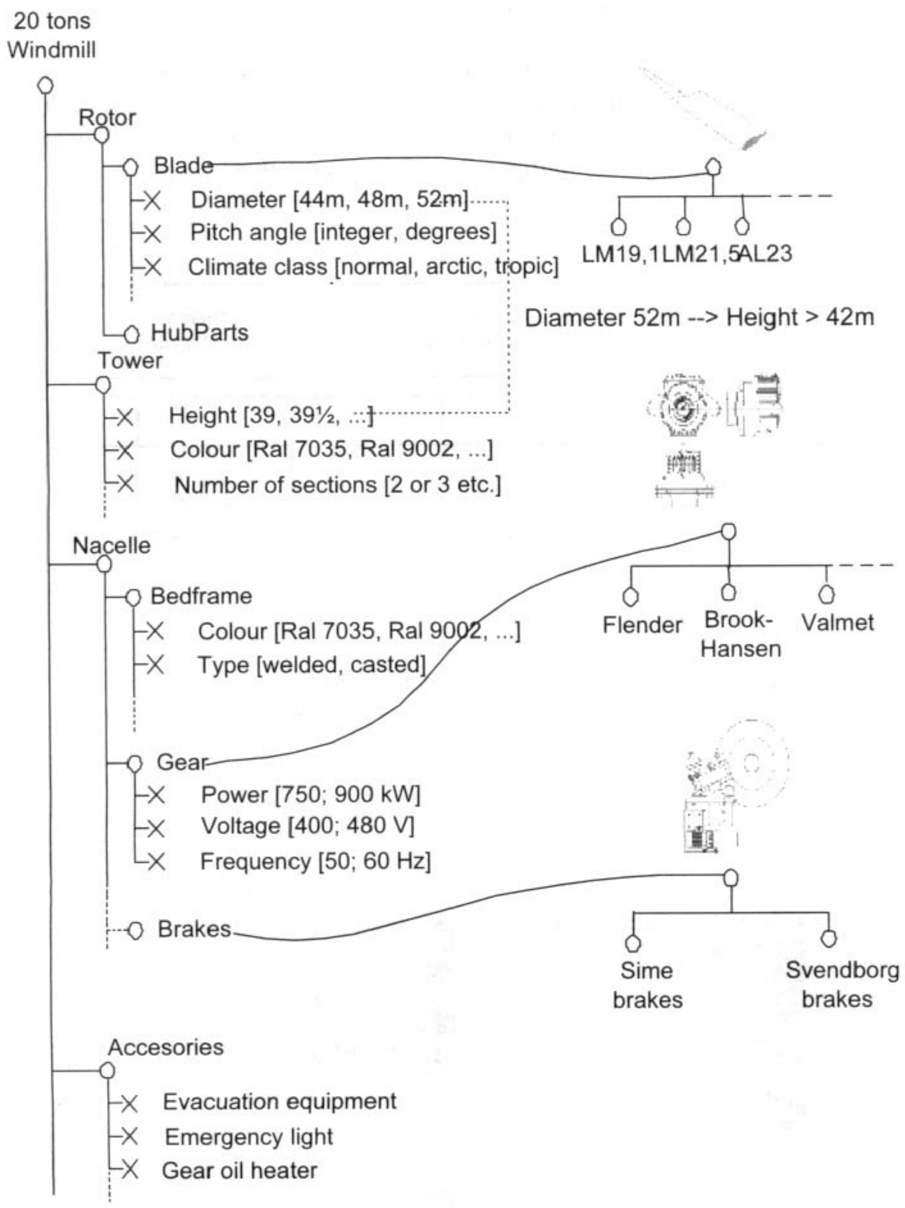


Figure 12. *A part of the product master for a windmill*

3.3. *Case Study Phase 3 – Object-Oriented Analysis model*

The OOA model is divided into five themes (generating documents, project information, configuration of the wind turbine, wind data and calculations). Each theme has classes connected. They are divided into attributes and methods. Figure 13 shows the contents of theme 3: Configuration of the wind turbine.

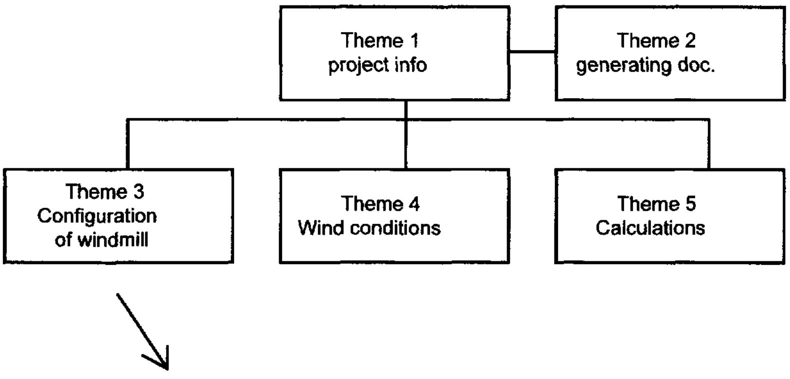
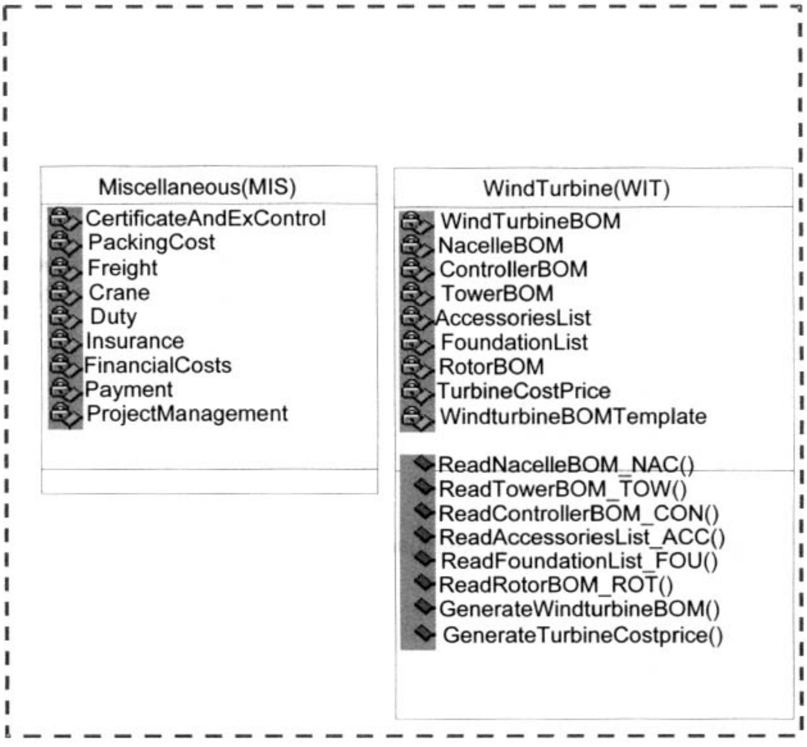


Figure 13. *Theme 3: Configuration of the turbine. Rational Rose*



3.4. *Case Study Phase 4 – Object-Oriented Design*

In this case a standard configuration system (Baan Configurator) was used. As with most standard systems, the design is frozen by the supplier.

3.5. *Case Study Phase 5 – Programming*

When programming a standard system the concepts for programming are set by the supplier. The Baan Configurator is logic and constraint based. The product attributes and constraints are programmed based on the attributes and methods listed on the CRC-cards from the OOA-model.

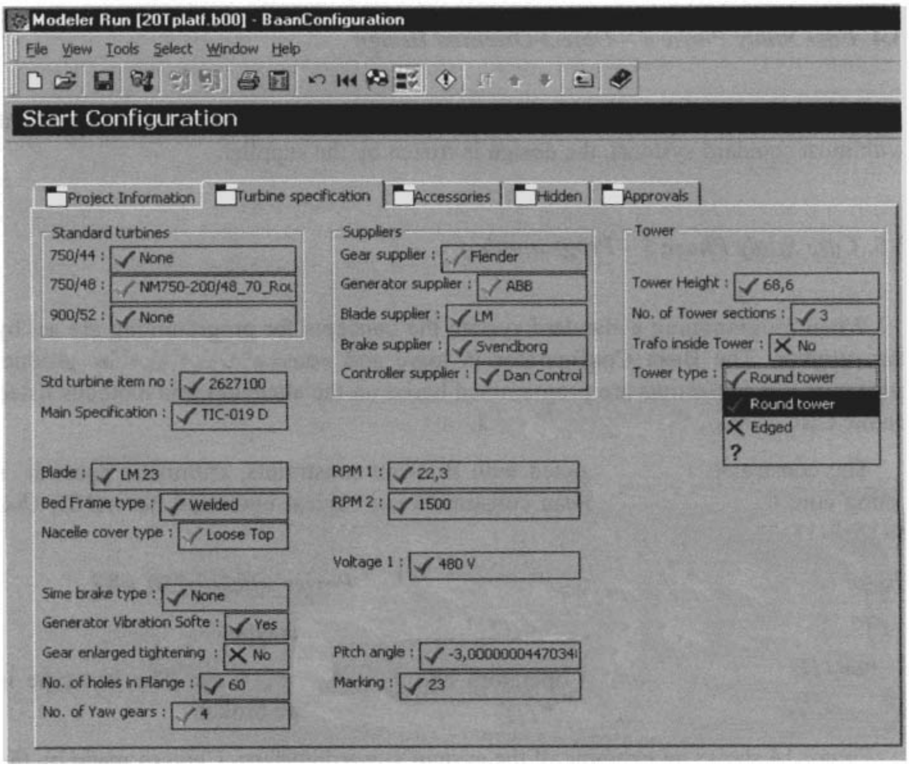
The constraints are constructed with Boolean constraints, arithmetic constraints and warning constraints. Boolean constraints use logical operators like AND, OR, NOT, TRUE, FALSE etc. For example:

Rotor_diameter[a44]OR Rotor_diameter[a48] <>Power_effekt[a750_kW].

The main part of the constraints in the system is of this type.

Arithmetic constraints use operators like +, -, *, /, <, > etc. The last type is warning constraints that give a message if some criteria are broken.

Figure 14 shows an example of the system's user interface. Choices made by the user are marked by a check mark. If the selection is against the rules, then the configurator will give an automatic warning and a note that indicates where the selections must be corrected/changed. The user is able to start the configuration at a random place and it is possible to optimise according to different resources (price, output etc.).



Standard turbines
750/44 : ✓ None
750/48 : ✓ NM750-200/48_70_Ro
900/52 : ✓ None

Suppliers
Gear supplier : ✓ Flender
Generator supplier : ✓ ABB
Blade supplier : ✓ LM
Brake supplier : ✓ Svendborg
Controller supplier : ✓ Dan Control

Tower
Tower Height : ✓ 68,6
No. of Tower sections : ✓ 3
Trafo inside Tower : X No
Tower type : ✓ Round tower
✓ Round tower
X Edged
?

Std turbine item no : ✓ 2627100
Main Specification : ✓ TIC-019 D

Blade : ✓ LM 23
Bed Frame type : ✓ Welded
Nacelle cover type : ✓ Loose Top

RPM 1 : ✓ 22,3
RPM 2 : ✓ 1500

Voltage 1 : ✓ 480 V

Sime brake type : ✓ None
Generator Vibration Softe : ✓ Yes
Gear enlarged tightening : X No
No. of holes in Flange : ✓ 60
No. of Yaw gears : ✓ 4

Pitch angle : ✓ -3,00000004470341
Marking : ✓ 23

Figure 14. *Screen shot of the configurator*

3.6. *Case Study Phase 6 – Implementation*

The final implementation of the product model is yet to be carried out. The prototype described in phase 6 shows a huge potential for implementing the product model. Several considerations must be encountered before a full-scale implementation is possible. A cost-benefit analysis of a full-scale implementation has been made. It is based on quantitative advantages, qualitative advantages, expenses for IT and internal activities. It is found that the payback period will be less than one year. Based on the prototype described in this paper, the company has decided to continue the work and implement a product configuration system.

3.7. *Phase 7 – Maintenance*

The product master and the object oriented model guides the set up of a structure in the software and serve as documentation of the system.

4. Conclusion

The proposed procedure is based on well known and proved theory elements. The aim of the procedure is to serve as guidance for engineers working with product modelling. The procedure has been tested at several manufacturing companies in Denmark and abroad with positive results.

The proposed procedure includes several fields of expertise:

- Business process reengineering, and business strategy
- Product design and manufacturing technology
- Theory for structuring mechanical systems, and structuring product and product related models
- Object oriented modelling
- IT, artificial intelligence and knowledge representation
- Organisational aspects of application of product modelling

The wide range of theory is included in the procedure in order to cope with the questions raised in the introduction of the paper. I.e. how to deal with the business processes affected by the models, how to analyse and structure products and how to implement the models in IT-systems.

Only few of us will claim to be an expert in all the fields mentioned. However, engineers working with product modelling will need to obtain some insight into the fields listed. Therefore many engineers working with product modelling could benefit from qualifying themselves within themes of relevance for product modelling, where they have little insight.

Based on the proposed procedure a 4 days intensive course in product modelling has been developed. The aim of the course is to introduce the modelling techniques and analysis methods in the proposed procedure. The experiences from the course have been positive. The procedure also form the basis for a course at DTU for engineering students at graduate level.

The challenge in the future is to rework and further integrate the steps in the procedure in order to make the procedure more operational. Also the development af a tool for documentation of the product models is needed in order to secure a smooth handling of the product master, the class diagram and the CRC-cards. Finally, the principles of the product master will have to further developed in order to include the life cycle properties of the products.

5. References

Andreasen, M.M. et al.: On structure and structuring Workshop – Fertigungsgerechtes Konstruieren, 1995.

Barfod, A & Hvolby H.: Ordrestyring - tidens indsatsområde, Industriens Forlag, 1997.

[Bellin et al., 1997]: David Bellin, Susan Suchman Simone; *The CRC Card Book;* Addison-Wesley Longman, inc., 1997.

[Bennett et al., 1999]: Simon Bennett, Steve McRobb, Ray Farmer; *Object oriented systems analysis and design*, University Press, Cambridge, Britan, 1999.

Booch G., Rumbaugh J. & Jacobson I.: The Unified Modeling Language User Guide, Addison-Wesley, 1999.

Coad, P. & Yourdon E.: Object-oriented analysis, second edition: Prentice Hall, New Jersey, 1991.

Hammer, M.: Reengineering Work: Don't Automate, Obliterate, Harvard Business Review, July-August, 1990.

[Hammer et al., 1993]: Michael Hammer, James Champy; *Reengineering the Corporation*, Harper Collins Publishers, 1993.

Harlou U.: A product family master plan as basis for product modeling and engineering design, 1999.

Hubka, V. & Eder, W.E.: Theory of Technical Systems, Springer-Verlag. Berlin, 1988.

Hvam, L.: Application of product modelling – seen from a work preparation viewpoint, Ph.D. thesis, Department of Industrial Management and Engineering, Technical University of Denmark, 1994.

Hvam, L. & Riis, J.: CRC Cards for Product Modelling, Department of Manufacturing Engineering, Technical University of Denmark, 1999.

Hvam L., Hansen B.L.: Strategic guidelines for application of product models; The 4th Annual International Conference on Industrial Engineering Theory, Applications and Practice, San Antonio, Texas, November 17-20 1999.

Hvam L., Mortensen N. H., Riis J. and Hansen B.L.: Produktmodellering – procesanalyse, produktanalyse, objektorienteret analyse, Department of Industrial Management and Engineering, Technical University of Denmark, 2000, 180 pages (In Danish).

ICAM project group: Integrated Computer-Aided Manufacturing (ICAM) Architecture part II, Vol. IV-Function Modelling Manual (IDEF-0); Soft Tech Inc, MA USA, Juni1981.

Johnson, G. & Scholes K.: Exploring Corporate Strategy, Prentice Hall 1993.

Krause, F.: Knowledge integrated product modelling for design and Manufacture, The Decond Toyota Conference, Aichi Japan, 1988.

Mertz, J. & Vølund, L.: Master thesis: Product Configuration at NEG Micon A/S, Technical University of Denmark, 2000.

Mortensen, N. H., Yu, B., Skovgaard, H. and Harlou, U.: Conceptual modeling of product families in configuration projects, 2000.

[Tiihonen et. al., 1996]: Tiihonen, J., Soininen, T., Männistö, T., Sulonen, R.; *State-of-the Practice in Product Configuration – a Survey of 10 Cases in the Finnish Industry*, Helsinki University of Technology, 1996.

[Weigel et. al., 1994]: Rainer Weigel, Boi Faltings; *Constraint-based knowledge representation for configuration systems*, Technical Report No. TR-94/54, Ecole Polytechnique Federale de Lausanne, Department d'Informatique, 1994.

This page intentionally left blank

Chapter 2

Identification of Scheduling Problems: The DeSAP Interface within the e-OCEA Environment

Claudine Tacquard and Franck Thibaut
University of Tours, France

1. Introduction

The study of scheduling problems involves the identification of the problem, its modelization and finally its solution using appropriate techniques and tools. Within the scope of these researches, the work presented in this paper focuses on the identification of scheduling problems. Our purpose is to provide the modeliser with a decision support system to first describe the very problem to be dealt with and then to determine the theoretical problem to be solved. This work is part of a more ambitious project named e-OCEA that is currently under development at the Laboratoire d'Informatique de Tours. e-OCEA software[1] offers a platform to develop tailored algorithms to solve specific scheduling problems. The e-OCEA environment includes several interconnected modules, sharing common data standards. Among those modules, DeSAP enables definition of the physical structure of a flexible manufacturing system (FMS), then the definition of the product to be manufactured and finally the identification of the scheduling problem that has to be solved in order to find the best way to optimise production.

The next section gives a global presentation of the DeSAP module and points out its specificities within the e-OCEA environment. The third section recalls some principles of the underlying notation. Section four presents an enhancement of this initial description to integrate the products to be manufactured within the workshop previously described. The last section gives an example and presents the whole analysis that DeSAP software can provide. In conclusion future prospects are discussed.

2. Global view of the DeSAP module

e-OCEA is a decision support system offering schedules generating tools and following their implementation into the workshop [T'KI 01]. Its architecture is built as a logical bus on which modules can be plugged. The master part of this bus is a database. This database contains information on existing scheduling algorithms (classified by problem and method), previously generated data and schedule sets and problem notations. Other modules are dedicated to the conception of algorithms, to data generation, to the comparison of algorithms etc. A major step before beginning the design of a scheduling algorithm is to identify the scheduling problem and search for information on pre-existing results. DeSAP[2] is one of the two graphical tools that helps the decision maker in this initial phase [TAC 01].

The DeSAP module is composed of three distinct parts:

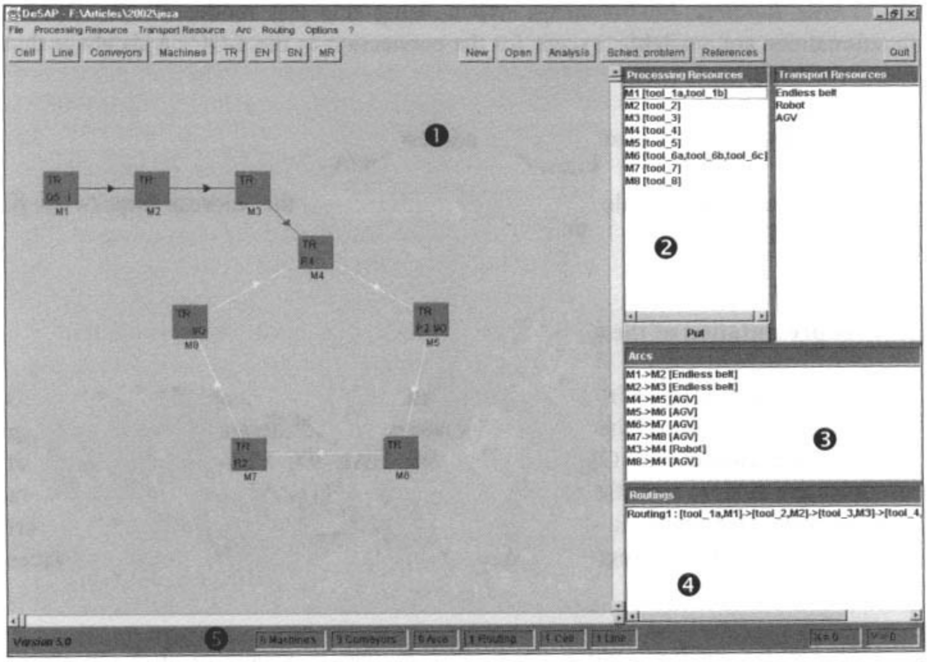
– A graphical interface that enables the user to describe a FMS and displays the notation results.

[1] The web site e-OCEA is available at the address http://www.ocea.univ-tours.fr

[2] DeSAP is a module of e-OCEA that is also available at the address http://www.li.univ-tours.fr/DeSAP

– The analyser that enables analysis of the initial FMS description, to carry out a consistency check, to identify complex structures and to generate the notation of the FMS.

– The scheduling identification that is connected to the e-OCEA database. Based on the definition of product routings, this part identifies the associated scheduling problems, provides bibliographical references and eventually appropriated algorithms to solve the scheduling problem.


DeSAP - F:\Articles\2002\insa
File Processing Resource Transport Resource Arc Routing Options ?
Cell Line Conveyors Machines TR EN SN MR
New Open Analysis Sched. problem References Quit
❶
M1 M2 M3 M4 M5 M6 M7 M8
❷
❸
❹
❺
Put
Version 5.0

Processing Resources
M1 [tool_1a,tool_1b]
M2 [tool_2]
M3 [tool_3]
M4 [tool_4]
M5 [tool_5]
M6 [tool_6a,tool_6b,tool_6c]
M7 [tool_7]
M8 [tool_8]

Transport Resources
Endless belt
Robot
AGV

Arcs
M1->M2 [Endless belt]
M2->M3 [Endless belt]
M4->M5 [AGV]
M5->M6 [AGV]
M6->M7 [AGV]
M7->M8 [AGV]
M3->M4 [Robot]
M8->M4 [AGV]

Routings
Routing1 : [tool_1a,M1]->[tool_2,M2]->[tool_3,M3]->[tool_4,

Figure 1. *Main window of the DeSAP module*

The DeSAP interface (Figure 1) contains the following elements:
❶ : A graphical display of the FMS elements,
❷ : The lists of resources involved in the FMS, be they processing resources or handling resources,
❸ : The links between resources,
❹ : When known, the routings of the products to be manufactured,
❺ : A recall of the analysed elements

The DeSAP software is developed with Visual Café WDE 2.0, running under the Windows NT environment [THI 00]. It is connected to the e-OCEA database by an ODBC driver and uses the JDBC technology. The structure of DeSAP allows two different modes of execution: an Applet mode that is a distant utilization via HTML pages and an autonomous mode that enables local execution on the user's computer.

The Applet mode is controlled by the *Security Manager* in the client's navigator; consequently restrictive conditions are applied (currently, the user's personal examples cannot be saved on the remote server). This applet mode uses the connection to the e-OCEA database and it is then possible to obtain information and references on the related scheduling problems. The autonomous mode can also be retrieved from the same web site. It requires a virtual machine (JVM) to interpret the compiled code. This virtual machine can either be JDK (*Java Development Kit*) or JRE (*Java Runtime Environment*). When using this last mode, all the DeSAP functionalities are available, except for the connection to the e-OCEA database that is not available.

The applet module is available at the address http://www.ocea.univ-tours.fr.

The autonomous module can be downloaded from the address http://www.li.univ-tours.fr/DeSAP.

3. Brief presentation of the notation

Our objective is to define a generic notation that can be applied to any FMS configuration. In most of the notations presented in the literature, only processing resources are considered [GRA 79, BLA 96, HAL 97, VIG 99]. The transport resources are seldom specified. Only the notation proposed in [LIU 96] presents a more realistic approach. It offers "standard configurations" involving transport resources, but with no regard to the resources number or the inter-resources connection.

The DeSAP analyser relies on a notation that is generated from an initial definition of three categories of basic elements: processing resources, handling resources and arcs associating processing resources with handling resources [MAR 99]. A basic principle in the notation is that any of the FMS resources is modelled as a generic storage means made up of three stocks (i.e. input, internal, output) respectively modelling the input capacity, the processing/handling capacity and the output capacity of the resource (Figure 2).

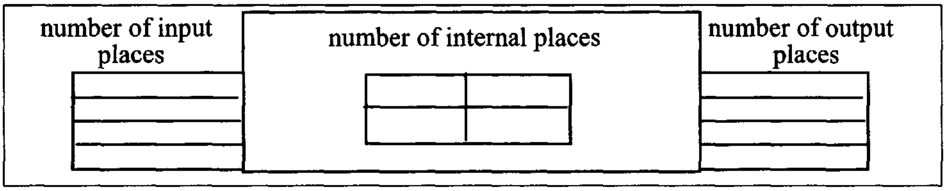


Figure 2. *Basic diagram for any resource in an FMS*

For example, an AGV with transport capacity reduced to only one part is modelled by the triplet (0, 1, 0); a CNC lathe machine able to simultaneously process n parts is modelled by the triplet (number of places in the input stock, n, number of places in the output stock).

A stage of machines as encountered in hybrid flow shop problems or in job shop with duplicated machines is also modelled by a unique processing resource labelled "stage". This "stage" resource contains several processing resources; thus, the capacity of its internal stock is equal to the number of resources it includes. These resources included can have their own processing characteristics (i.e. identical, proportional or unrelated). In this configuration a stage of machines containing N processing resources with common input and output stocks is modelled by the triplet (number of places in the input stock, N, number of places in the output stock).

3.1. *Basic elements of the notation*

Relying on the previous principle, the description of the FMS structure consists of the definition of the different resources it includes. This is done using the following elements only. The DeSAP software offers appropriate menus to facilitate the description of each of those three types of basic elements. After their definition, the user can graphically display them on the left part of the screen to reproduce the FMS under study (Figure 1).

Processing resources gathered:

– Real processing resources (CNC machines, balances, warehouse etc),

– Virtual processing resources that do not realize physical transformations in the product, but refer to decisional treatments related to the manufacturing process: (switching node on an AGV path, exchange zone that permits the transfer of product between two manipulators),

– Meta-resources that are fictitious processing resources including several resources. They can be described as FMS. This type of resources allows a recursive modelling of the FMS by authorizing the insertion of already partially modelled FMS.

A list of characteristics describes any processing resource:

– identification,

– nature (Real: TR, Virtual: EN or SN, Meta: MM),

– number of elements (a processing resource modelling a stage of several resources can specify if they have identical P, proportional Q or unrelated R operational characteristics, otherwise 1),

– list of tools available on this resource,

– characteristic of an exchange resource: this field is instantiated by the analyser (E: Exchange, N: None ; see § 2.3).

– functioning mode (series S/derivation D: all parts in the input stock will or not be processed on the resource),

– interaction with the outside (to specify if a product can enter or leave the workshop: I, O, IO),
– number of places in the stocks and their time boundaries.

During the analysis phase, the definition of each processing resource is transcribed using a standardised notation (see §3.3). For example, in the notation given in Figure 5, the string:

machine1 TR 1 (Tool1_M1) NDIO(0:-,1:-,0:-)

models the real processing “machine1” using the tool1_M1. This resource is in derivation (D). The products can enter or leave the FMS on this resource (I/O). The capacities of its stocks are (0, 1, 0) with no time bound. The character N is generated during the analysis phase (see § 3.2). It represents the interaction of this resource with other complex structures in the FMS. In this case the machine considered has no direct contact with other analysed complex structures (N stands for None, otherwise E for Exchange).

Handling resources are elements ensuring the transfer of the parts between processing resources: AGV, conveyor, belt, robot, arm manipulator, palletiser, and crane etc. They are gathered in a pool if they obey the two following conditions: to have identical characteristics, to be able to ensure the same transport.

A pool is characterised by the following characteristics:
– identification,
– number of identical resources in the pool,
– physical limits of the part to be handled (weight, shape, size etc),
– number of places in the stocks.

The FMS notation also recalls those characteristics. For example, in Figure 5, the string:

“conveyor (1) [ ] (0,1,0)”

models a pool of 1 unit of the handling resource “conveyor”, with no physical limit and with the stock capacities (0,1,0).

Arcs model the transportation of parts between processing resources. An arc models a physical unit when it represents the connection between two processing resources using a conveyor belt. But an arc can be more abstract when it models the transportation of an AGV or a robot between several processing resources.

An arc is described by:
– the name of the handling pool ensuring the part transfer,
– the identification of the resources situated at the origin and destination,
– the orientation of the transport,

- the transportation time (loaded or not),
- the number of tracks (when several tracks exist, more than one handling resource belonging to the pool can simultaneously use the arc without collision risk).

In the notation (Figure 5), the string:

(machine1,machine2)R(10-15,1)

models an arc between the two processing resources machine1 and machine2. Transportation times are specified by the user (10-15) and only one track is available. Transportation can occur in both directions (R for return).

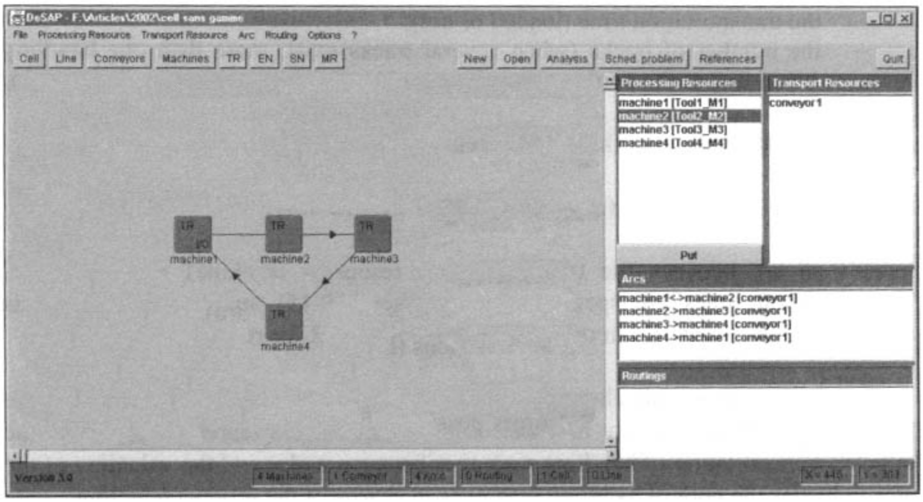
From those three basic elements, it then becomes possible to represent the whole FMS using a directed graph, in which nodes are processing resources, arcs are the directed transportation of handling resources between nodes and the valuation of the arcs indicates the pool of handling resources in charge of the product handling.

The next step in the DeSAP modelization process is to analyse this graph in order to extract an organization. An objective of the notation is then to provide the modeliser with a synthetic view of the FMS that presents more structured and controllable elements. The generated notation also becomes an entry to another software that automatically generates the control software of the studied FMS [MAR 99].

3.2. *The complex structures analysed*

In order to provide a notation in which more structured items appear, the DeSAP analyser identifies two types of complex structures.

A **cell structure** is made up of at least two processing resources served by a single handling pool that operates on a set of arcs. For example, this structure can refer to a cell of the flexible ring type (Figure 3).


File Processing Resource Transport Resource Arc Routing Options ?
Cell Line Conveyors Machines TR EN SN MR
New Open Analysis Sched problem References Quit
TR
machine1
machine2
machine3
machine4
Processing Resources
Transport Resources
Put
Arcs
Routings
Version 5.0

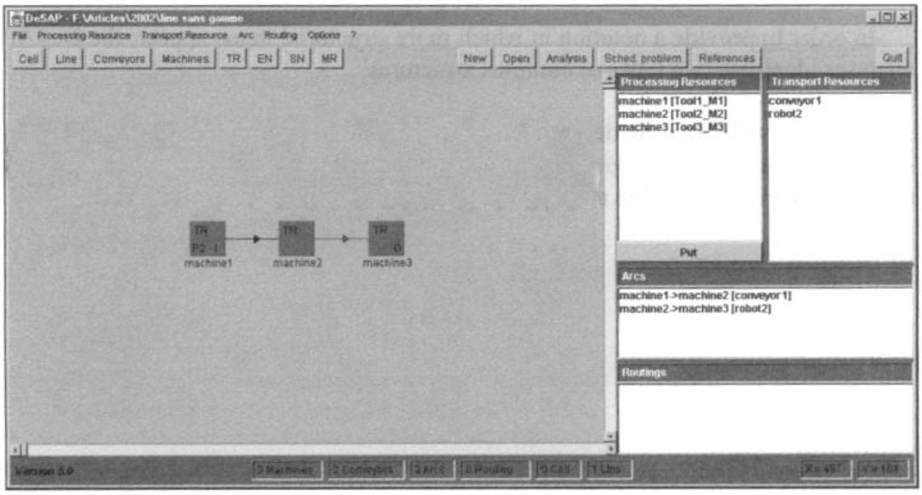
Processing Resources:
machine1 [Tool1_M1]
machine2 [Tool2_M2]
machine3 [Tool3_M3]
machine4 [Tool4_M4]

Transport Resources:
conveyor1

Arcs:
machine1<->machine2 [conveyor1]
machine2->machine3 [conveyor1]
machine3->machine4 [conveyor1]
machine4->machine1 [conveyor1]

Figure 3. *A cell structure*

A **line structure** is made up of at least two processing resources and several handling resources assigned to subsets of disjoined arcs (Figure 4). When several processing resources operate on a line, each determines an elementary line and the common processing resource is labelled "exchange resource". For example, this line structure can refer to a flow-shop or a hybrid flow-shop. It contains two elementary lines and machine 2 is labelled "exchange".


File Processing Resource Transport Resource Arc Routing Options ?
Cell Line Conveyors Machines TR EN SN MR
New Open Analysis Sched problem References Quit
TR
machine1
machine2
machine3
Processing Resources
Transport Resources
Put
Arcs
Routings
Version 5.0

Processing Resources:
machine1 [Tool1_M1]
machine2 [Tool2_M2]
machine3 [Tool3_M3]

Transport Resources:
conveyor1
robot2

Arcs:
machine1->machine2 [conveyor1]
machine2->machine3 [robot2]

Figure 4. *A line structure*

Any combination of lines or cell structures can be identified. In any of those configurations, the processing resources ensuring the connection between two complex structures are also recognised as exchange resources and the string corresponding to their definition also receives an additional “E”.

3.3. *The notation generated*

The DeSAP analyser uses a grammar that identifies the basic elements involved in the FMS and generates a notation according to five levels of description. The full description of the grammar can be found in [TAC 01].

Each level of the notation gives details on the elements of the FMS. The first levels present a raw identification of the complex structures analysed while the last recalls the definition of all the basic elements and gathers them in complex structures.

When several complex structures are identified, they appear successively in the notation. Each cell is specified between the characters {} and each line between [ ] (see also §5, Figure 13). Within those characters, the description of the complex structure is as follows: notation of each processing resource involved in the complex structure { / notation of one handling resource // notation of the arcs shared by this handling resource }*.

Considering the previous cell example (Figure 3), the five levels of its notation are given in Figure 5. In the detailed notation (level 5) the different characteristics of the resources (i.e. identification, nature, interaction) are recalled along with the definition of the path followed by the handling resource.

Level	Analysis
1.	1C ** This FMS contains 1 cell...
2.	1C ; 4M ; 1T ** ... including 4 processing resources and 1 handling resource.
3.	1C ; 4D ; 1T **All the processing resources are of the derivation type
4.	1C {4D / 1T } ** If several complex structures exist, each is summed up in notation 4
5.	1C {machine1 TR 1 (Tool1_M1) NDIO(0:-,1:-,0:-), machine2 TR 1 (Tool2_M2) NDN(0:-,1:-,0:-), machine3 TR 1 (Tool3_M3) NDN(0:-,1:-,0:-), machine4 TR 1 (Tool4_M4) NDN(0:-,1:-,0:-) / conveyor1(1) [] (0,1,0) // (machine1,machine2)R(10-15,1) ; (machine2,machine3)(5-10,1) ; (machine3,machine4)(5-10,1) ; (machine4,machine1)(10-15,1)}

Figure 5. *Notations generated by the DeSAP analyser for a cell structure (Figure 3)*

The next figure (Figure 6) only presents the level 5 of the notation corresponding to the line structure. It models a 3 stages hybrid flow shop. The first stage contains two identical processing resources. The two last stages contain one processing resource. Two different pools of a single handling resource respectively ensure the transportation of parts between the first two stages and the last two stages.

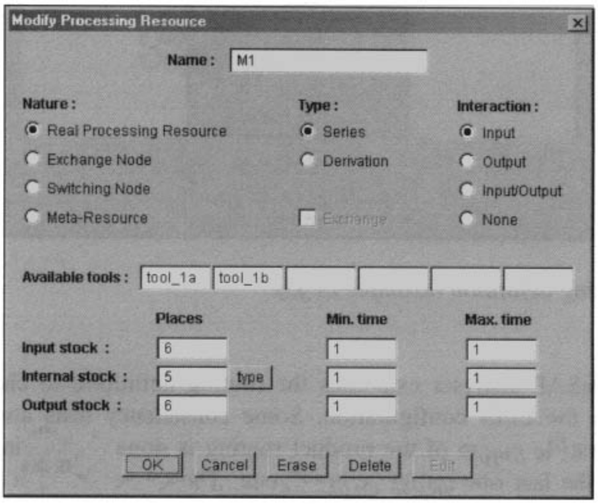
Level	Analysis
5.	1L [machine1 TR P (Tool1_M1) NSI(0:-,2:-,0:-), machine2 TR 1 (Tool2_M2) ESN(0:-,1:-,0:-), machine3 TR 1 (Tool3_M3) NSO(0:-,1:-,0:-) / robot2(1) [] (0,1,0) // (machine2,machine3)(3-3,1) / conveyor1(1) [] (0,1,0) // (machine1,machine2)(5-10,1)]

Figure 6. *Level 5 of the notation generated for a line structure (Figure 4)*

Within those different levels of notation, only the most detailed one is used to automatically generate the control software of the FMS [MAR 99].

4. Identification of scheduling problems

The third function ensured by DeSAP is the identification of scheduling problems associated with a specific production on the FMS under study. The analysis is based on both the FMS configuration given by level 5 of the notation and the description of the products to be manufactured. For that purpose, each processing resource is associated with one or several tools (Figure 7).


Modify Processing Resource

Name : M1

Nature : [✓ Real Processing Resource] Exchange Node Switching Node Meta-Resource

Type : [✓ Series] Derivation

☐ Exchange

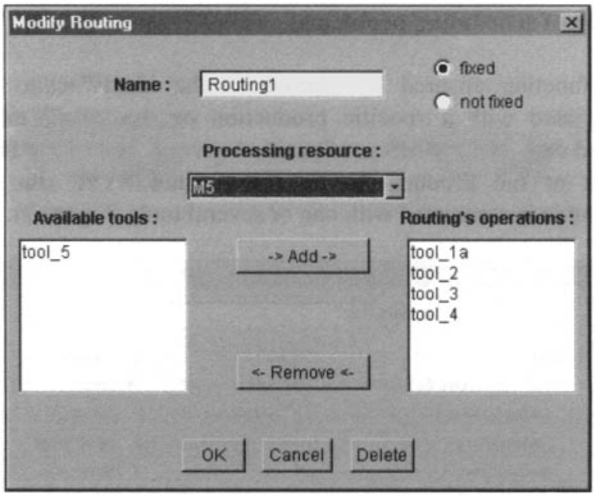
Interaction : [✓ Input] Output Input/Output None

Available tools : tool_1a tool_1b

	Places	Min. time	Max. time
Input stock :	6	1	1
Internal stock :	5	1	1
Output stock :	6	1	1

Figure 7. *Tools definition (example in §5)*

We consider that a tool is situated only on one resource. Then, a routing is defined by a sequence of operations where each operation requires a single tool (Figure 8). It is possible to specify different routings to manufacture different types of products. A set of operations without precedence constraints between them (the case in an open shop) is defined by a routing with the characteristic "not fixed" meaning that the operations are unordered. For the time being, DeSAP do not permit the definition of alternative routings with optional operations.


Modify Routing

Name: Routing1

[✓ fixed] not fixed

Processing resource: M5

Available tools: tool_5

Routing's operations: tool_1a, tool_2, tool_3, tool_4

Figure 8. *Routing definition (example in §5)*

First the DeSAP analyser examines the routing definition to check whether it corresponds to the FMS configuration. Some consistency tests are conducted to verify if the first operation of the product routing is done on an "input processing resource" and the last one on an "output" one. Then a second test focuses on the sequence of operations that must be realistic, i.e. when two consecutive operations are not processed on contiguous resources, these must belong to a complex structure in which the resources that are not involved in the routing are either defined in a "derivation" mode or are a "virtual processing resource". It enables the product to go through the resource without undergoing any processing operation.

For example, considering the previous cell example where all the processing resources are defined in a "derivation" mode, the next routing R1 is valid. This routing does not reference any tool situated on machine M4.

R1 = Tool1_M1->Tool3_M3->Tool2_M2->Tool1_M1

But, considering the line example where all the processing resources are defined in a "series" mode, the next routing R2 cannot be validated because M2 must be included in the routing.

R2 = Tool1_M1->Tool3_M3

4.1. *Classification of scheduling problems*

In the first step, the identification process situates the scheduling problem in the classification proposed by [MAC 93]. Depending on the number of valid routings, the number of operations on the resources and the possible presence of stages, DeSAP identifies an element among the following classification [T'K 00]:

– Single machine (1): the products require only one operation performed on a unique machine.

– Parallel machines (P, Q, R): the products need only one machine that is present for a single stage of identical, uniform or unrelated machines.

– Flow shop (Fn): each product type follows an identical flow pattern in the FMS. Thus, only one routing is defined.

– Job shop (Jn): each product type follows its own individual flow in the FMS. Thus several routings must be considered.

– Open shop (On): there is no specific flow pattern for any product type. The operations on the products can be done in any order.

– Hybrid flow shop (FHn): flow shop in which there are ki machines in each stage (i = 1,..., m).

– Job shop with duplicated machines (JDn): job shop in which there are ki machines in each stage. For the FH and JD classes, any product requiring a stage needs to be processed on only one of those machines.

This first analysis references only one of the previous classes with no details about the real configuration of the FMS. The skeleton of the identification algorithm is described below.

```
if (number of unordered routing == 0) then
    if (number of routings == 1) then
        if(number of operation == 1) then
            if (internal place == 1) then Class ← 1
            else Class ← P, Q or R
        else [number of operation > 1]
            if (all the internal places == 1) then Class ← F
            else (there is at least one resource with internal place > 1)
                Class ← FH
    else [number of routings > 1]
        if (all the internal place == 1) then Class ← J
        else [there is at least one resource with internal place > 1]
            Class ← JD
else [number of unordered routing > 0] Class ← O
```

Then, for the classes F, FH, J, JD and O, the number of distinct resources referenced in the routings permits specification of n, (i.e. FH3 for an hybrid flow shop with 3 stages).

4.2. *Definition of the $\alpha/\beta/\gamma$ notation used to retrieve references from the database*

The next phase consists in refining the previous description using a more detailed notation in the $\alpha / \beta / \gamma$ format. It provides more precise information on the type of resources visited by the product routings. This step is essential to retrieve problems and bibliographical references from the e-OCEA database. This database uses a notation commonly used by researchers in the scheduling community. The objective is then to find the best match between a problem identification provided by DeSAP and the description of more specific problems, as stored in the e-OCEA database.

Most of the different notations proposed in the literature [GRA 79] [BRU 95] [PIN 95] [BLA 96] do not specify all the characteristics or the handling resources in charge of the product transportation. Only the notation proposed by [VIG 99] gives some details of stages in the hybrid flow shop case. The α notation we retain in the DeSAP identification enhances the latter to take the specificities of the handling resources into account. It is as a mix between the very detailed notation (level 5) and those proposed in the literature. The fields β and γ depend on constraints relative to the jobs and to criteria that are not included in level 5 of the generated notation, thus they are not automatically deduced.

The field $\alpha = \alpha_1\alpha_2,((\alpha_3\alpha_4^{(l)})_{l=1}^{\alpha_2})(\alpha_5^{(m)})$ describes the resources environment.

α_1 : Class of scheduling problem (ϕ, 1, m, F, FH, J, JD, O) where m ∈ {P, Q, R}
α_2 : Number of stages (ϕ, N).
α_3 : Type of the resource at stage l (ϕ, P, Q, R), where:
- P (identical processing resources)
- Q (uniform processing resources)
- R (unrelated processing resources)

α_4 : Number of processing resources at stage l (ϕ, N).
The couple $\alpha_3\alpha_4$ is repeated for each stage.
α_5 : Number of sample of the handling resource in charge of the transport between two consecutive resources (be they a stage or not).

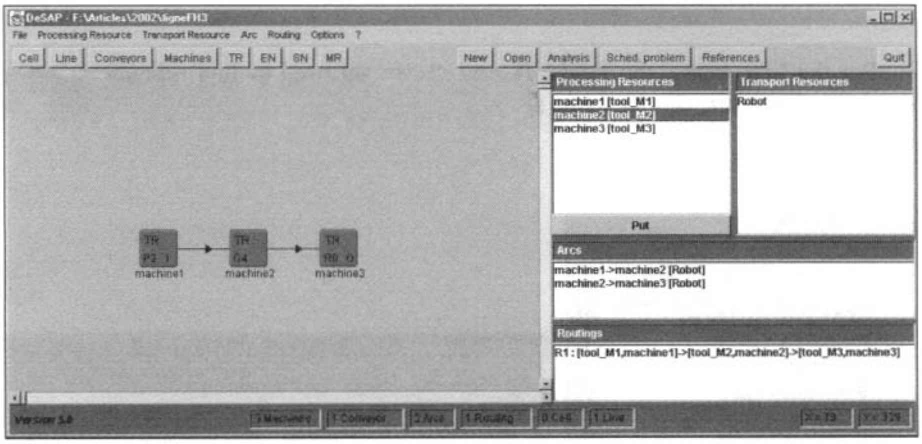
An advantage to the DeSAP modelization process is to help users in the definition of their scheduling problem. Thus, without being an expert in scheduling identification, specialists can easily describe their real FMS configuration and the routings of their products. It often appears that very different FMS configurations can produce the same α notation and lead to identification of identical theoretical scheduling problems. The following example illustrates this possibility.

Let us consider the α field:

$$\alpha = \mathrm{FH3(P2,Q4,R8)((1:}^{1\text{-}2,2\text{-}3}))$$

It describes an hybrid flow shop (FH) with three stages where there are two identical processing resources at the first stage, four uniform processing resources at the second stage and eight unrelated processing resources at the third stage. One pool of handling resources is present in this hybrid flow shop, that ensures the transportation of parts between the three stages.

This α field can be generated for different physical configurations of FMS. Figures 9 and 10 respectively present FMS with a line structure and a cell structure. The definitions of the routings R1 and R2 of the products to be manufactured allow an identical analyse of the associated scheduling problem.



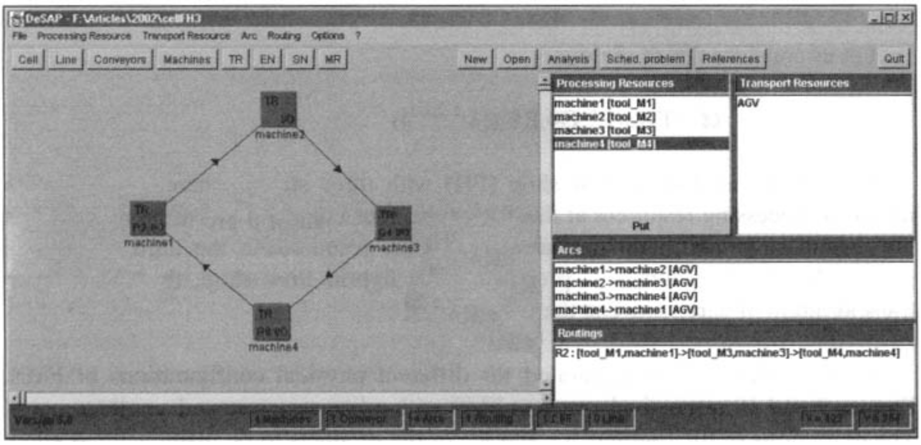
Processing Resources
machine1 [tool_M1]
machine2 [tool_M2]
machine3 [tool_M3]

Transport Resources
Robot

Arcs
machine1->machine2 [Robot]
machine2->machine3 [Robot]

Routings
R1 : [tool_M1,machine1]->[tool_M2,machine2]->[tool_M3,machine3]

Figure 9. *FMS with a line configuration*


DeSAP - F:\Articles\2002\cellFH3
File Processing Resource Transport Resource Arc Routing Options ?
Cell Line Conveyors Machines TR EN SN MR
New Open Analysis Sched. problem References Quit
machine1
machine2
machine3
machine4
Put

Processing Resources
machine1 [tool_M1]
machine2 [tool_M2]
machine3 [tool_M3]
machine4 [tool_M4]

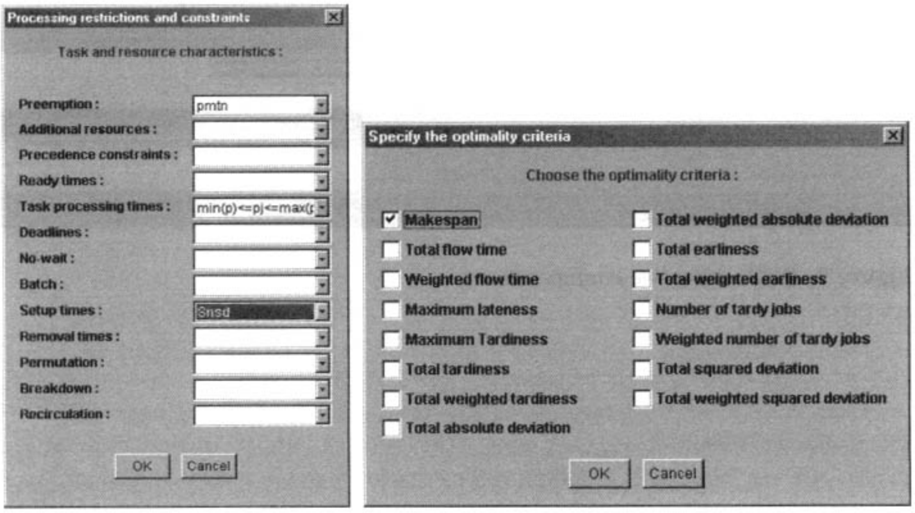
Transport Resources
AGV

Arcs
machine1->machine2 [AGV]
machine2->machine3 [AGV]
machine3->machine4 [AGV]
machine4->machine1 [AGV]

Routings
R2 : [tool_M1,machine1]->[tool_M3,machine3]->[tool_M4,machine4]

Figure 10. *FMS with a cell configuration*

For the time being, the fields β and γ are not used to find references in the database. However, the user can specify them via menus to obtain a full notation (Figure 11).


Processing restrictions and constraints
Task and resource characteristics :
OK
Cancel
Specify the optimality criteria
Choose the optimality criteria :
OK
Cancel

Preemption :	pmtn
Additional resources :	
Precedence constraints :	
Ready times :	
Task processing times :	min(p)<=pj<=max(
Deadlines :	
No-wait :	
Batch :	
Setup times :	Snsd
Removal times :	
Permutation :	
Breakdown :	
Recirculation :	

- [✓] Makespan
- [ ] Total flow time
- [ ] Weighted flow time
- [ ] Maximum lateness
- [ ] Maximum Tardiness
- [ ] Total tardiness
- [ ] Total weighted tardiness
- [ ] Total absolute deviation
- [ ] Total weighted absolute deviation
- [ ] Total earliness
- [ ] Total weighted earliness
- [ ] Number of tardy jobs
- [ ] Weighted number of tardy jobs
- [ ] Total squared deviation
- [ ] Total weighted squared deviation

Figure 11. *Definition of fields β and γ*

At the end, the identification process generates a more standardised notation using the three fields $\alpha / \beta / \gamma$:

FH3(P2,Q4,R8)((1:$^{1-2,2-3}$)) / pmtn, min(p)<=pj<=max(p), snsd / Cmax

This complete notation is stored in a file named "desap.not" and allows querying of the database. Some researches are under development to automatically identify some characteristics of the β field [SOU 01]. They are based on a more precise definition of the resources. For example, the presence or absence of input or output stocks in the resource, the possibility to have several handling resources simultaneously delivering a part to a processing machine.

5. Example of analysis

Let us consider an example of FMS (Figure 12). It includes eight processing resources: M1..M8; the products can enter the FMS on M1, M5 or M8 and can leave it on M5 or M8; The processing resources M1, M2 and M3 are in "series" and all the others are in "derivation".

The handling system is composed of three resources: one endless belt (Tapis1), one robot (Robot1) and one AGV (Chariot1).

Figure 1 also illustrates the graphical representation of this FMS using DeSAP.

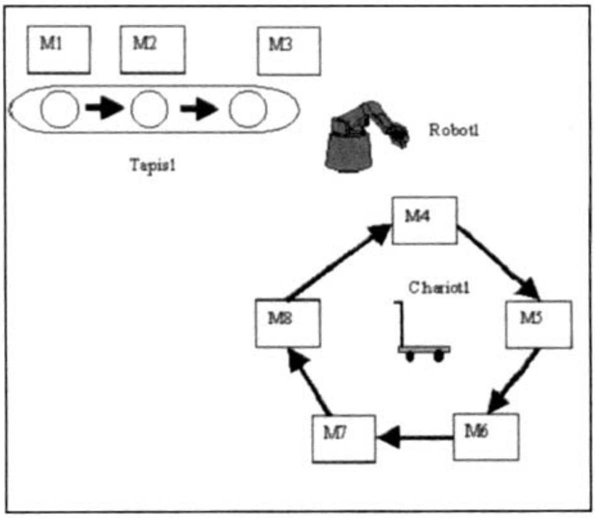


Figure 12. *FMS example*

A first analysis of the FMS, independent of any routing definition, generates the notation with five levels of details. It identifies several complex structures: one cell and one line. Figure 13 presents the most detailed notation. In this example, resource

M4 is an exchange resource that appears in the two complex structures. Resource M3 is also an exchange resource between two elementary lines involved in the line structure (the first elementary line involves resources M1, M2 and M3 and uses "tapis1" and the second involves M3 and M4 and uses "Robot1").

Each processing resource is associated with tools (i.e. M1 has the tools tool_1a, tool_1b; M2 has the tool_2 etc). Only one routing is defined in this example: it contains five operations, each one requiring a specific tool that is available in only one processing resource.

Routing 1: tool_1a (M1) -> tool_2 (M2) -> tool_3 (M3) -> tool_4 (M4) -> tool_5 (M5)

When a product follows this routing, all the resources of the line structure are used, but only a reduced part of the cell structure is involved (i.e. resources M4 and M5). The handling resource "chariot 1" ensures the product transfer from M4 to M5. Then, unloaded, it continues its way through the cell structure to perform the next product transfer.

The same FMS configuration can be used with any other routing specification, with no need to re-analyse its physical characteristics and generate a new notation.

Level	Analysis
5.	1C , 1L {M4 TR R (tool_4) EDN(7:2-2,4:2-2,7:2-2), M5 TR P (tool_5) NDIO(2:1-9,2:2-5,2:4-4), M6 TR 1 (tool_6a,tool_6b,tool_6c) NDN(4:7-8,1:4-5,8:1-2), M7 TR R (tool_7) NDN(2:1-1,2:1-1,2:1-1), M8 TR 1 (tool_8) NDIO(5:5-5,1:4-5,5:4-5) / AGV(1) [] (0,1,0) // (M4,M5)(-,) ; (M5,M6)(-,) ; (M6,M7)(-,) ; (M7,M8)(-,) ; (M8,M4)(-,)} [M1 TR Q (tool_1a,tool_1b) NSI(6:1-1,5:1-1,6:1-1), M2 TR 1 (tool_2) NSN(5:4-8,1:2-2,5:7-9), M3 TR 1 (tool_3) ESN(2:1-7,1:7-8,6:6-6), M4 TR R (tool_4) EDN(7:2-2,4:2-2,7:2-2) / Robot(1) [] (0,1,0) // (M3,M4)(-,) / Endless belt(1) [v<2ms-1] (0,3,0) // (M1,M2)(2-3,1) ; (M2,M3)(-,)]

Figure 13. *The notation of level 5 of the FMS*

DeSAP identifies if the scheduling problem defined by routing 1 belongs to the Hybrid Flow Shop class and can be roughly noted as FH5. Actually, the routing defines a sequence of five operations that are processed on contiguous resources. In addition, three of those resources have internal capacities greater than one (i.e. M1, M4 and M5 have an internal stock respectively equal to 5, 4 and 2), which indicates the presence of stages of resources.

After the user has specified further characteristics of β and γ (given here as an example), the $\alpha / \beta / \gamma$ notation generated in "desap.not " is the following:

FH5(Q5,1,1,R4,P2)((1:1-2,2-3)(1:3-4)(1:4-5)) / pmtn, prec, ri, Snsd / Cmax

When DeSAP is run in the e-OCEA environment, it can retrieve information on this type of scheduling problem from the e-OCEA database. The result is displayed in the following format (Figure 14): first the database records matching exactly to the problem under study (here FH5), then the records where field α begins with the same scheduling class (here FH).

	Notation	Reference	Title	Year	Book	Authors
1.	FHk/Ssd/Cmax	[AGH 95]	Hybrid flow shop ...	1995	IEPM'95	Aghezzaf, Artiba, Moursly...
2.	FHk/no-wait/Cmax	[GUI 96]	A computational ...	1996	IJPR	Guinet, Solomon, Kedia...
3.	FHk//$\bar{C}$	[VIG 96]	Resolution of some 2 stages hybrid ...	1996	SMC'96	Vignier, Billaut, Proust...

Figure 14. *Examples of references in relation with the FH5 scheduling problem*

The DeSAP module also provides a report on the whole analysis in a HTML file. This last one details the different phases of the analysis (notation, classification, bibliographical references etc).

6. Conclusion and prospects

In this paper, a decision support system for describing flexible manufacturing systems and identifying associated scheduling problem was presented. A function of the DeSAP module is to offer a platform permitting an easier communication between the various FMS designers and users. Industrialists and scheduling researchers can use it with different objectives. The first can find an assisted support system to describe their production workshop and the last ones can use it an interface to the other e-OCEA modules. The FMS notation is automatically generated by an analysis of basic elements such as processing and handling resources. Then, from this notation and thanks to the DeSAP/e-OCEA interaction via a database, it becomes possible to identify scheduling problems and to propose information for their solution.

7. References

[AGH 95] AGHEZZAF E.H., ARTIBA A., MOURSLI O., TAHON, C., "Hybrid flow-shop problems, a decomposition based heuristic approach", Proceeding of international Conference on Industrial Engineering and Production Management (IEPM'95), Marrakech, 1995, pp 43-56.

[BLA 96] BLAZEWICZ J., ECKER K.H., SCHMIDT G., WEGLARZ J., *Scheduling in computer and manufacturing systems*, Springer-Verlag, 1996.

[BRU 95] BRUCKER P., *Scheduling Algorithms*, Springer-Verlag, 1995.

[GRA 79] GRAHAM R., LAWLER E., LENSTRA J., RINNOOY KAN A., "Optimization and approximation in deterministic sequencing and scheduling theory: a survey", *Annals of Discrete Mathematics*, vol.5, p. 287-326, 1979.

[GUI 96] GUINET A., SOLOMON M.M., KEDIA P. K., DUSSAUCHOY A., "A computational study of heuristics for two stages flexible flowshop", *International Journal of Production Research*, vol. 34, n° 5, 1996, p. 1399-1415.

[HAL 97] HALL N., KAMOUN H., SRISKANDARAJAH C., "Scheduling in robotic cells: classification, two and three machine cells", *Operations Research*, vol. 45, n° 3, 1997, 421-439.

[LIU 96] LIU, J and MACCARTHY B., "The classification of FMS scheduling problems", *International Journal of. Production Research*, vol. 34, n° 3, 1996, 647-656.

[MAC 93] MACCARTHY B., and LIU J., "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling", *International Journal of Production Research*, vol. 31, n° 1, 1993, p. 59-79.

[MAR 99] MARTINEAU P., TACQUARD C., ROUILLON A., "Génération automatique de la conduite d'un système flexible de production", *Journal Européen des Systèmes Automatisés*, vol. 33, n° 7, 1999, p. 815-853.

[PIN 95] PINEDO M., *Scheduling Theory, Algorithms and Systems*, Englewood Cliffs, NJ: Prentice Hall, 1995.

[SOU 01] SOUKHAL A., *Ordonnancement simultané des moyens de transformation et de transport*, PHD dissertation, Université de Tours, 2001.

[T'K 00] T'KINDT V., BILLAUT J-C., *L'ordonnancement multicritère,* Presses Universitaires de Tours, 2000.

[T'K 01] T'KINDT V., BILLAUT J-C., MARTINEAU P,. TACQUARD C., PROUST C., "The OCEA Project: Towards a decision Support System for Solving Scheduling Problems", Internal report, November 2001, LI/E3i/Université de Tours.

[TAC 01] TACQUARD C., and MARTINEAU P., "Automatic notation of the physical structure of A flexible manufacturing system", *International Journal of Production Economics*, vol. 74, n°1-3, 2001, p. 279-292.

[THI 00] THIBAUT F., TACQUARD C., "Spécification et notation d'un Système Flexible de Production: Problèmes d'ordonnancement associés et interface avec OCEA", End of course project, E3i/ Université de Tours, 2000.

[VIG 96] VIGNIER A., BILLAUT J-C., PROUST C., T'KINDT V., "Resolution of some 2-stage hybrid flowshop scheduling problems", IEEE Conference on Systems, Man and Cybernetics (SMC'96), Pekin (China), vol. 4, 1996, pp. 2934-2941.

[VIG 99] VIGNIER, A., BILLAUT J-C., PROUST C., "Les problèmes d'ordonnancement de type flow-shop hybride - Etat de l'art", *RAIRO / Recherche Opérationnelle*, vol. 33, n° 2, 1999, p. 117-183.

This page intentionally left blank

Chapter 3

Product Generic Modelling for Configuration: Requirement Analysis and Modelling Elements

Michel Aldanondo, Khaled Hadj-Hamou and Jacques Lamothe
Centre de Génie Industriel, Ecole des Mines d'Albi-Carmaux, France

Guillaume Moynard
Centre de Génie Industriel, Ecole des Mines d'Albi-Carmaux, France, and Lapeyre-GME, Aubervilliers, France

1. Introduction

To improve competitiveness, companies try to launch products with customisation capabilities. They therefore include configuration software *(configurator)* in their information systems. In consequence, most of the ERP providers include in their software packages configuration modules (see the surveys of (Sabin *et al* 1998)). In order to function, these configuration modules require a generic model of the product, which is not normally defined by a computer science specialist. Therefore, there is a strong need for friendly generic modelling approaches and relevant tools in order to set up configurators in industry.

Many papers dealing with configuration are more concerned with a solutions approach and the relevant generic modelling problem is normally not addressed. As Wielinga explains in (Wielinga *et al* 1997)

"Many authors prefer a parsimonious set of knowledge structures that suit their favourite problem solving method or their domain".

Oriented towards configuration requirements and modelling, our contribution targets two goals. The first, dealing mainly with product generic modelling requirements, is to identify and to classify the product customisation possibilities that should be fulfilled by configurators. The second, dealing with modelling, is to analyse how the *constraint satisfaction problem* (CSP) framework can handle these modelling requirements and to comment how existing algorithms can assist the configuration process.

We first recall configuration basic definitions, discuss the people involved in the configuration process, point out two kinds of configurators, underline general modelling needs and justify our CSP-based modelling choice. We then identify a first set of modelling requirements corresponding to what we call the "central problem" and provide modelling elements. Then, in order to match other particularities of industrial problems, we sequentially add modelling requirements to this "central problem" and analyse how CSP based generic modelling elements can fulfill them. These complementary elements gather: a product descriptive approach, tailored or parametric components, layout definition and hierarchical bill-of-materials.

Our intention is therefore to propose and discuss generic modelling elements without focusing on problem solving. We are clearly interested in showing how CSP can be used to represent various aspects of product generic modelling for the configuration problem. A *custom storage system* provided by the Lapeyre Company will be used as an example to illustrate our ideas throughout this paper.

2. Configuration, configurators and generic modelling

This section recalls general aspects of configuration, introduce the product generic modelling need and justify our CSP-based approach.

2.1. *Configuration definition and configurator presentation*

In previously published works concerning configuration, (Mittal *et al* 1989), (Soininen 1998), (Kühn 1999), (Friedrich *et al* 1999) and (Aldanondo *et al* 2000), common features defining configuring are:

hypothesis: a product is a set of components,

given: (i) a generic model of a configurable product able to represent a family of products with all possible variants and options, in which a generic model is a set of components plus a set of various constraints; and (ii) a set of customer requirements, in which each requirement can be expressed by a constraint,

configuring can be defined as "finding at least one component set that satisfies all the constraints".

It is important to note that, according to these common features, the configuration result is a set of components (a single level bill-of-materials).

A *configurator* is a software package that assists the person in charge of the configuration task. It is composed of a knowledge base that stores the generic model of the product and a set of assistance tools that helps the user to find a solution or select components. In any case, the fundamental assistance requirement is to guarantee the consistency of the configured product with both generic model and customer requirements.

2.2. *Configuration situations*

According to the previous definition, configuration can exist in any customer/supplier relation when defining the product object of a deal. It is nevertheless possible to identify two main classes of configuration problems and relevant configurators.

The first class deals with the selling side and is conducted by the sales teams of companies. In that case, the cycle time order of magnitude for setting a solution is less than an hour and sometimes less than a minute when configuring on line on a web site for example. This is the target of the ERP configuration modules that assist the business to customer (B2C) customer/supplier relation and needs to be able to support mass configuration or a great number of configuration tasks (for example: 1 to 100 configuration tasks per day).

The second class is close to a product design activity and is mainly achieved by the research and development teams of companies. An order of magnitude of the cycle time to provide a solution could be between a week and a month. In most of these cases, previous ERP configuration modules cannot be used and specific configurators are necessary. For that class of problems, the customer/supplier relation is more on the business to business side (B2B) and the number of configuration tasks that needs to be achieved is comparatively small (no mass configuration, for example: from 1 to 10 configuration tasks per month).

As the configurators of the second class are specific, each of them (including its generic model) needs to be designed and maintained by computer science specialists. On the other hand, the configuration modules provided by software companies should propose (and provide most of the time) a generic modelling environment enabling a non-computer science specialist to describe the generic model of the product. Our contribution is clearly positioned on the generic modelling side of the first configurator class, frequently called *commercial configurator* or *sales configurator*.

2.3. *Main suitable characteristics of generic product model*

In this section we list and briefly explain the main characteristics that should fulfil the generic model of the product. They will be used in the next section for justification of our a priori choice of modelling formalism. They deal with:

Easy modelling by a non-computer science specialist

The product model and the modelling task should rely on "natural" or easy understanding concepts. The configurable product model should be clearly separated from the configuration process. A graphical representation of the model is a necessity for easy comprehension and a smaller maintenance effort. Pieces of "visible" written code or pseudo-code (whatever the language is) and source of errors during modelling, should be avoided as much as possible.

Generic model expressivity and ability to be used efficiently

The generic model can be different according to the configurator user's level of knowledge of the product. For a product expert, a generic model gathering detailed components plus constraints is suitable, whereas a clearer product representation in terms of product characteristics is necessary for a non expert. In the work of (Mannisto *et al* 1996) this is addressed as an *explicit model (component oriented)* versus an *implicit model (description oriented)*.

Generic model testability

As generic models mainly rely on components plus constraints, it is necessary to have confidence in the model consistency. This is a difficult point and the configuration technique operating on the model should be open to debugging and inconsistency detection techniques.

Structured generic model and reusability of generic model parts

In order to avoid redundancy of parts of generic model, a structured generic model is of interest and permits re-use of these generic model parts in generic models of different products.

2.4. *Model characteristics, configuration techniques and CSP approach*

Most of the work recently published on configuration relies on propositional logic, first order logic and constraint satisfaction problem (CSP) frameworks. Generic modelling with propositional logic requires a large number of Boolean variables and Boolean rules (and, or, not), resulting models are therefore complicated, with low expressivity. First order logic is a good generic modelling tool but is not easy to handle by non computer science specialists; some work (Felfering *et al* 1999) has been done to provide a high level of graphical formalism avoiding coding.

The utilisation of "pure" CSP, introduced by (Mackworth 1977), presents interesting concepts for configuration, but we will see, in section 3.1.2, why its dynamic extension, the DCSP, proposed by (Mittal *et al* 1990), and discussed by (Soininen *et al* 1999) is much better for configuration problem handling.

The clear separation between the model and the propagation or resolution techniques, the concepts of variables/domains/constraints natural for non computer specialist and good graphical representation possibilities make CSP a good candidate for configuration modelling.

The CSP concepts, variables and constraints fit two possible configurable product descriptions: set of components and/or set of characteristics. Furthermore, the work of (Schiex *et al* 1995) exploiting the notion of preference in CSP allows CSP based configurator to take into account the preference of the user during the configuration process.

In terms of model testability, some works (Keirouz *et al* 1995) and (M. Sabin *et al* 1999) have presented some possibilities about CSP model debugging and inconsistency detection, but the proof of a full consistence model still relies in the complete analysis of the solution space. In order to avoid this last drawback, consistency restoration and explanation during configuration can be supported by CSP techniques as explained in (Amilhastre *et al* 2000).

CSP approaches do not deal easily with structural aspects. Very little work has been done in this field; composite CSP has been proposed by (D. Sabin *et al* 1996) and some extensions of Dynamic CSP trying to match this aspect have been discussed in (Véron *et al* 2000). As far as we know, this is clearly not a good argument for CSP utilisation in configuration.

The kinds of product where various aspect of CSP-based configuration have been studied are diverse. A car configuration problem introduced in (Mittal *et al* 1990) and discussed by (M. Sabin *et al* 1999), (D. Sabin *et al* 1996) and (Véron *et al* 2000), an industrial mixer (Soininen *et al* 1999), a machining operation (Geneste *et al* 2000), and an automotive wiring system (Aldanondo *et al* 2001) are typical examples of product where a CSP based configuration has been used. For this work, it must be noted that: the car and industrial mixer product model examples have been used to illustrate theoretical points of CSP based processing, while the goal of the work dealing with machining operation and automotive wiring system examples was to have a product-specific operating configurator without providing any new theoretical points. An interesting survey of product customisation variety has been reported (Soininen 1998) but without CSP based support. Our contribution, therefore aims to study how the diversity of product customisation, or product modelling requirements, can be handled with CSP based elements. This is the aim of the rest of the paper.

3. Product modelling requirements and generic modelling elements

In this chapter, we first present what we call the *central problem*, then each following section will describe some additional modelling requirements for the central problem. For each section, the configuration problem hypotheses are defined or refined, an example is given for clarification and a CSP based model is provided and discussed.

3.1. *From basic to central configuration problem*

3.1.1. *Basic configuration problem and CSP approach*

The simplest configuration problem can be defined as follows:

- h1: all the components are "standard" or completely defined; it is not possible to create a new component during the configuration task,

- h2: the components are gathered in groups; each component must belong to only one group, the purpose of the group notion being to gather components that support the same functionalities,

- h3: each group is present in any configured product,

- h4: the constraints represent the allowed combinations of components,

- h5: the customer or configurator user requirement corresponds to the selection of one component in each group,

- h6: a configured product is a component set, satisfying both constraints and requirements, where one and only one component must be selected in each group.

The example in Figure 1, a "custom storage system", illustrates this problem. The generic product gathers 3 components: a Bookcase (BC), a High Cabinet (HC) and a Low Cabinet (LC), where:

- any component of each group exists in two finishes: Painted (P) or Wood (W),

- the BC height can be 72 cm or 216 cm, LC height is 72 and HC height is 144,

- 3 groups exist (i) BC: {BC72P, BC72W, BC216P, BC216W}, (ii) LC: {LC72P, LC72W}and (iii) HC: {HC144P, HC144W}

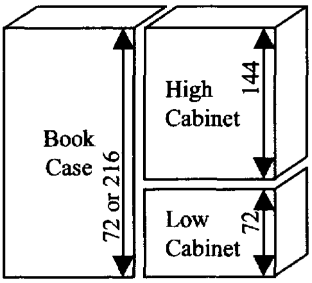


Figure 1. *Basic problem*

- a single constraint states that any configured product must be of the same colour.

CSP, defined (Mackworth 1977) as a triplet {X, D, C} where X is a set of variables, D a set of finite domains (one for each variable) and C a set of constraints (defining the possible combinations of variables value), matches this basic problem. Each group of components is associated with a variable. Each component corresponds to one value of the variable. The constraint (solid lines of Figure 2) represents the allowed combinations of components. A generic model of the product of example 1 could be as the one shown in Figure 2.

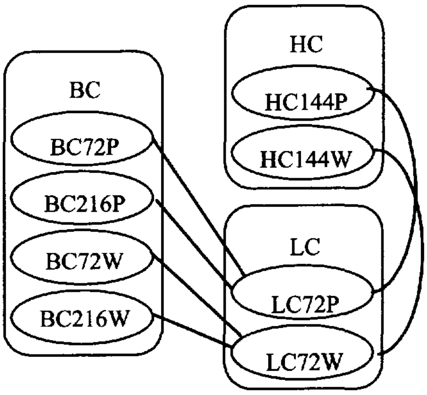


Figure 2. *Basic problem CSP model*

3.1.2. *Central configuration problem and DCSP approach*

This previous problem is extremely rare. Very frequently, the existence of a group of components is optional or restricted for product feasibility reasons or customer wishes. Therefore, two kinds of groups must be defined: the groups which always exist in any configured product and the ones that may exist according to the customer requirements or product feasibility reasons. The central configuration problem can therefore be described as follows:

- h1, h2, h6: unchanged,

- h3 - a group is either always present in any configured product or its existence depends on: (i) the existence of other groups and/or (ii) the selection of other components,

- h4 - the constraints represent the allowed combinations of (i) components and/or (ii) group existences,

- h5 - the customer or configurator user requirement corresponds to the selection of (i) one component in each group and/or (ii) decision of a component group existence.

The example of 3.1.1 is modified with the addition of the following constraints:

- The Bookcase must be present in all configured Storage Systems.

- The High Cabinet can exist if and only if a Low Cabinet exists and the Bookcase is 216 cm high.

This "central problem" is supported by most of the configurator packages provided by software companies (for example: Siebel, Oracle, Cincom, Baan or Trilogy) and is frequently associated with what is called "pick to order" or "assemble to order" industrial situations.

DSCP, proposed by (Mittal *et al* 1990), adds to "pure CSP" the notions of:

- *Initial variables*: variables that exist in any configured product.

- *Compatibility constraints*: equivalent to the constraints defined in section 3.1.1.

- *Activity constraints*: allowing of control the variable existence in the following ways:

- *Require*: a specified value of a variable "x" implies the existence of the variable "y",

- *Always Require*: any value of a variable "x" (or "x" exists) implies the existence of the variable "y",

- *Not Require*: a specified value of a variable "x" implies the non existence of the variable "y",

- *Always Not Require*: any value of a variable "x" (or "x" exists) implies the non existence of the variable "y".

A generic model of the central problem example could be as the one shown in Figure 3. It is clear that DCSP matches exactly the "central problem" requirements. In the following sections we are going to show that the requirements presented are not sufficient to take into account various industrial situation needs and provide complementary requirements and analyse how DCSP can fulfil them.

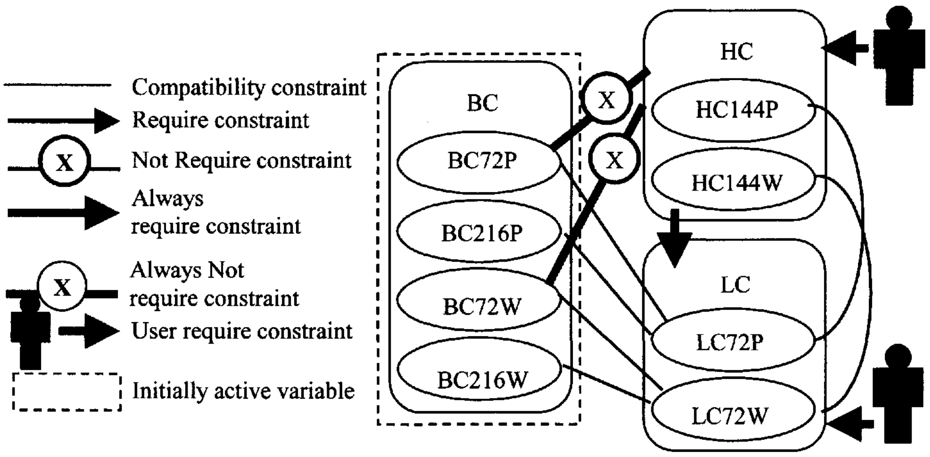


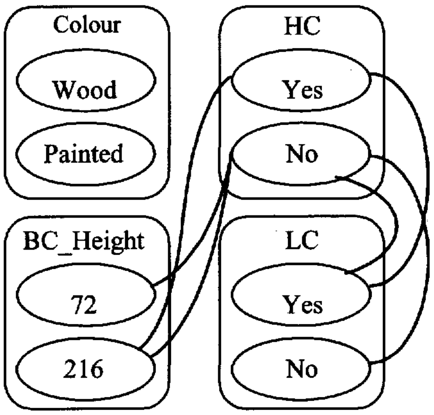
Figure 3. *Central problem DCSP model*

3.2. *Extending the central problem and adding modelling elements*

3.2.1. *Physical versus descriptive model*

The previous model in Figure 3 is a "physical" model because each variable corresponds to a group of components. But very often, for a non product expert, it is necessary to define "descriptive" attributes that do not correspond to a component group. In that modelling approach, the values of the descriptive attributes permit one to identify a component (for example the height and the colour allow identification of a component). The advantages are, on the one hand, that descriptive attributes (defined by the person in charge of modelling) can be much more expressive for the user than a list of components (that does not interest the user most of the time) and, on the other hand, that configuration models can involve much fewer configuration variables and constraints. But the second approach needs to maintain identification tables or functional constraints allowing one to derive the component list from the values of the descriptive attributes.

The example of Figure 3 is presented in Figure 4 with this second modelling approach with the same DCSP formalism. The four descriptive attributes, necessary to represent exactly the same set of solutions, correspond with the colour (wood / painted) the book case height (72 / 216) the existence of the lower cabinet and higher cabinet (yes / no). The component identification tables permit derivation, from the four previous variables, of the final result of configuration as a set of BC, LC and HC components.



Colour	BC_Heig.	Component
Wood	72	BC72W
Wood	216	BC216W
Painted	72	BC72P
Painted	216	BC216P

Colour	HC	Component
Wood	Yes	HC144W
Painted	Yes	HC144P

Colour	LC	Component
Wood	Yes	LC72W
Painted	Yes	LC72P

Figure 4. *Central problem descriptive model*

Extended definition of the central problem

In the first definition, components were gathered in groups. With the descriptive approach we added descriptive attributes and values. We therefore introduce the notion of configuration variables that gathers these two elements and can propose the following definition:

- H1 - All the components, "standard" or completely defined, are gathered in groups; each component must belong to only one group. Each group is associated with a configuration variable whose definition domain is a list of values equal to the component list.

- H2 – A product can be characterised by descriptive attributes. Each descriptive attribute is associated with a configuration variable whose definition domain is a list of values, where a value cannot be a component.

- H3 - A configuration variable is either always present in any configured product or its existence depends on: (i) the existence of other configuration variables and/or (ii) the selection of other configuration variable values.

- H4 - The constraints represent the allowed combinations of (i) configuration variable values and/or (ii) configuration variable existences.

- H5 - The customer or configurator user requirement corresponds to the selection of (i) one value for each configuration variable and/or (ii) decision of a configuration variable existence.

- H6 - A configured product is a set of configuration variable values, satisfying both constraints and requirements, where (i) one and only one component is selected in each group and (ii) one and only one value is selected for each existing descriptive attribute.

Discussion

Globally, the descriptive model aims to displace the configuration from a component plus constraint problem to a descriptive attribute plus constraint problem plus component identification tables. This permits a kind of disconnection between the product configuration process and the bill-of-materials generation. The interest of this disconnection lies in a kind of delinking of selling and manufacturing concerns allowing remote configuration possibilities for sale without handling all the product technical data. Therefore this approach is of interest for commercial configurators.

When there is no descriptive attribute (i.e. all configuration variables represent component groups), this definition corresponds to the physical problem of section 3.1.2. When all the configuration variables associated with the component groups are only present in component identification tables or functional constraints, this definition corresponds to a "pure" descriptive problem. Most of the time, it is unfortunately necessary to express the requirements of the configurator user in terms of both component selection and product characteristic valuation. The two modelling approaches are therefore mixed in a same configuration problem. DCSP handles this problem, relying on configuration variables gathering group/component and descriptive attribute/value. The next sections will identify other kinds of attribute that will be associated with configuration variables allowing representation of other configuration modelling requirements.

3.2.2. *Tailored components, component quantity and numerical constraints*

Standard and tailored components

Until now, all the components are completely defined with entirely frozen characteristics (hypothesis H1). Very frequently, industrial cases need to tailor components. For example, when a door needs to be replaced in an old house, it rarely corresponds to a standard offer. In order to capture this market, many companies propose customisation possibilities that are not restricted to standard component assembling.

We therefore propose to characterise each tailored component by numerical tailoring attributes with a continuous definition domain defining the range of possible values. Therefore, modelling requires association of each tailoring attribute with a configuration variable defined in a continuous definition domain (Figure 5).

In the example of Figure 5, the Bookcase and the High Cabinet are now tailored components with tailoring attributes BC_height (chosen in a range between 72 cms and 216 cms) and HC_height (chosen in a range between 50 cms and 144 cms).

Component quantity

In the central problem, hypothesis H6 assumes that a configuration solution contains only one component in a group. Very often, it is necessary to model that the quantity of a selected component might be different from 1.

In that case, a quantity attribute is necessary to characterise the quantity selected for a component chosen in a group. This attribute can be defined either on a discrete or continuous domain. This quantity attribute must also be associated with a configuration variable.

In our example of Figure 5, the High cabinet can contain drawers and roll-out-shelves. For each of them, it is possible to select a quantity between 0 and 3 thanks to the quantity attributes for drawers (Qtt_Draw) and roll-out-shelves (Qtt_Ros).

Numerical constraints on discrete variable

Until now, compatibility and activity constraints were discrete. The activity and compatibility constraints represented allowed the combinations of configuration variable values. Very frequently during modelling, it is simpler to define constraints through the expression of a mathematical formula that avoids a huge combinatory even when dealing with discrete variable domains.

Therefore a strong modelling requirement is to handle constraints expressed with formulae. These kinds of formula allow one to take into account among other things that some component selections or attribute valuations are restricted by some kind of resource (space, power, length, etc) provided by other components or attribute values. This behaviour is similar to the resource/provide/consume elements introduced by (Heinrich *et al* 1991) and discussed in (Sabin *et al* 1999).

In our example in Figure 5, let us consider that up to four elements among Drawers and Roll-Out-Shelves can be chosen for the High Cabinet. It is much simpler to express this modelling requirement with a numerical constraint stating that Qtt_Draw.+ Qtt_Ros. ≤ 4 by than describing in extension the possible combinations of quantity.

Numerical constraints on discrete and continuous variable

Before the identification of tailored and quantity attributes, configuration variables were discrete. With these two continuous attributes, it is necessary to express constraints on continuous variables.

A first modelling solution that avoids mathematical formulae is to discretise the variable domains and to use discrete constraints. This is shown in our example of Figure 5 with the compatibility constraint between BC_height and HC existence, where BC_height is discretised in two intervals [72,122] and [122,216], the first interval forbids the HC existence while the second leaves the two possibilities.

When discretisation is not suitable or requires a too large combinatory description, it is necessary to use a mathematical formula. This is shown in the example of Figure 5 where a constraint, between BC_height and HC_height expresses that the top of Bookcase and High Cabinet must be at the same height: BC_height = HC_height + 72.

Tailored attribute, quantity attribute and numerical constraint example

In the example of Figure 5, the Bookcase and the High Cabinet are tailored components with tailoring attributes BC_height and HC_height, with the following constraints:

- the top of Bookcase and High Cabinet must be at the same height: BC_height = HC_height + 72,

- the High Cabinet can exist if: BC_height ≥ 72 + 50 = 122.

Two new component groups, Drawers and Roll-Out Shelves, are added to the High Cabinet. A constraint states that a maximum of four elements among these groups can be chosen. This is modelled with:

- quantity attribute for drawers (Qtt_Draw) and Roll-Out Shelves (Qtt_Ros),

- a numerical constraint expressing that: Qtt_Draw. + Qtt_Ros. ≤ 4.

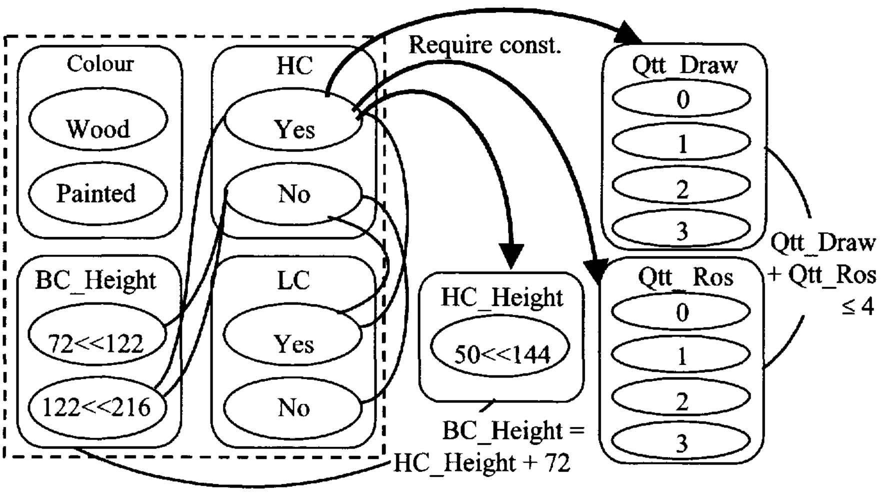


Figure 5. *Tailoring and quantity attributes with constraints*

Discussion

So far as modelling is concerned, configuration variables corresponding with tailoring attributes and quantity attributes, numerical constraints, continuous variables and constraints can be added to the model of the central problem. The problem raised in this section is on the side of configuration processing, DCSP solving and constraint propagation require discrete variables and discrete constraints.

Therefore, other processing techniques must be investigated for assisting the configuration process, which should deal with cooperation of constraint solvers dealing with discrete and numeric variables as proposed in (Tinelli *et al* 1996). As far as we know, very few commercial configurator software programs support discrete and continuous variables, discrete and continuous constraints, and constraints expressed with formulae in a proper way.

3.2.3. *Layout*

In the previous cases, the configuration result is always a set of components plus attribute values. As explained in (Brown 1998), a same set of components can correspond with two different products according to two different layouts. It is therefore sometimes necessary to geometrically locate each selected component among others. This is close to a design activity whose aims are either to locate a component in an absolute referential or to locate the position of one component in relation to another component.

Layout with component absolute locations

Layout requirements using absolute position can be handled thanks to the association of each component with location attributes. Each attribute is a coordinate of a referential and corresponds to a configuration variable. The values of the location attribute can be either discrete or continuous. Discrete and continuous layout constraints can link location attributes of different components. These modelling elements are similar to quantity and tailored attributes modelling. Consequently, they lead to the same configuration process drawbacks.

Without layout configuration, the four layouts in the upper left part of Figure 6 are possible. But let us consider that only the first and second are valid: the LC must be under the HC and the BC near the LC. Then, absolute position layout modelling requires us to define different possible component locations. The space is therefore mapped in 6 possible locations identified by a number: 1/lower left, 2/upper left, 3/upper right, 4/lower right, 12/left, 34/right, as described in the upper right of Figure 6. The model in the lower part of Figure 6 shows that each component is characterised by a location attribute and compatibility constraints define the acceptable layout solutions (1) and (2).

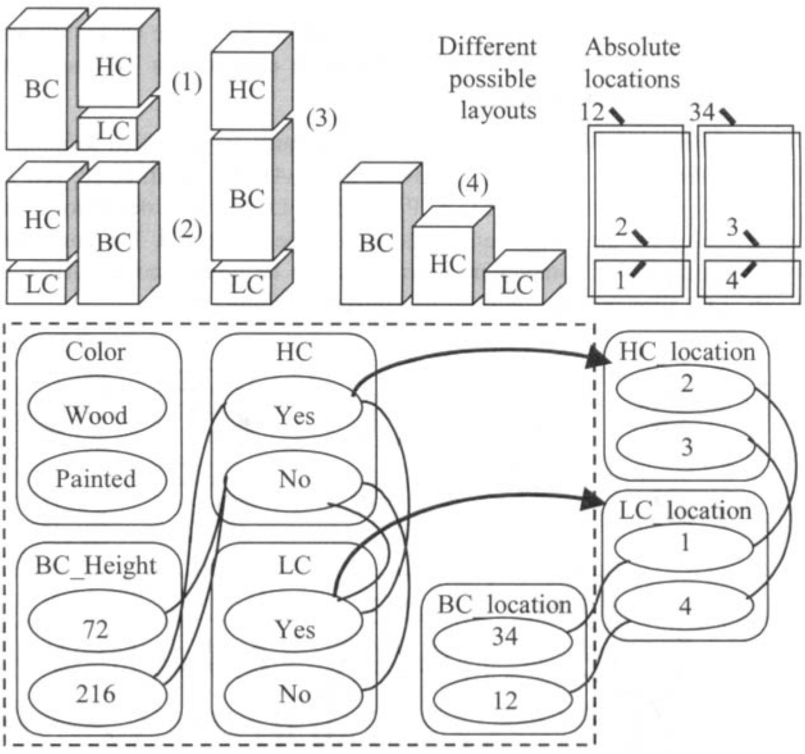


Figure 6. *Layout modelling with location attributes*

Layout with component relative locations

Layout requirements using relative location of component couple have been introduced with the notion of "port and connection" by (Mittal *et al* 1989), defining that two components can be connected through component ports (port_m of component_i can be connected to port_n of component_j). Each port is a characteristic of a component and can be associated with a component port attribute.

In our example, this approach requires location of each component with respect to each other at a certain place (on, under, lined up, etc.) in a way very close to the CAD system. Therefore each component must be characterised by some ports, each port corresponds to a particular side (or piece of side) where a port of another component can be "in contact" as shown in the upper part of Figure 7. Constraints should explain how components could be relatively located.

Layout modelling with relative location or ports is much harder to handle with a CSP based approach but we propose some ideas dealing with this modelisation problem, such as: (i) define a configuration variable for each component port attribute, (ii) define allowed values of the component port configuration variable as the list of component ports that can be connected to it plus the value "∅" specifying that the port is not used, (iii) define possible connections between component couples with compatibility constraints.

The model in the lower part of Figure 7 illustrates the proposed solution. The BC has four ports (P_BC-1, P_BC-2, P_BC-3, P_BC-4), the HC three (P_HC-1, P_HC-2, P_HC-3) and LC three (P_LC-1, P_LC-2, P_LC-3). The definition domains of these variables are the component ports that can be connected. The constraints show how three couples of components plus ports can be connected (BC,HC), (BC,LC) and (HC,LC) allowing the solutions of sub-section Figure 6.

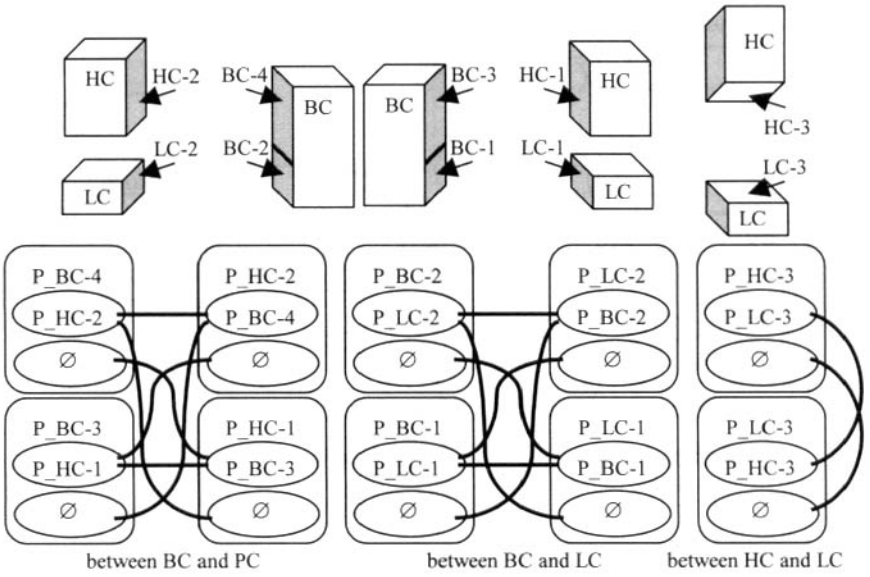


Figure 7. *Layout modelling with ports and connections*

Discussion

Configuration with layout requirements is not easy to handle with our CSP based elements. When the location problem is not too complicated, for example when locating less than 20 pieces of furniture on a single axis with a unique coordinate, absolute position location is fine. Layout requirements taking advantage of the port and connection approach are typical of electronic product configuration. When components and possible connections are numerous, the CSP based model presented becomes quickly complicated and modelling and maintenance not possible.

It is clear that layout configuration is close to a design activity. Simple cases, close to schematic drawings, can be handled with commercial configurators as shown for example in the configuration problem of a train car layout addressed with the Baan configurator in (Heiderscheilt *et al* 1999). But as explained in (Aldanondo *et al* 2001) configuration including hard layout problems are better solved thanks to the cooperation of configurator and CAD system.

3.2.4. *Hierarchical bill-of-materials*

Many industrial situations require, mainly for production management reasons, to have the configuration result (the component set) presented as a hierarchical bill-of-materials instead of a single level bill-of-materials.

For the example of Figure 5 gathering up to five of the components, Bookcase (BC), High Cabinet (HC), Low Cabinet (LC), Drawers (Draw) and Roll-Out Shelves (Ros); the single level bill-of-material is as shown in left part of Figure 8. The hierarchical bill-of-material presentation need may correspond, for example, with the identification of the following sub-assemblies:

- High Cabinet plus Accessories (HCA) gathering: High Cabinet, Drawers and Roll-Out Shelves,

- High and Low Cabinet (HLC) gathering: Low Cabinet and High Cabinet plus Accessories,

and the resulting hierarchical bill-of-materials would be as shown in the right part Figure 8.

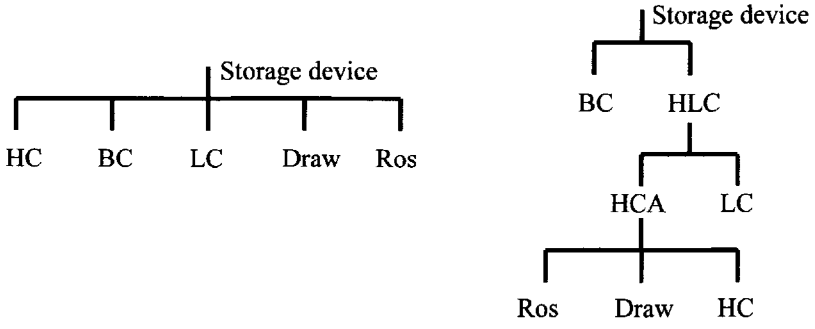


Figure 8. *Single level and hierarchical bill-of-materials*

As this is mainly a presentation requirement, we propose to keep the previous modelling elements (CSP based generic model plus component identification tables representing single level bill-of-materials) and to add a piece of generic model allowing to derive a hierarchical bill-of-materials. The added pieces of model gather a generic hierarchical bill-of-materials plus existence conditions of the generic sub-assembly. The leaves of the generic hierarchical bill-of-materials are the component groups and the intermediate elements are the generic sub-assemblies of the product. The sub-assembly existence conditions correspond with functional activity constraints that modulate the sub-assembly existence as a function of the variables of the CSP based generic model.

With these elements, the configuration can be as follows: (i) configuration of the product (ii) identification of the component (iii) existence validation of the generic sub-assemblies (iv) hierarchical bill-of-material generation. The last step is achieved with a substitution in the generic hierarchical bill-of-materials of each component group by relevant identified component at step (ii). When a sub-assembly does not exist, direct bill-of-material links, between lower level sub-assemblies or components and upper sub-assembly of top level product, replace the generic hierarchical bill-of-materials upstream and downstream links. Lastly, in order to differentiate each sub-assembly, each of them should be codified according to the lower level components and/or sub-assemblies.

For the example in Figure 8, these elements both presented in a table form would be as shown in Figure 9, the hierarchical bill-of-materials in the left part and sub-assembly conditions in the right part.

Father	Son
Storage device	BC
Storage device	HLC
HLC	LC
HLC	HCA
HCA	HC
HCA	Ros
HCA	Draw

generic hierarchal bill-of-materials

Sub-assembly	Existence condition
HLC	Value(HC) = "Yes"
HCA	Value(Qtt_Draw) + Value(Qtt_Ros) > 0

generic sub-assembly existence condition

Figure 9. *Added pieces of generic model*

According to these elements, the whole configuration process would go as follows:

- (i) configuration process with user requirements as: Colour = "Wood", BC_Height = 216, LC = "Yes", HC = "yes", HC_Height = 144, Qtt_Draw = 0, Qtt_Ros = 0.
- (ii) component identification: BC216W (BC group), LC72W (LC group), HC144W (HC group).
- (iii) sub-assembly existence validation: HLC.
- (iv) hierarchical bill-of-material generation: as shown in Figure 10.

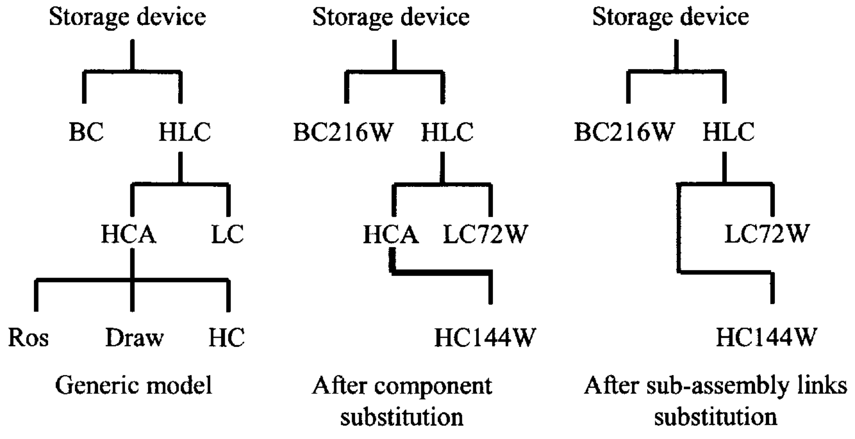


Figure 10. *Hierarchical bill-of-material generation*

Discussion

The added pieces of the model show how a hierarchical bill-of-materials can be generated as a configuration result with our CSP based approach. Some complementary work is necessary to model a product where a same component group is present in different generic sub-assemblies or when the hierarchical bill-of-materials does not have a tree shape.

A main drawback of our entire CSP based approach, which clearly appears in this section, is that it is not easy to configure in a generative way. Generative configuration should be understood as the ability to configure a product with more than one instance of a same generic sub-assembly or with a number of instances unknown at the beginning of the configuration task. Very little work has been done in "generative configuration", an application specific configurator has been designed for telecommunication system (Fleischandel *et al* 1998) and very recently some ideas about duplicating variable groups in CSP model in order to permit generative configuration have been proposed (Véron 2001).

4. Conclusions

Our goal was to identify and classify configuration modelling requirements and analyse how the CSP framework could be a good support for generic modelling and if relevant propagation and solving techniques were adequate.

In terms of identification of requirements and generic modelling, we started with the central problem and a DCSP physical model composed of (i) variables, defined in a discrete domain, associated with component groups, and (ii) compatibility and activity discrete constraints.

We extended the previous central problem, in order to take into account the following generic modelling requirements, and proposed: descriptive approach, parametric or tailored component, multi-presence of a same component, layout configuration and expression of the solution as a hierarchical bill-of-materials. We proposed, for modelling, (i) to associate configuration variables, defined in a discrete or continuous domain, with: component groups, descriptive attributes, tailoring attributes, quantity attributes, location attributes, component ports attributes and generic sub-assemblies; and (ii) constraints, combining the previous variables, that can be compatibility constraints or activity constraints, expressed with allowed combination of values or numerical formulae.

The complexity of the modelling task, with the previous elements, comes from: (i) the mix of all kinds of attributes and the various constraints that can exist in a single problem and (ii) the customisation complexity and size of the product itself. Nevertheless, the descriptive approach, allowing some kind of disconnection between product configuration process and bill-of-materials generation, tends to reduce the modelling task difficulty.

The main drawback of this approach, at present, lies in the fact that the model is not structured. This forbids easy re-use of model parts without "cut and paste", and makes generative configuration very difficult or impossible when the number of model instances is not known before configuring. This important aspect needs further work with probably some object-oriented approach.

In terms of propagation and solving techniques, continuous domain variables and constraints expressed with mathematical formulae are not compatible with the DCSP solving algorithms and most of the work done in consistency checking, inconsistency explanation proposition, constraint propagation and CSP resolution have been achieved with discrete domain variable and discrete constraints. As far as we know, there is no proper way to overcome the problem at present without a delicate combination of: discretisation of continuous variable domains, algorithms reasoning on intervals, or delicate cooperation of different solvers.

Setting an easy to use modelling tool in order to "put on paper" a generic model of a configuration situation is a big issue for configurator deployment in industry. Friendly-user modelling is a necessity, especially for "commercial configurator". The proposed elements present the important interests of being "visible" on a schema and easy to understand by non-specialists thanks to CSP formalism. The model expressivity mixing component groups, various attributes allows one to build configuration model that can be used during configuration by product experts (component oriented) and by non product experts (characteristics oriented).

Many of the proposed modelling elements and a specific model structure approach have been included in a commercial configurator software package called "Caméléon Visual Expert" and in a generic modelling method "Caméléon Model Designer" designed and distributed by Access-Commerce (web site:

http://www.access-commerce.com). This configurator is also the configuration module of the two ERP Mfgpro and Mapics and has been integrated with many ERP.

Thanks to the elements proposed and the "Caméléon Model Designer" modelling method, a great variety of industrial customisable products that did not require a specific configurator have been successfully modelled. Initially defined for manufacturing products, as the example running all through this paper; the proposed modelling elements can be used for service and software configuration.

5. References

Aldanondo M., Lamothe J., Hadj Hamou K., "Configuration and CAD modeler: gathering the best of two worlds", *IJCAI Workshop on Configuration*, Seattle, USA, 2001, p. 1-7.

Aldanondo M., Moynard G., Hadj Hamou K., "General configurator requirements and modeling elements", *ECAI Workshop on Configuration*, Berlin, Germany, 2000, p. 1-6.

Amilhastre J., Fargier H., Marquis P., "Consistency restorations and explanations in dynamic CSP – Application to configuration", *ECAI Workshop on Modeling and Solving Problems with Constraints*, Berlin, Germany, 2000.

Brown D. C., "Defining Configuring", *AI EDAM (Artificial Intelligence for Engineering Design, Analysis and Manufacturing)*, vol. 12, n°4, 1998, p. 301-306.

Felischandel G., Friedrich G., Haselblock A., Stumptner, "Configuring Large Systems Using Generative Constraint Satisfaction", *IEEE Intelligent Systems & their applications*, vol. 13, n°4, July/August 1998, p. 59-68.

Felfering A., Friedrich G., Jannach D., "UML as domain specific language for the construction of knowledge base configuration system", *11th Int Software Engineering and Knowledge Engineering conference*, Kaserlautern, Germany, 1999, p. 337-345.

Freidrich G., Stumptner M., "Consistency-Based Configuration", *AAAI Workshop on Configuration*, Orlando, Florida, 1999, p. 35-40.

Geneste L., Ruet M., Monteiro T., "Configuration of a machining operation", *ECAI Workshop on Configuration*, Berlin, Germany, 2000, p. 44-49.

Heiderscheilt, D., Skovgaard, HJ., "Visualization of configurations: simplifying configuration user interfaces", *AAAI Workshop on Configuration*, Orlando, Florida, 1999, p. 114-118, Orlando, USA.

Heinrich M., Jungst E., "The resource-based paradigm: Configuring technical systems from modular components", *IEEE Conf on Artificial Intelligence Applications*, 1991, p. 257-264.

Keirouz W., Kramer G., Pabon J. "Principles and Practice of constraint programming", Chapter: Exploiting Constraint Dependency Information for Debugging and Explanation, MIT press, 1995, p 183-196.

Kühn C., "Requirements for Configuring Complex Software-Based Systems", *AAAI Workshop on Configuration*, Orlando, Florida, 1999, p. 11-16.

Mackworth A. K., "Consistency in networks of relations", *Artificial Intelligence,* vol. n°8, 1977, p. 99-118.

Mannisto T., Peltonen H., Sulonen R., "View to Product Configuration Knowledge Modeling and Evolution", *Technical Report FS-96-03, Workshop on configuration,* AAAI Press, 1996, p. 111-118.

Mittal S., Frayman F., "Towards a generic model of configuration tasks", *International Joint Conference on Artificial Intelligence IJCAI,* Detroit, USA, 1989, vol. n°2, p. 1395-1401.

Mittal S., Falkenhainer B., "Dynamic Constraint Satisfaction Problems", *9th National Conference on Artificial Intelligence AAAI,* Boston, USA, 1990, p. 25-32.

Sabin D., Freuder E., "Configuration as Composite Constraint Satisfaction", *Technical Report FS-96-03, Workshop on configuration,* AAAI Press, 1996, p. 28-36.

Sabin D., Weigel R., "Product Configuration Frameworks – A Survey", *IEEE Intelligent Systems & their applications,* vol. 13, n°4, July/August 1998, p. 42-49.

Sabin D., Freuder E., "Optimization Methods for Constraint Resource Problems", *AAAI Workshop on Configuration,* Orlando, Florida, 1999, p. 83-89.

Sabin M., Freuder E., "Detecting and Resolving Inconsistency and Redundancy in Conditional Constraint Satisfaction Problems", *AAAI Workshop on Configuration,* Orlando, Florida, 1999, p. 90-94.

Schiex T., Fargier H., Verfaillie G., "Valued Constraint Satisfaction Problems: Hard and Easy Problems", *International Joint Conference on Artificial Intelligence IJCAI,* Montreal, Canada, 1995, p. 631-637.

Soininen T., Tiihonen J., Männistö T., Sulonen R., "Towards a General Ontology of Configuration", *AI EDAM (Artificial Intelligence for Engineering Design, Analysis and Manufacturing),* vol. 12, n°4, 1998, p. 357-372.

Soininen T., Gelle E., "Dynamic Constraint Satisfaction in Configuration", *AAAI Workshop on Configuration,* Orlando, Florida, 1999, p. 95-100.

Tinelli C., Harandi M., "Constraint logic programming over unions of constraint theories", *Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming,* vol. 1118, 1996, p 436-450.

Véron M., Aldanondo M., "Yet another approach to CCSP for configuration problem", *ECAI Workshop on Configuration,* Berlin, Germany, 2000, p. 59-62.

Véron M., "Modélisation et résolution du problème de configuration industrielle: utilisation des techniques de satisfaction de contraintes", Phd Thesis INP Toulouse, 2001.

Wielinga B., Schreiber G., "Configuration design problem solving", *IEEE Intelligent Systems & their applications,* vol. 12, n°2, March/April 1997, p.49-56.

Chapter 4

Production Management Systems

Farid Ameziane and Stéphane Lasserre
Ecole d'Architecture de Marseille, France

1. Introduction

Building construction faces problems in information management and communication throughout all the designing, engineering, realisation and maintenance stages.

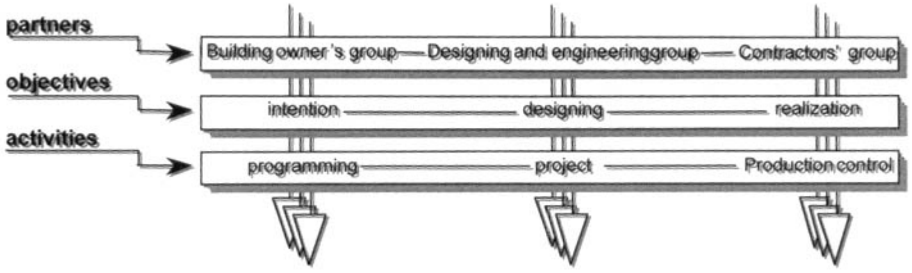

partners
objectives
activities
Building owner's group—Designing and engineeringgroup— Contractors' group
intention — designing — realization
programming — project — Production control

Figure 1. *Schematic representation of partners, activities and objectives in the building construction field*

The architectural project tends to be complex because it must integrate a growing number of regulation requirements and constraints:

- user conveniences (accessibility, acoustic and thermic conveniences, etc.),

- construction (paraseismic properties, complex cost management, etc.),

- realisation (product selection, technical regulations, quality management, reduced construction project duration),

- management of the dynamic information generated by a growing number of partners with distinct skills and viewpoints.

Thanks to the new technologies, which enable a better information exchange between all the partners involved, productivity in the building production activity can be improved. Starting from practice observation, the goal of our research program is to share and extract data from the building description and to keep track of all the information changes and updates throughout the building's lifecycle.

To reach these objectives, it is necessary to build an agile information system framework that would be able to support this data processing. In addition, it means that there must be a normative approach to the description of the building elements in order to ease access to and re-use of technical solutions. Many research programs around the world are focused on these normalisation topics in the building field and more in industrial fields. We will also talk about re-engineering and knowledge capitalisation which were first used in industry.

2. Data exchange in the building context

Building production is a non-linear and complex activity. It is sequenced by distinct stages in which many actors enrich the project description with their own vocabulary and competence. All the actors involved are working towards the same goal: the completion of the construction project. Each one helps the others by sharing the graphical and textual data he produces. These representation and communication tools represent the most efficient way to reach an accurate information flow.

Many research programs focus on building information management through their lifecycle. The goal is to provide the actors with the tools towards concurrent engineering (Chen & Wu, 1993) (Darses, 1997) and more generally towards co-operation among the partners in the building production processes (Chan, and Gu, 1993). The problem of data exchange between heterogeneous CAD systems (Belhi, Erard & Bouras, 1999) produced small or no successfully file exchange formats: DXF, IGES, SET, VDA, CALS, STEP and more recently Industry Foundation Classes (IFC) from the IAI (International Alliance for Interoperability). It is probably the most spectacular one, because it links the major CAD system firms, many CAD users and institutional partners. French research projects (SUC, MOB, GSD, SIGMA, etc.) and international projects (COMBINE, ATLAS, RATAS, IAI, etc.) consider that there is a strong analogy between the architectural and the industrial fields (Ameziane, 1998).

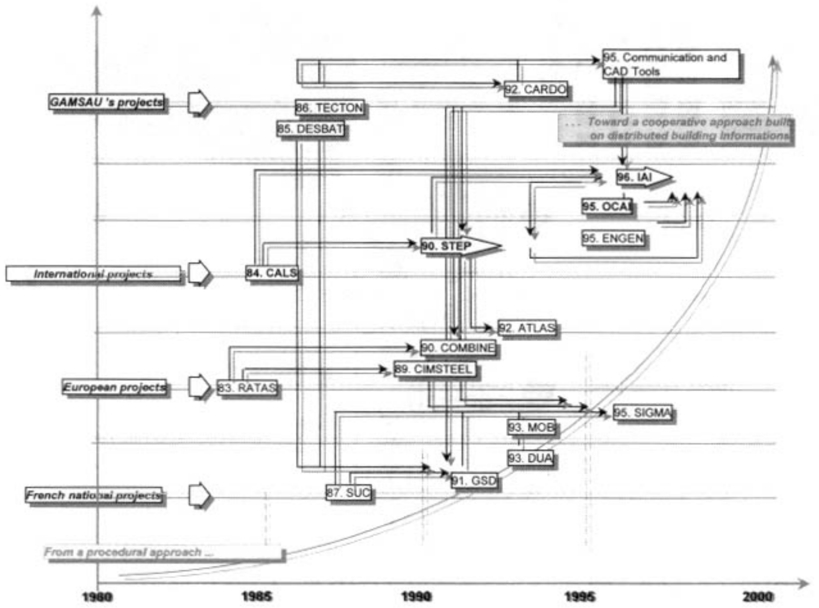


Figure 2. *Building research programs worldwide from 1980 to 2000*

The various research works feature two major trends in the way to describe a building:

- Some research programs focus on the building construction work, and

- Other research programs focus on the process description that leads the construction of the building.

The results of those research works (Ghodous and Vandorpe, 2000) have been used to develop the 'Communication and CAD Tools' research program (Figure 3) in our laboratory.

3. The 'Communication and CAD Tools' research project

Our work originates from the results of previous projects we have dealt with, that are part of the 'product/process model' research programs.

It aims at easing the setting up of a collaborative management information system in the building sector (Ameziane, 1998). It is to be initiated during the engineering stage, enriched through the execution stage, and updated by the owner when the building is in use. Each actor is then able to get the building representation needed. This is represented by Figure 3.

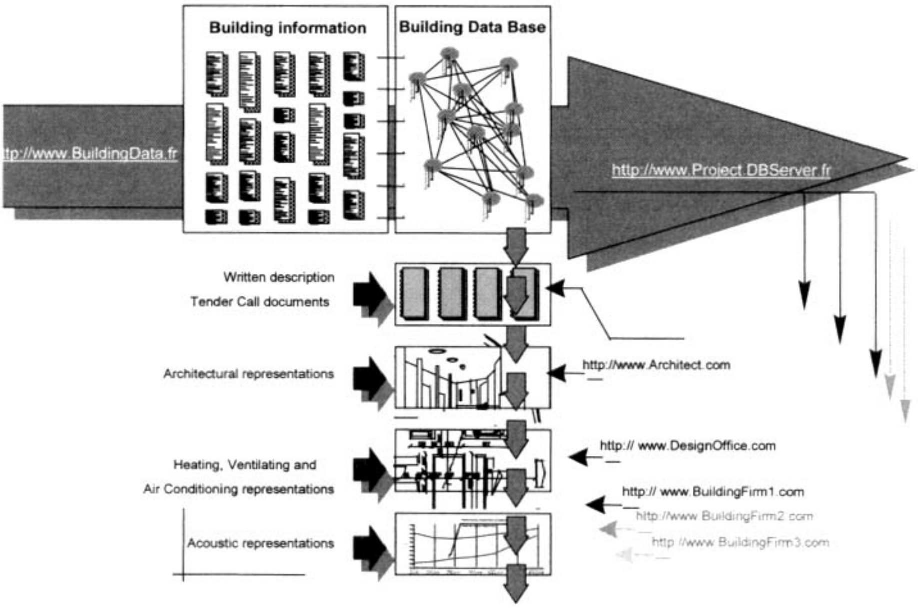

Building information
Building Data Base
tp://www.BuildingData.fr
http://www.Project.DBServer.fr
Written description
Tender Call documents
Architectural representations
Heating, Ventilating and
Air Conditioning representations
Acoustic representations
http://www.Architect.com
http:// www.DesignOffice.com
http:// www.BuildingFirm1.com
http://www.BuildingFirm2.com
http://www.BuildingFirm3.com

Figure 3. *Overview of the 'Communication and CAD Tools' objectives*

This project offers a conceptual schema of structured data matching the knowledge fields of architectural design. This schema allows us to build a consistent group of entities in a data base management system in order to give particular representations of these complex assemblies, and to manipulate them by remote control. Our work was led by the following hypotheses:

- the knowledge environment of the building construction area is distributed but it can be accessed through a network,

- a building may be described by a series of construction works and the spaces it contains. This is from the construction economist's stand point,

- this description may grow step by step, and every partner can access it,

- we consider that a 3D CAD model was elaborated just before the engineering stage. It seems to be the best media to support the finalisation of the building project description.

To answer these hypotheses, we produced a conceptual data diagram that describes the building with the construction works it is composed of.

Under a generic class of 'entity_of_building' we organise all the objects that are part of the construction of a building in a hierarchic set of sub-classes. Each of them is composed of attributes (product specifications) and methods (knowledge relative to a specific product – for example, a HVAC calculation depends on the volume and the capacity of a room).

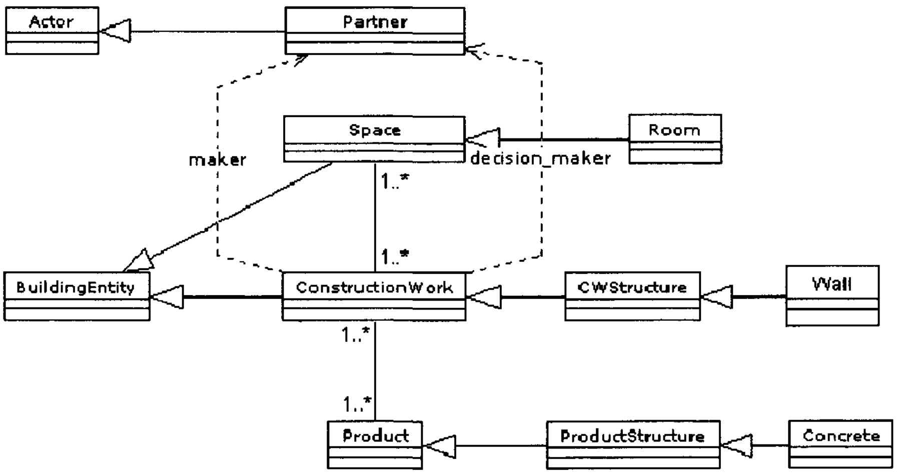
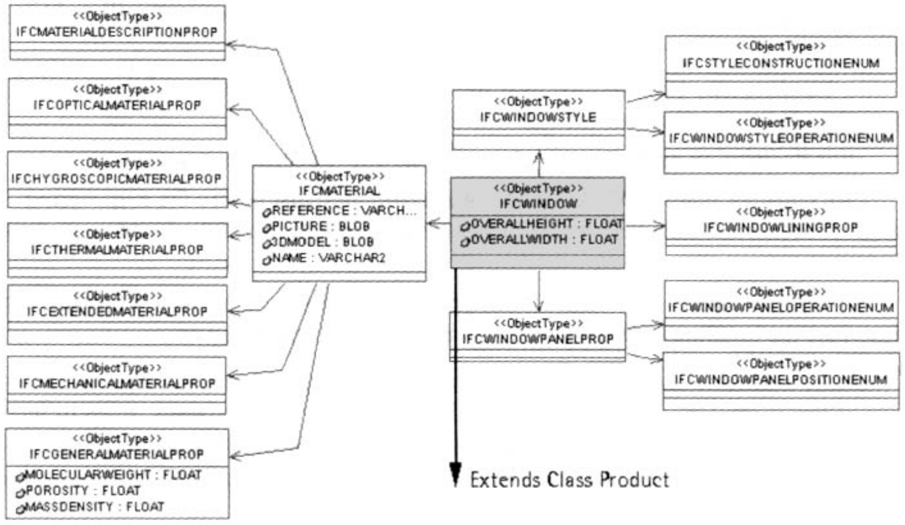


Figure 4. *The information organisation – overview of main classes*

In that prospect, we represent the set of objects manipulated in the description in a computerised environment by coupling any ODBC data source (which specifications map our building data schema) with a solid and parametric CAD system (Mechanical Desktop by Autodesk Inc.) and the Internet. This environment enables the users to maintain the building description through its data diagram which needs to be frequently updated because no real normative project has been elaborated yet.



<<ObjectType>> IFCMATERIALDESCRIPTIONPROP
<<ObjectType>> IFCOPTICALMATERIALPROP
<<ObjectType>> IFCHYGROSCOPICMATERIALPROP
<<ObjectType>> IFCTHERMALMATERIALPROP
<<ObjectType>> IFCEXTENDEDMATERIALPROP
<<ObjectType>> IFCMECHANICALMATERIALPROP
<<ObjectType>> IFCGENERALMATERIALPROP
MOLECULARWEIGHT : FLOAT
POROSITY : FLOAT
MASSDENSITY : FLOAT
<<ObjectType>> IFCMATERIAL
REFERENCE : VARCH...
PICTURE : BLOB
3DMODEL : BLOB
NAME : VARCHAR2
<<ObjectType>> IFCWINDOWSTYLE
<<ObjectType>> IFCWINDOW
OVERALLHEIGHT : FLOAT
OVERALLWIDTH : FLOAT
<<ObjectType>> IFCWINDOWPANELPROP
<<ObjectType>> IFCSTYLECONSTRUCTIONENUM
<<ObjectType>> IFCWINDOWSTYLEOPERATIONENUM
<<ObjectType>> IFCWINDOWLININGPROP
<<ObjectType>> IFCWINDOWPANELOPERATIONENUM
<<ObjectType>> IFCWINDOWPANELPOSITIONENUM
Extends Class Product

Figure 5. *The 'Communication and CAD tools' information organisation extract with IFC building object descriptions (Industry Foundation Classes from IAI). View of an Oracle database schema sample using the UML formalism*

A database is attached to each architectural project in order to complement its morphologic description with textual data that usually characterises a construction work.

3.1. *Help interface for decision support system*

Very often, data manipulated by partners who collaborate in raising the building are compiled into professional systems and cannot be re-used by the other users. Data are transformed through these systems and the added information cannot be understood.

Furthermore, once the building has been built, its owner may legally request all the information compiled through the stages. So, through the operating stage of the building (the longest one), searching information is harder when we have to maintain or to find responsibilities for specific problems.

Our project aims at finding an answer to this problem by sharing a unique data model with all the partners involved in the building construction processes.

The following figure describes what factors lead a decision regarding a construction work choice during the engineering stage in France (Armand and Raffestin, 1993). We also illustrate the role of each actor in this process and the knowledge fields shared between one another.

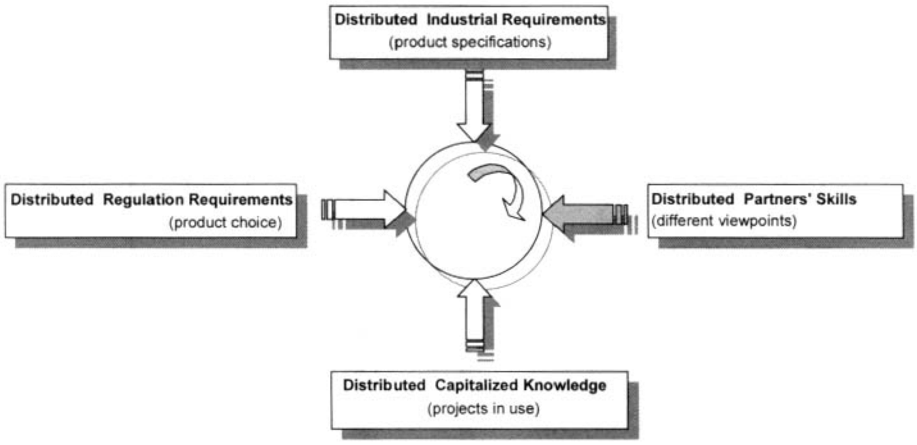

Distributed Industrial Requirements
(product specifications)
Distributed Regulation Requirements
(product choice)
Distributed Partners' Skills
(different viewpoints)
Distributed Capitalized Knowledge
(projects in use)

Figure 6. *Distributed knowledge for construction works decisions*

Each actor may want to make a proposal for a construction work according to his own knowledge and viewpoint on the building. We found that they all have common legal constraints and need to respect the products' functional features.

As for the building lifecycle, legal aspects may be considered as the leading decision-making factor:

- during the engineering stage in order to comply with the product specifications and use,

- during the realisation as it is necessary to check the compliance of the works to improve the quality process,

- when the building is in use, in order to improve its maintenance and find responsibilities for bad implementations by tracking back the design history of the building.

Consequently, by taking information continuity problems into consideration at every stage, we are refining our building description schema with detailed legal specifications of construction works. We consider that the latter are part of the most important points that lead decision-making and building completion processes.

The following picture represents a framework using 'radio buttons' that enables the end user to configure the system according to his own needs. At this stage, we need to stress the fact that we do not really want to avoid final confrontation between partners, but only to get rid of unnecessary exchanges.

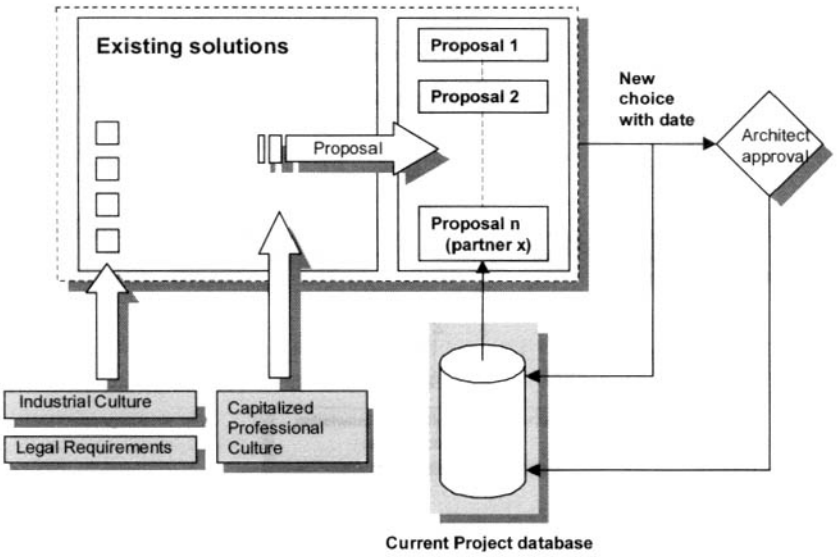


Figure 7. *Three viewpoints to ease decision-making*

Furthermore, we also ease these confrontations by first filtering specific expectations corresponding to the discipline or field of the end user:

- Functional expectations or features (mass, acoustic specifications, etc.),
- Cost constraints,
- Design (geometry, colour, etc.),
- Construction (specific legal requirements).

Each viewpoint allows the amount of data picked up to be reduced.

3.2. *Professional approaches*

We have seen that it is necessary to take partner choices and decisions into account by integrating their knowledge during elaboration processes. They also need to be able to make specific queries into the building database and to get a personal and updated representation of the building.

Professional representation of data

In France, with the statutory files related to construction specifications ('Cahier des Clauses Techniques Particulières', 'Cahier des Clauses Administratives Particulières', 'Dossier des Ouvrages Exécutés', etc.) a set of simple representations of the building is issued for each partner. The contents of these documents depend on the partner's needs.

So, we must be able to filter the entire pieces of information in order to deliver a specific answer to a particular partner request.

By anticipating the product descriptions we can access today through the network, the objective is to demonstrate that the specifications of a construction work may be extracted from an exhaustive set of information which use depends on professional needs (Celnik, Coste and Vincent, 1997).

This goal is very important, because answering to the multiple-expertise issues in a building context means that we must be able to give a 'real time' representation of the building to each partner. This representation is strictly made up from the information he needs to achieve his expert's report.

Professional profiles

The lack of normative data classification and the speed of their progression during the production process give a dynamic aspect to the building information. Consequently, sequenced representations of these data are soon out of date.

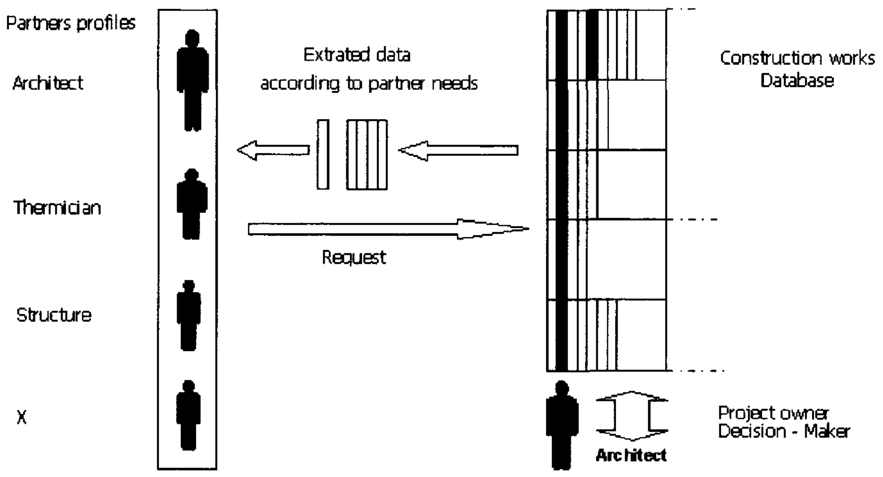

Partners profiles
Architect
Thermician
Structure
X
Extrated data according to partner needs
Request
Construction works Database
Architect
Project owner Decision - Maker

Figure 8. *The multi-actors aspect: specific information for each actor (Ameziane, 1998)*

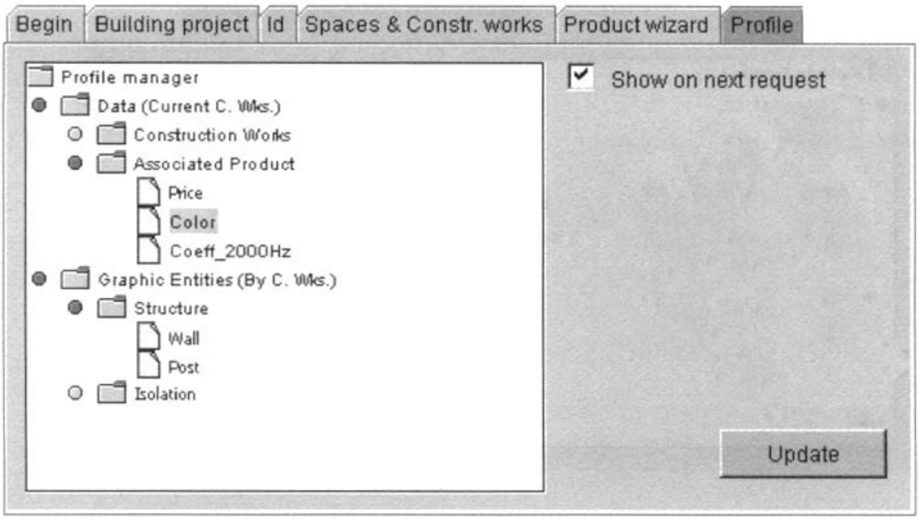
The architect being the actual database owner, he is responsible for coordinating the inputs from all the actors and updating the database, in order to provide the users with the latest and most accurate view of the building.

Figure 8 shows the construction works description multiple viewpoints based on existing databases and the various professional profiles associated to it.

Furthermore, these pieces of information are not really confidential for other partners. Our professional approach is only focused on the possibility of making them more understandable.

In that way, we are sure to improve the quality and the efficiency of the actor's expertise with the help of refined search.

We have worked on electronic representation of the files created during the design and engineering stages (Figure 9). It comes with personal information associated to a professional profile. Each actor is able to customise his default profile. He can select the data he wants to screen on the next request, as well as graphic representations. Each space comes as a dynamic 3D file that depends on the selected graphical entities. This interactive 3D representation uses the *virtual reality modelling language* (VRML).



Profile manager
● Data (Current C. Wks.)
○ Construction Works
● Associated Product
Price
Color
Coeff_2000Hz
● Graphic Entities (By C. Wks.)
● Structure
Wall
Post
○ Isolation

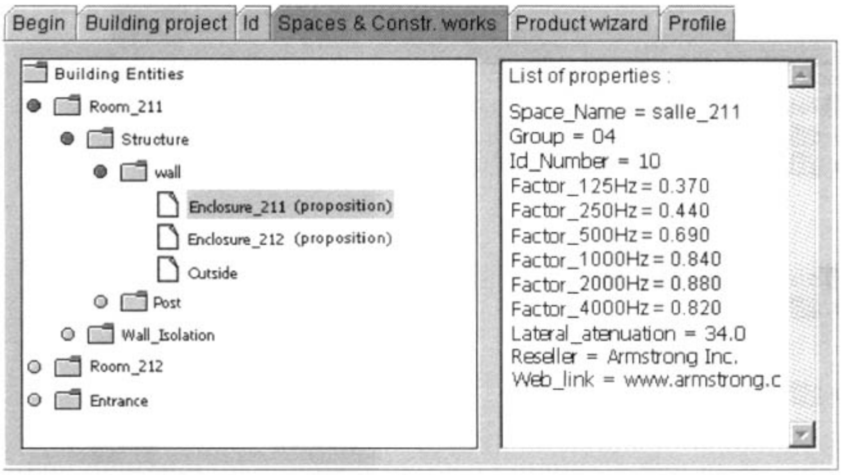
Figure 9. *Customise your profile*

3.3. *Queries and user guide*

At this stage, we suggest the development of an open request system for a set of experts whose search sequences are based on different viewpoints. We are focused on actors' experiences during the engineering stage because this project step provides most graphical and textual medias, exchanges and communications between partners.

Figure 10 shows that the structure of our query system may be seen as an incremental information research. It is sequenced by the following steps:

- query database identification,
- partner login,
- space choice,
- selection of a construction work associated to the space,
- information filtering mechanisms.



```
Building Entities
● Room_211
    ● Structure
        ● wall
            Enclosure_211 (proposition)
            Enclosure_212 (proposition)
            Outside
        ○ Post
    ○ Wall_Isolation
○ Room_212
○ Entrance
```

```
List of properties :
Space_Name = salle_211
Group = 04
Id_Number = 10
Factor_125Hz = 0.370
Factor_250Hz = 0.440
Factor_500Hz = 0.690
Factor_1000Hz = 0.840
Factor_2000Hz = 0.880
Factor_4000Hz = 0.820
Lateral_atenuation = 34.0
Reseller = Armstrong Inc.
Web_link = www.armstrong.c
```

Figure 10. *Incremental query scenario*

3.4. *Technology and tools*

To answer a collaborative context, we have based this sub-system on a web environment that allows one to manage good concurrent access. We chose the Oriented Object Language JAVA (Harold, 1997) that offers good opportunities in:

- Network mechanism implementation for client/server applications,
- Good security management,
- Cross platform applications,
- Database access through Java Data Base Connectivity and a three-party architecture that is independent from specific DBMS.

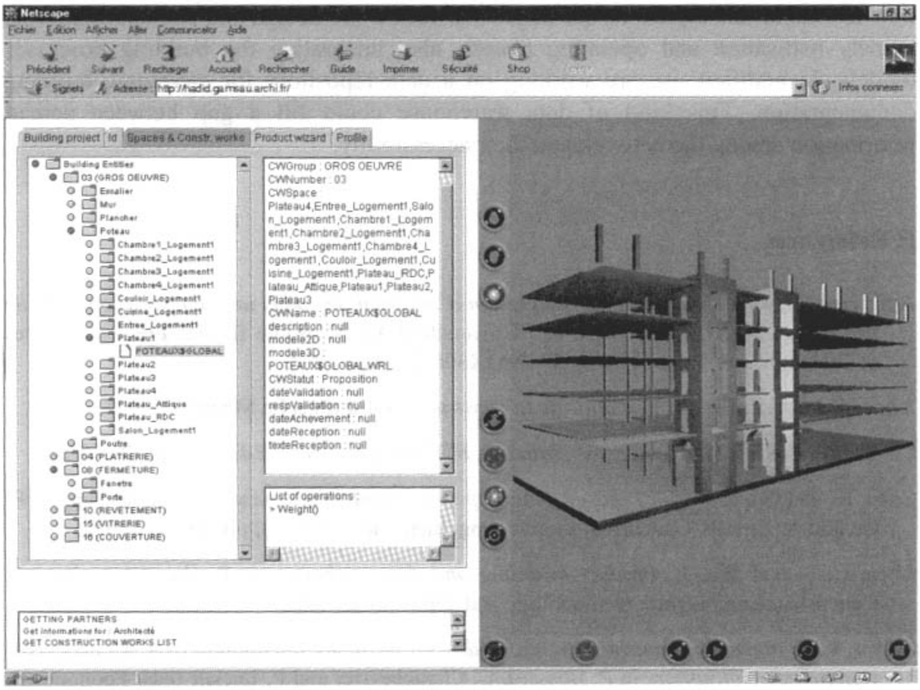
To easily maintain and update our information system, we found a good response in the powerful Lotus Notes client from Lotus Inc. The graphic files we use are:

- DWF (Drawing Web Format) from AutoDesk Inc. in order to publish vector drawings, and
- VRML (Virtual Reality Modelling Language) as a 3D graphic interface support.

3.5. *Graphical interface of the experimental system*

Each query sequence is stepped by 2D or 3D interactive graphics. In the building communication process, graphic drawings are the most used media because they are the only one building partners have in common. We want to increase interactivity by using them as interfaces to point out the right spaces or construction works.

The following figure shows the system status at the end of a query. Once the end user (e.g. the architect) has logged in, he gets his personal profile with an interactive 3D space representation. He may use it to select a specific construction work for which he gets a detailed description with all the corresponding attributes on a separate frame.



CWGroup : GROS OEUVRE
CWNumber : 03
CWSpace :
Plateau4,Entree_Logement1,Salon_Logement1,Chambre1_Logement1,Chambre2_Logement1,Chambre3_Logement1,Chambre4_Logement1,Couloir_Logement1,Cuisine_Logement1,Plateau_RDC,Plateau_Attique,Plateau1,Plateau2,Plateau3
CWName : POTEAUX$GLOBAL
description : null
modele2D : null
modele3D :
POTEAUX$GLOBAL.WRL
CWStatut : Proposition
dateValidation : null
respValidation : null
dateAchevement : null
dateReception : null
texteReception : null

List of operations :
- Weight()

GETTING PARTNERS
Get informations for : Architecte
GET CONSTRUCTION WORKS LIST

Figure 11. *System status at the end of a query Main user displaying construction works, geometry and textual descriptions*

4. Conclusions and prospects

Our work aims to:

- Ensure data circulation between distant partners,

- Be able to share a common description diagram in a DBMS environment,

- Manipulate heterogeneous documents (texts, drawings, multimedia files, etc.) in an efficient manner,

- Ensure professional approaches on a specific industrial product,

- Help the users by developing decision support systems.

Those results are important for our research centre because this project has been part of a CAD software leader research since January 2000 at Nemetscheck A.G. (Allplan, Allfa, palladio X, etc.). This firm wants to improve the links between the building CAD systems and the DBMS using the Internet for distributed partners to share graphical and non-graphical building information data.

Our work has been focused on the building product model. Our next objective is now to extend this system to the stages that follow the design and engineering steps, namely realisation and operating phases, also integrating the building processes models. In addition, we are working on a data repository for electronic building documentation. This kind of data warehouse could fill a gap between spread information among the network and their accessibility.

5. References

Ameziane, F., *Structuration et représentation d'informations dans un contexte coopératif de production du bâtiment*, Thèse de l'Université d'Aix-Marseille III, Faculté des Sciences et Techniques de Saint-Jérôme, N° 98AIX30012, Mars 1998.

Armand, J. and Raffestin, Y., *Guide de la construction*, Editions Le Moniteur, 1993.

Celnik, O., Coste E. and Vincent P., *Internet, BTP et architecture*, Editions Eyrolles, 1997.

Chan, K. and Gu, P., *A STEP-based generic product model for concurrent engineering*, in P. Gu and A. Kusiak: Concurrent Engineering: methodology and applications, 1993.

Chen, C. S. and Wu, J., *Product modeling and data exchange*, in P. Gu and A. Kusiak: Concurrent Engineering: methodology and applications, 1993.

Darses, F., *L'ingénierie concourante: un modèle en meilleure adéquation avec les processus cognitifs de conception*, in P. Bossard, C. Chanchevrier and P. Leclair (eds. Economica): Ingénierie concourante, de la technique au social, 1997.

Ghodous P. & Vandorpe D., *Advances in CONCURRENT ENGINEERING - CE 2000.* Technomic Publishing Company., Inc. Lancaster, Pennsylvania (USA), 2000.

Harold, E. R., *Programmation réseau avec JAVA*, Editions O'Reilly, 2000.

Belhi A., Erard P-J. & Bouras A., *Swiss Conference of CAD/CAM'99* – Proceedings Neuchâtel University, Switzerland, 1999.

Acknowledgements

The authors acknowledge M. Gunther Wildermuth Civil Engineering Manager of the Nemetschek Company, M. Bernard Giry Director of Nemetschek France and M. Michel Florenzano, Director of the UMR CNRS/MCC 694 MAP research laboratory for their support to the research program discussed in this paper. Special thanks for proof reading to Miss Carole Koch, English Teacher in the Building Production Engineering Master's Degree (DESS IPB) of the Architectural School of Marseilles.

This page intentionally left blank

Chapter 5

Agent-based Agile Manufacturing System Scheduling

David He and Astghik Babayan
Dept of Mechanical and Industrial Engineering, The University of Illinois at Chicago, USA

1. Introduction

Producing customised products in a short time at low cost is one of the goals of agile manufacturing. To achieve this goal in a manufacturing system, products are differentiated either by a machining-driven or an assembly-driven differentiation strategy [HE 96]. The successful implementation of these two strategies lies in efficient scheduling of the system. To react fast to changes in the market, agile manufacturing demands its manufacturing operations be distributed, its manufacturing system be modular, interactive, and robust, and the scheduling problems be solved fast. Yet, most existing scheduling systems are developed based on centralised structures, which makes manufacturing systems scheduling complicated. The purpose of applying agent-based approach to solve scheduling problems is to make the scheduling system easier to design and implement, more robust and less prone to errors, easier to use, faster, cheaper, and so on.

In this paper, agent-based approaches are applied to solve a complex scheduling problem in an agile manufacturing system. The structure of the manufacturing system that implements the product differentiation strategy in agile manufacturing is shown in Figure 1. It consists of two stages: machining and assembly, where there are m number of identical machines at the machining stage and q number of identical assembly machines at the assembly stage. The manufacturing system with this configuration is referred as $\{m, q\}$ manufacturing system.

To solve the scheduling problem, the representation of assembly sequence of a product produced in the system follows the *digraph* representation in [KUS 89]. In a digraph G, each node represents a part or a subassembly/assembly, and an arc represents a precedence relationship between two nodes. The level of assembly in a digraph is assigned as follows: value of 1 is assigned to the root node (assembly) and working backward from the root node, values of increment 1 are assigned to each subassembly node. Part nodes have the same assembly level as the corresponding subassembly or assembly nodes. Two products and the corresponding digraph representations of their assembly sequences are illustrated in Figure 2. The digraphs representing the assembly sequences of products can be classified into two types: simple digraph and complex digraph. A simple digraph is a digraph in which at most one subassembly node can be found at every assembly level (see Figure 2(a)). A simple digraph represents a linear assembly sequence of a product design. In a complex digraph, more than one subassembly node can be found on at least one assembly level (see Figure 2(b)). Throughout the paper, an assembly sequence of a product represented by a simple digraph is referred to as a *simple assembly sequence* and an assembly sequence of a product represented by a complex digraph is referred to as a *complex assembly sequence.*

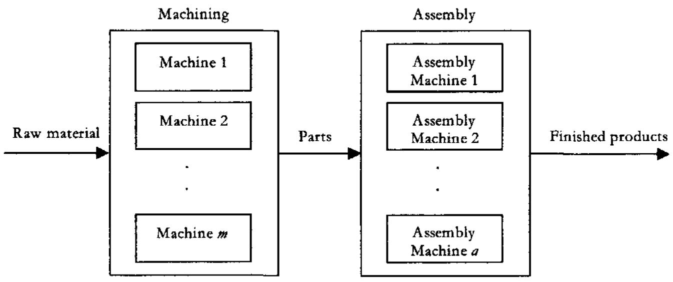


Figure 1. *General structure of {m, q} manufacturing system*

The scheduling problem to be solved in this paper is defined in the following: there are *m* number of identical machines at the machining stage and *q* number of identical assembly equipment at the assembly stage (see Figure 1). The objective of the scheduling problem is to assign parts and subassemblies/assemblies to the machines at the machining and assembly stages and determine the processing sequences on the machines so that the makespan ($C_{\max}$), i.e., the maximum completion time, is minimised.

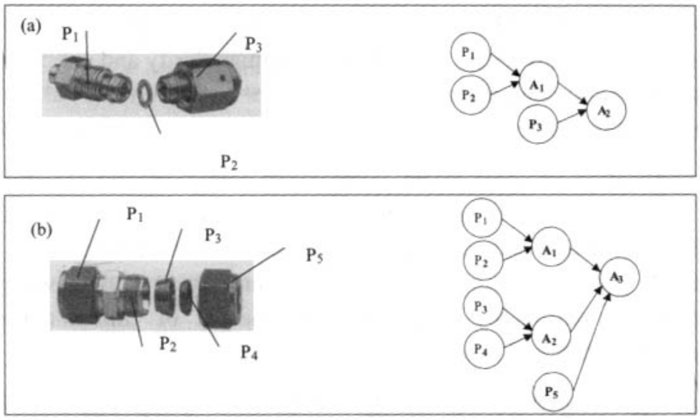


Figure 2. *Example of (a) a product and its simple digraph and (b) a product and its complex digraph*

2. Agent-based scheduling literature

The scheduling problem to be solved in this paper is considered as a combination of two classical scheduling problems with complex assembly sequence: two-machine flow shop scheduling problem and parallel machine scheduling problem. The two-machine flow shop scheduling problem can be solved by Johnson's algorithm [JOH 54]. Parallel machine scheduling problem ($P \| C_{\max}$) has been proved by [GAR 78] as *NP*–hard in the strong sense when the number of machines is unlimited. However, the problem is solvable in pseudo-polynomial time when the number of machines is fixed and thus NP–hard only in the ordinary sense. Most of the algorithms developed for solving $P \| C_{\max}$ are heuristics (e.g., [GRA 69], [COF 78], [FRI 86]). There are some problem characteristics that complicate solving scheduling problem: (*i*) precedence constraints between operations (machining/assembly); (*ii*) assignment of operations to machines; and (*iii*) sequence of unrelated (independent) operations (machining/assembly).

Scheduling problems have attracted various efforts in multi agent approaches (see for example [SOU 97], [RAB 94], etc.). The notion of agent was found in the wide range of research in *computer science* (CS), *distributed artificial intelligence* (DAI), etc. The multi agent systems paradigm represents one of the most promising approaches to the development of agile scheduling systems in manufacturing [RAB 99]. In fact, distributed systems have the following advantages [DEC 87]: (*i*) can simplify problem solving by splitting the problem into simple tasks; (*ii*) can tolerate uncertain data and knowledge; (*iii*) offer conceptual clarity and simplicity of design; (*iv*) present graceful degradation in computational complexity; (*v*) allow incremental modification of the system boundary; and (*vi*) suit well to distributed problems.

The application of agent-based approach to manufacturing scheduling is based on the idea that scheduling agility can be improved because of distributed autonomous systems and negotiation based decision-making in a multi-agent environment. [MIN 86] suggests the use of multi-agent system where there is a wide-range of reasonably self-contained pieces of functionality that are independently distributed in terms of timing and resources.

A survey by [SEN 99] reports 30 projects using agent technology for manufacturing planning, scheduling and execution control where agents represent physical entities, processes, operations, parts, etc. Real world examples of multi-agent application include: power systems management (see for [JEN 95]; [VAR 94]), particle accelerator control [JEN 93], telecommunications network management [WEI 94], spacecraft control [SCH 93], computer integrated manufacturing [PAR 95], job shop scheduling [MOR 93], etc. Assembly line design problem with agent application was studied by [SPR 95]. The design problem was solved by distributing overall systems among agents responsible for tasks, workstations, resource cells, etc. The coordination among distributed subsystems was achieved by negotiation and self organisation of agents. The objective of the design was minimisation of technological cost of the equipment that was achieved by applying simulated annealing.

A common definition for agent or "processor" was not found in the literature. In spite of this, [SIC 92] consider a "processor" as an agent when it possesses at least the following three properties: (*i*) has a certain degree of autonomy to reason about and to make decisions by itself; (*ii*) has the capability to interact with other agents; and (*iii*) has the knowledge to independently solve a part of the global problem.

An agent can possess various properties depending on the nature of the problem under consideration and the environment. The agents involved in the decision making process are not necessarily with same functionality. The level of functionality is defined in the problem decomposition process. Some of the agent properties found in the literature (*i.e.* [GEN 94], [GAL 88], [ROS 85]) are appropriate for the type of the agents used in our system. Here are the properties we believe our agents possess: *autonomy, sociability, reactivity, pro-activity, veracity, benevolence, and rationality.*

3. Agent-based scheduling framework

In this section, the general framework of the agent-based approach for scheduling in an agile manufacturing environment is presented. The successful implementation of agent-based approach strongly depends on how the system is decomposed into interrelated subsystems so that the integrity of the system is not violated. The digraph representation of product assembly sequence allows successful decomposition of large size scheduling problems into sub-problems, so that the integrity of original problem is not violated.

The digraph decomposition procedure described by [KUS 89] is used for decomposing the overall scheduling problem into sub-problems. The decomposition procedure consists in the following; a complex digraph is decomposed into simple sub-digraphs by removing root node A_i (A_3 in example presented by Figure 2(b)) from it. Then, moving forward we remove all the assembly nodes until the digraph is decomposed to the level where all resulting sub-digraphs have simple structure. Products with simple digraph representation have linear assembly structure and their optimal scheduling can be achieved by polynomial algorithms (see [HE 02]). As a result of decomposition, a set of sub-digraphs and single nodes (machining and assembly operations) is generated. Each decomposed sub-digraph, generated during the decomposition process, is an individual job that needs to be scheduled according to its precedence relationships with other jobs. For each sub-digraph an agent is assigned as an entity of the system that is responsible and authorised only for scheduling his job in the system. If a single assembly node generated during the decomposition process has immediate preceding part nodes all the part nodes immediately preceding to that assembly node are combined with that assembly node and assigned to an agent. As a result each agent will have at least one assembly operation to schedule. If designers make some changes in the product assembly structure, the sub-digraph corresponding to the redesigned component will be changed, hence the agent responsible for that component will take care of rescheduling. In this way the decomposition method, and the schedule obtained using agent-based approach is robust.

The autonomous nature of system agents also offers flexibility of decision making under different circumstances. However, like many other agent-based systems our system requires a significant monitoring overhead which is described next.

Besides the agents owning individual jobs, another type of agent, or so called *monitoring overhead* or *scheduling manager* (*SM*) is added to the decision making process in the system. Unlike the other agents in the system, scheduling manager is not responsible for scheduling activities. He/she is the decision-maker at a higher level and is responsible for choosing the winner among other agents during bidding process.

Before presenting agent-based scheduling algorithm, the following notations and definitions are introduced:

Define:

$J=(J_1, J_2, J_3, \ldots, J_k)$ the set of the jobs to be scheduled;

SAJ=set of schedulable jobs, i.e., set of the jobs that can be scheduled because all their preceding jobs are scheduled, or they have no preceding job;

NSJ=set of non-schedulable jobs, i.e., set of the jobs that cannot be scheduled because some of their preceding jobs have not been scheduled yet;

SJ=set of scheduled jobs.

Let ***stage*** be a step of the scheduling process, when a change in the set of schedulable/scheduled jobs occurs.

If we define t as the index of a stage, accordingly, the sets SAJ, NSJ, SJ will be defined for stage t as: SAJ_t, NSJ_t, and SJ_t.

At any stage t the following equation holds: $J=SAJ_t \cup NSJ_t \cup SJ_t$. Next, the agent-based scheduling algorithm is presented.

3.1. *Agent-based scheduling algorithm*

Step 1. Decompose the digraph into a set of simple digraphs and assign an agent to each subdigraph according to the decomposition order.

Step 2. Initialise the system by setting t=0 and defining sets SAJ_t, NSJ_t, and SJ_t.

Step 3. Schedule manager gives a signal to all scheduling agents, having job in the set of schedulable jobs (SAJ_t), to schedule their jobs in the system.

Step 4. The agent with the minimum makespan is the winner, and schedules his job.

Step 5. Set t=t+1 and update the sets SAJ_t, NSJ_t and SJ_t.

Step 6. *If* SJ_t=J *then* GoTo Step 7 *else* GoTo Step 3.

Step 7. End.

The individual jobs are scheduled by corresponding agents using appropriate scheduling algorithms developed in the literature. [HE 02] showed that products with simple/linear assembly structure can be scheduled with the same makespan on $\{m, q=1\}$ and $\{m, q>1\}$ systems. Since each agent possesses a job with simple assembly structure, mixed integer programming can be used to achieve optimal scheduling of individual jobs (see [HE 01] for formulation).

4. The negotiation model for agent-based scheduling

In this section, the model of negotiation among scheduling agents is presented. The purpose of negotiation is to reach an agreement about the assignment of operations to machining and assembly equipment and to improve the overall schedule. The model defines a range of strategies and tactics that an agent can deploy to generate new scheduling offers. The model is based on computationally tractable algorithms.

The following assumptions are valid for the model: (*i*) there is no deadline for the scheduling decisions to be reported to schedule manager; (*ii*) each agent starts decision-making process, as soon as he receives the necessary information from the schedule manager; (*iii*) agents report about their decisions to the schedule manager as soon as their decision is made; (*iv*) schedule manager announces his decision to the agents as soon as his decision is made.

Before describing the negotiation mechanism the following definitions and notations are introduced:

a_j	agent j, j=1, ..., k;

t_{ij}^{t} completion time of the last machining operation on machine i (i=1, ..., m) obtained by agent a_j (j=1, ..., k) at stage t;

T_{ij}^{t} completion time of the last assembly operation on assembly machine i, (i=1, ..., q) obtained by agent a_j (j=1, ..., k) at stage t;

$Sa_j^t = (t_{1j}^t, t_{2j}^t, ..., t_{mj}^t, T_{1j}^t, T_{2j}^t, ..., T_{qj}^t)$ completion time-vector of schedule Sa_j^t obtained by agent a_j (j=1, ..., k) at stage t;

$C_{\max}(j,t) = \max\{t_{1j}^t, t_{2j}^t, ..., t_{mj}^t, T_{1j}^t, T_{2j}^t, ..., T_{qj}^t\}$ maximum completion time of the schedule obtained by agent a_j (j=1, ..., k) at stage t;

At stage t, each agent schedules his job in the $\{m, q\}$ system and obtains $(m+q)$ completion time-vector Sa_j^t. This $(m+q)$ vector is being reported by each agent in SAJ_t to SM. Based on this information, SM makes decision and identifies who schedules his job on the system at the current stage. The value of the following expression is evaluated:

$$C_{\max}(t) = \min_j\{C_{\max}(j,t)\}$$

Agent a_j is the winner at stage t if the following is true:

$$C_{\max}(j,t) = C_{\max}(t)$$

Once an agent becomes the winner at one stage, he enters the negotiation process with certain input. The players in the negotiation process are all the agents already having scheduled their jobs in the system. Basically the input brought by an agent to the negotiation process is the values of the decision variables that uniquely define the total schedule obtained that far. Once the winner is identified (or new player entering the negotiation process is identified) the set of *schedulable jobs* (*SAJ*), the set of *scheduled jobs* (*SJ*), and the set of non-schedulable jobs (*NSJ*) are updated. The current stage index is incremented by one. The winner, the agent who is scheduling his job at the current stage, observes his schedule, and decides if he is going to request for negotiation from the agents in the negotiation process. Listed below are some conditions that indicate the need of negotiation.

– The makespan obtained exceeds the estimated upper bound on makespan. In this case the agent last entered the negotiation process initiates a negotiation with others.

– Even if the makespan does not exceed the estimated upper bound it is close enough to that.

– If for any assembly operation scheduled by an agent starting time is greater than the completion time of its preceding operations (machining and assembly).

Basically this fact indicates that the agent is delaying his job because of other agents. Hence, he may consider a negotiation with others.

Suppose agent a_1 was the winner at stage t=1, and agent a_2 becomes the winner at stage t=2. The overall schedule obtained so far is the combination schedule of the jobs owned by agent a_1 and agent a_2. Let's assume that for some reason agent a_2 initiates a negotiation process in order to improve the schedule. As a result of negotiation, a series of feasible schedules is generated by combined efforts of agent a_1 and agent a_2. The result is represented in the matrix presented below:

$$\begin{bmatrix} C_{\max}(\hat{1},1) & \times & \times & \times & \times & \times \\ C_{\max}(2,\hat{1}) & C_{\max}(\hat{2},2) & \times & \times & \times & \times \\ \times & C_{\max}(3,\hat{2}) & C_{\max}(\hat{3},3) & \times & \times & \times \\ \times & \times & C_{\max}(4,\hat{3}) & C_{\max}(\hat{4},4) & \times & \times \\ \times & \times & \times & C_{\max}(5,\hat{4}) & C_{\max}(\hat{5},5) & \times \\ \times & \times & \times & \times & \Lambda & \Lambda \end{bmatrix}$$

The elements of the matrix are maximum completion time of the combination schedule (job 1 and job 2). Agent a_2 enters the negotiation process, where agent a_1 is a player with some schedule (defined as schedule 1 of agent a_1), with some initial schedule (defined as schedule 1 of agent a_2).

When agent a_2 enters the negotiation process, the schedule offered by him is the best that he is able to find with consideration of decision variables defined at stage 1 by agent a_1.

The first element $C_{\max}(\ddot{1},1)$ of the matrix is the maximum completion time that agent a_2 obtained when he entered the negotiation process. The *cup* sign above number 1 (which indicates the schedule 1 by agent a_1) indicates that agent a_2 obtained schedule 1 without changing the values of decision variables defined by agent 1 in schedule 1. From this point on, a series of feasible schedules is obtained as described. Consider the element $C_{\max}(2,\ddot{1})$. This is the maximum completion time that was obtained by agent a_1 (at this point it is considered as schedule 2 by agent a_2), where the cup above 1 indicates that agent a_1 obtained schedule 2 without changing the values of decision variables defined by agent a_2 in schedule 1. The rest of the matrix is constructed similarly.

At some point, the negotiation process should be terminated. The following conditions are recommended to be checked for termination:

– the *current maximum completion time* is equal to the estimated lower bound on makespan, which means the solution obtained is optimal;

– agents involved in the negotiation process are satisfied by the results obtained;

Once the negotiation process is terminated, the schedule obtained is passed to *SM* and *SM* announces about it to the agents in the *SAJ* list.

The best schedule obtained is the one corresponding to the element of the matrix that satisfies the following condition:

$$\min_i \left\{ \min_j \{ C_{\max}(i, j) \} \right\}$$

This schedule is reported to the schedule manager. The output schedule at stage t represents the input for stage $t+1$. This means that agent a_1 and agent a_2 will participate in the scheduling process at stage $t=3$ with the corresponding schedules, obtained as a result of negotiation process carried out at stage $t=2$, numbered as 1.

The scenario will change once agent a_3 enters the negotiation process. Now agent a_3 negotiates with agent a_1 and agent a_2 together, since they both have separate units to control in the schedule. And each one is responsible and authorised for changes concerning to the job's schedule owned by him. However, starting from stage $t=3$ the negotiation process keeps same tactics as at stage $t=3$ which is described below.

As the process continues, agent a_1 becomes the winner at stage $t=1$, agent a_2 becomes the winner at stage $t=2$, and agent a_3 becomes the winner at stage $t=3$. The overall schedule obtained so far is the combination schedule of the jobs owned by agent a_1, agent a_2, and agent a_3. In order to improve the schedule after entering the scheduling process, agent a_3 initiates a negotiation process with agent a_1 and agent a_2. Each of them tries to improve the schedule by rescheduling the corresponding schedules they obtained as an output of stage $t=2$. The matrixes corresponding to negotiation processes are:

$$\begin{array}{cc} & \text{Agent } a_3 \\ \text{Agent } a_1 & \begin{bmatrix} C_{\max}(\hat{1},\hat{1},1) & \times \\ C_{\max}(2,\hat{1},\hat{1}) & \times \end{bmatrix} \end{array} \qquad \begin{array}{cc} & \text{Agent } a_3 \\ \text{Agent } a_2 & \begin{bmatrix} C_{\max}(\hat{1},\hat{1},1) & \times \\ C_{\max}(\hat{1},2,\hat{1}) & \times \end{bmatrix} \end{array}$$

By comparing $C_{\max}(2,\hat{1},\hat{1})$ and $C_{\max}(\hat{1},2,\hat{1})$ one can find out who generated better schedule. The following three cases are possible:

– $C_{\max}(2,\hat{1},\hat{1}) \geq C_{\max}(\hat{1},2,\hat{1})$ and $C_{\max}(2,\hat{1},\hat{1}) > C_{\max}(\hat{1},\hat{1},\hat{1})$. In this case agent a_2 becomes the winner and he gets the right to initiate the next negotiation process.

– $C_{\max}(\ddot{1},2,\ddot{1}) \geq C_{\max}(2,\ddot{1},\ddot{1})$ and $C_{\max}(\ddot{1},2,\ddot{1}) > C_{\max}(\ddot{1},\ddot{1},\ddot{1})$. In this case agent a_2 becomes the winner and he gets the right to initiate the next negotiation process.

– $C_{\max}(2,\ddot{1},\ddot{1}) \leq C_{\max}(\ddot{1},\ddot{1},\ddot{1})$ and $C_{\max}(\ddot{1},2,\ddot{1}) \leq C_{\max}(\ddot{1},\ddot{1},\ddot{1})$. None of them is a winner and the negotiation process is over at the current stage. The stage index is incremented by one.

Suppose agent a_1 is the winner. He initiates a negotiation with agent a_2 and agent a_3. Each of them tries to improve the current overall schedule. At the current moment the overall makespan obtained is $C_{\max}(1,1,1)$.

$$\begin{array}{cc} & \text{Agent } a_1 \\ \text{Agent } a_2 & \begin{bmatrix} C_{\max}(\hat{1},\hat{1},1) & \times \\ C_{\max}(\hat{1},2,\hat{1}) & \times \end{bmatrix} \end{array} \qquad \begin{array}{cc} & \text{Agent } a_1 \\ \text{Agent } a_3 & \begin{bmatrix} C_{\max}(\hat{1},\hat{1},1) & \times \\ C_{\max}(\hat{1},\hat{1},2) & \times \end{bmatrix} \end{array}$$

Considering three corresponding possible cases discussed at the previous iteration the winner is identified and the next iteration is started, or the stage counter t is incremented by one. Eventually the last agent will enter the scheduling process and after necessary negotiations the final schedule will be obtained.

Next, the formal agent-based scheduling algorithm with negotiation mechanism is presented. In addition to the notations and definitions prior introduced the counter of iteration is defined by c.

4.1. *Agent-based scheduling algorithm with negotiation*

Step 1. Decompose digraph into a set of simple digraphs and assign an agent to each job according to the decomposition order.

Step 2. Initialise the system by setting t=0 and defining sets SAJ_t, NSJ_t, and SJ_t.

Step 3. Schedule manager gives a signal to all scheduling agents, having job in the set of schedulable jobs (SAJ_t), to schedule their jobs in the system.

Step 4. The agent who generates the shortest combined makespan with the scheduled jobs is the winner and schedules his job in the system.

Step 5. Set t=t+1, c=0 and update the sets SAJ_t, NSJ_t and SJ_t.

Step 6. The last agent entering the scheduling process checks if negotiation is necessary. If so, he initiates a negotiation with the other agents in the scheduling process. The counter of iteration is set c=c+1. A series of feasible schedules is generated with each agent in the scheduling process, and makespan matrixes are constructed.

If no negotiation is conducted *then* GoTo Step 3.

Step 7. The agent generating the minimum makespan with the agent last entering the scheduling process is the winner in that iteration.

Set *last agent entering the scheduling process = winner at iteration c.*

Step 8. Check for termination of negotiation.

If negotiation is terminated GoTo Step 9 *else* GoTo Step 6.

Step 9. *If* $SJ_t=J$ *then* GoTo Step 10 *else* GoTo Step 3.

Step 10. End.

Note that the decomposition in Step 1 is performed by the digraph decomposition algorithm described in [KUS 89].

An illustrative example of agent-based scheduling algorithm *with* negotiation is presented in Appendix I.

5. Objective measures of effectiveness

In assessing the benefits of agent-based scheduling in terms of solution quality one should notice that solution quality strongly depends on the scheduling techniques used by individual agents for scheduling their tasks. Also note that each subsequent sub-stage in a negotiation process gives a better output than its previous sub-stage. Thus, the longer the negotiation process lasts the better solutions will be obtained. Next, lower and upper bound estimates on makespan are developed.

Suppose a product with corresponding digraph is going to be scheduled on a $\{m, q\}$ manufacturing system.

Let P_1, P_2, ..., P_n be all part nodes in the digraph, and A_1, A_2, ..., A_N be all assembly nodes in the digraph (see example in Figure 2). The following notations are introduced:

k = number of agents or decomposed sub-digraphs.

n_i = number of part nodes in the sub-digraph corresponding to the sub-problem handled by agent a_i.

N_i = number of assembly nodes in the sub-digraph corresponding to the sub-problem handled by agent a_i.

$P_1^i, P_2^i, ..., P_{n_i}^i$ are part nodes in sub-digraph corresponding to the sub-problem handled by agent a_i.

$A_1^i, A_2^i, ..., A_{N_i}^i$ are assembly nodes in sub-digraph corresponding to the sub-problem handled by agent a_i.

$t^i_{(1)} \le t^i_{(2)} \le \ldots \le t^i_{(n_i)}$ are ordered processing times corresponding to part nodes $P^i_1, P^i_2, \ldots, P^i_{n_i}$.

$T^i_{(1)} \le T^i_{(2)} \le \ldots \le T^i_{(N_i)}$ are ordered processing times corresponding to assembly nodes $A^i_1, A^i_2, \ldots, A^i_{N_i}$.

Then lower bound on makespan is calculated using the above defined terminology by equation [1]. Basically, what it represents is the best-case performance of the scheduling, when assembly starts right after the shortest machining time in the digraph and after that there is no idle time on assembly line.

$$LB = \min_{i=1}^{k}\{t^i_{(1)}\} + \frac{\sum_{i=1}^{N-1} t(A_i)}{q} + t(A_N) \qquad [1]$$

For developing upper bound on makespan the following notations are introduced. Let $\lceil(\bullet)\rceil$ be the smallest greater or equal integer of evaluated expression $(\bullet)$, then define $k_i = \left\lceil\left(\frac{n_i}{m}\right)\right\rceil$ as average number of parts per machine for agent a_i and $r = \left\lceil\left(\frac{k}{q}\right)\right\rceil$ average number of agents/individual jobs per assembly line.

The maximum completion time for the schedule by agent a_i can be computed using the following formula $T_i = \sum_{j=0}^{k_i-1} t^i_{(n_i-j)} + \sum_{j=1}^{N_i} T^i_j$. The expression $T^i = \sum_{j=1}^{N_i} T^i_j$ represents total assembly time corresponding to the job possessed by agent a_i. The expression $\sum_{j=0}^{k_i-1} t^i_{(n_i-j)}$ represents the machining time, when first k_i largest machining operations in the job possessed by agent a_i are assigned to the same machine, which corresponds to the worst possible performance. Then, T_i is the completion time, when assembly operations start right after all machining is completed.

The ordered total assembly time corresponding to each agent is $T^{(1)} \le T^{(2)} \le \ldots \le T^{(k)}$.

Using these notations and formulations the upper limit of total completion time is calculated by equation [2].

$$UB = \sum_{i=1}^{k} \sum_{j=0}^{k_i-1} t^i_{(n_i-j)} + \sum_{j=0}^{r-1} T^{(k-j)}, \text{ where } t^i_{(n_i-j)} = \begin{cases} t^i_{(n_i-j)}, \text{if } n_i - j \ge m; \\ t^i_{(n_i)}, \text{if } n_i - j < m \end{cases} \qquad [2]$$

5.1. *Computational experiments*

To show the effectiveness of the proposed agent-based scheduling methodology in terms of solution quality computational experiments were conducted. We chose the system of consisting of one machine and one assembly station because optimal algorithms for scheduling on these systems were developed by [KUS 89] and we were able to compare the results obtained by applying agent-based approach with the optimal scheduling solutions.

The developed agent-based approach was coded in C++ and used for testing problems. A total of 4 types of problem with different assembly sequences were tested. For each testing problem, 5 instances were generated, and for each instance, assembly sequence, number of part nodes, number of subassembly nodes, maximum level of assembly, machining times, and assembly times were randomly generated. The data was generated based on the real assembly application information from industrial assembly handbooks ([NOF 96]; [BOO 92]; and [LOT 89]). The average size of the problems corresponds to 37 part nodes, 26 assembly nodes, and 7 assembly levels. The machining and assembly times were randomly generated that follow uniform distribution U(7, 25) and U(10, 30) respectively for the first type of the problems. For the second 5 instances of the problems the generated machining and assembly times follow uniform distribution U(6, 14) and U(6, 19) respectively. The third sample followed U(3, 12) and U(4,12). And finally, for the fourth sample we had U(5,16) and U(6,20) correspondingly. The results of the computations are provided in Table 1.

Table 1. *Results of the computational experiment*

Problem instance	No.	$C_{\max}$	C_{AB}	$C_{AB\text{-}N}$	$C_{AB}/C_{\max}$	$C_{AB\text{-}N}/C_{\max}$
Machining times: U(7, 25) Assembly Times: U(10, 30)	1	683	704	683	1.031	1.000
	2	638	641	638	1.005	1.000
	3	631	714	631	1.132	1.000
	4	582	640	598	1.100	1.027
	5	585	619	605	1.058	1.034
Machining times: U(6, 14) Assembly Times: U(6, 19)	1	404	434	422	1.074	1.045
	2	399	440	399	1.103	1.000
	3	392	417	407	1.064	1.038
	4	391	437	391	1.118	1.000
	5	405	443	408	1.094	1.007
Machining times: U(3, 12) Assembly Times: U(4, 12)	1	320	335	321	1.047	1.003
	2	335	345	337	1.030	1.006
	3	329	342	330	1.040	1.003
	4	301	317	310	1.053	1.030
	5	315	326	323	1.035	1.025
Machining times: U(5, 16) Assembly Times: U(6, 20)	1	487	504	502	1.035	1.031
	2	435	442	434	1.016	1.000
	3	443	452	448	1.020	1.011
	4	432	453	443	1.049	1.025
	5	464	492	478	1.060	1.030

To evaluate the effect of negotiation in agent-based scheduling, for each problem we tested, the completion time of product was obtained without applying negotiation. The results are presented in Table 1. The notations in Table 1 are as follows:

$C_{\max}$ – completion time corresponding to optimal schedule;

C_{AB} – completion time obtained by applying agent-based approach *without* negotiation;

$C_{AB\text{-}N}$ – completion time obtained by applying agent-based approach *with* negotiation;

The ratios in Table 1, i.e., $C_{AB}/C_{\max}$, $C_{AB\text{-}N}/C_{\max}$, represent the effectiveness of the tested agent-based approaches.

Comparing the results in ratio columns corresponding to agent-based approach with and without application of negotiation, one can see that application of negotiation dramatically improves the solution quality. Agent-based scheduling *without* applying negotiation provided 0.5%-13.2% deviation in completion time from the optimal schedule when agent-based scheduling *with* application of negotiation provided 0.0%-4.5% deviation in completion time from the optimal schedule.

Concluding the experimental results, we can say that agent-based scheduling approach presented in this paper provides optimal and near optimal solutions.

6. Conclusions

In this paper, an agent-based approach was designed for manufacturing system scheduling in an agile manufacturing environment. The general framework of the agent-based approach was presented. The successful implementation of agent-based approach strongly depends on how the system is decomposed into interrelated subsystems and the negotiation among the agents so that the integrity of the system is not violated. A decomposition method was proposed for successful decomposition of large size scheduling problems into sub-problems. Then negotiation mechanism was developed and incorporated into decision-making process in a multi agent environment. A lower and upper bound on maximum completion time were.

The decomposition method and agent-based tool were designed in a way that each agent is assigned a separate individual job to be scheduled in the system according to the precedence relationships of other agents' jobs. If any changes are made by users in component designs, or assembly structures the product designs, rescheduling can be done easily by corresponding agents in the scheduling system. In this sense, the schedule obtained by agent-based approach is robust. The developed approach integrates data and decisions associated with several entities within a scheduling system. This approach can be used to model and solve large-scale scheduling problems in an agile manufacturing environment. To demonstrate the effectiveness of proposed methodology, computational experiments were conducted. The results of tested problems show that the scheduling method obtains optimal or near optimal solutions.

7. References

[BOO 92] BOOTHROYD, G., *Assembly Automation and Product Design*, Marcel Dekker, New York, 1992.

[COF 78] COFFMAN E. G., GAREY M. R. AND JOHNSON D. S. "An application of bin-packing to multiprocessor scheduling", *SIAM Journal of Computing*, vol. 7, 1978, p. 1–17.

[DEC 87] DECKER K. S., "Distributed problem solving techniques: a survey", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC–17, no. 5, 1987, p. 729–740.

[FRI 86] FRIESEN D. K., LANGSTON M. A., "Evaluation of a multifit-based scheduling algorithm", *Journal of Algorithms*, vol. 7, 1986, p. 35–59.

[GAL 88] GALLIERS J. R., (1988). “A theoretical framework for computer models of cooperative dialogue. Acknowledging multi-agent conflict”, *PhD thesis*, Open University, UK.

[GAR 78] GAREY M. R., JOHNSON D. S., “Strong NP–completeness results: motivations, examples and implications”, *Journal of Association of Computing and Mathematics*, vol. 25, 1978, p. 499–508.

[GEN 94] GENESERETH M. R., KETCHPEL S. P., “Software agents”, *Communications of the ACM*, vol. 37, no. 7, 1994, p. 48–53.

[GRA 69] GRAHAM R. L., “Bounds on multiprocessing timing anomalies”, *SIAM Journal of Applied Mathematics*, vol. 17, 1969, p. 416–429.

[HE 02] HE D., BABAYAN A., “Scheduling manufacturing systems for implementation of delayed product differentiation strategy for agile manufacturing”, *International Journal of Production Research*, 2002 (to appear).

[HE 01] HE D., BABAYAN A., KUSIAK A., “Scheduling manufacturing systems in an agile manufacturing environment”, *Robotics and Computer Integrated Manufacturing*, vol. 17, 2001, p. 87–97.

[HE 96] HE D., KUSIAK A., “Performance analysis of modular products”, *International Journal of Production Research*, vol. 34, no. 1, 1996, p. 253–272.

[JEN 95] JENNINGS N. R., CORERA J., LARESGOITI I., MAMDANI E. H., PERRIOLAT F., SKAREK P., VARGA L. Z., “Using ARCHON to develop real word DAI applications for electricity transportation management and particle accelerator control”, *IEEE Expert* – Special Issue on Real World Applications of DAI, 1995.

[JEN 93] JENNINGS N. R., VARGA L. Z., AARNTS R. P., FUCHS J, SKAREK P., “Transforming standalone expert systems into a community of cooperating agents”, *International Journal of Engineering Applications of Artificial Intelligence*, vol. 6, no. 4, 1993, p. 317–331.

[JOH 54] JOHNSON S. M., “Optimal two and three-stage production schedule with setup times included”, *Naval Research Logistics Quarterly*, vol. 1, no. 1, 1954, p. 61–68.

[KUS 89] KUSIAK A., “Aggregate scheduling of a flexible machining and assembly system”, *IEEE Transactions on Robotics and Automation*, vol. 5, no. 4, 1989, p. 451–459.

[LOT 89] LOTTER B., *Manufacturing Assembly Handbook*, Butterworth, London, 1989.

[MIN 86] MINSKY M., *The Society of Mind*, Simon and Schuster, New York, NY, 1986.

[NOF 96] NOF S.Y. WILHELM W. E., WARNECKE H.-J., *Industrial Assembly*, Chapman Hall, New York, NY, 1996.

[PAR 95] PARUNAK H.V.D., “Applications of distributed artificial intelligence in industry”, In *Foundations of Distributed Artificial Intelligence*, editors G.M.P. O’HARE and N.R. JENNINGS, Wiley, 1995.

[RAB 94] RABELO R.J., CAMARINHA-MATOS L.M., “Negotiation in multi-agent based dynamic scheduling”, *Journal on Robotics and Computer Integrated Manufacturing*, vol. 11, no. 4, 1994, p. 303–310.

[RAB 99] RABELO R.J., CAMARINHA-MATOS L.M. AND AFSARMANESH H., "Multi-agent-based agile scheduling", *Robotics and Autonomous Systems*, vol. 27, 1999, p. 15–28.

[ROS 85] ROSENSCHEIN J.S., GENESERETH M.R., "Deals among rational agents", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI–85),* 1985, p. 91–99, Los Angeles, CA.

[SCH 96] SCHWUTTKE U.M., QUAN A.G., "Enhancing performance of cooperating agents in real time diagnosis systems", *Proceedings of. 13th International Joint Conference on Artificial Intelligence,* Chamberry, France, 1993, pp 332–337.

[SEN 99] SHEN W., NORRIE D.H., "Agent-based systems for intelligent manufacturing: A state of the art survey", *Knowledge and Information Systems, an International Journal,* vol. 1, no. 2, 1999, pp. 129–156.

[SIC 92] SICHMAN J., DEMAZEAU Y., "When can knowledge-based systems be called agents?", *Proceedings IX Brazilian Symposium on Artificial Intelligence,* Rio de Janeiro, Brazil, 5–8 October, 1992.

[SOU 97] SOUSA P., RAMOS A., "A dynamic scheduling Holos for manufacturing systems" *Proceedings of the Second World Congress on Intelligent Manufacturing Processes and Systems*, Budapest, Hungary, 10–13 June, 1997.

[SPR 95] SPRUMONT F., GHEDIRA K, MÜLLER J.-P., "AMACOIA: a multi agent approach to the design of flexible lines: Preliminary report", *Proceedings of IJCAI Workshop on Intelligent Manufacturing Systems*, Montreal, Canada, 1995.

[WEI 94] WEIHMAYER R., VELTHUIJSEN H., "Application of distributed AI and cooperative problem solving to telecommunications", In *AI Approaches to Telecommunications and Network Management*, editors J. Liebowitz and D. Prereau, IOS Press, 1994.

Appendix I: Illustrative example of agent-based scheduling algorithm

Consider a product to be produced in $\{m=2, q=1\}$ manufacturing system. The assembly sequences of the product is shown in Figure 3. The machining and assembly times are provided in Table 2.

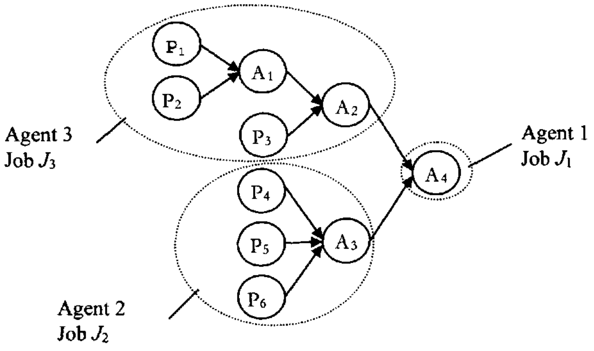


Figure 3. *Assembly structure of a product*

Table 2. *Machining and assembly times*

Part	P_1	P_2	P_3	P_4	P_5	P_6
Machining Time	6	4	6	1	6	9
Assembly	A_1	A_2	A_3	A_4	–	–
Assembly Time	3	4	3	5	–	–

To decompose the digraph, we remove the root node A_4. All the resulted sub-digraphs have simple assembly structure. After decomposition, an agent is assigned to each job. The set of following jobs is defined: $J=\{J_1, J_2, J_3\}$.

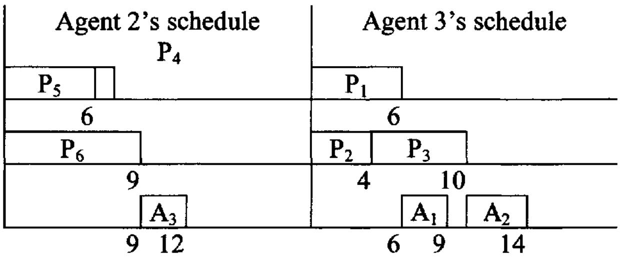
Initialise the system:

$t=0$ $SAJ_0=\{J_2, J_3\}$

$NSJ_0=\{J_1\}$

$SJ_0=\{\varnothing\}$

At this stage the schedule manager gives a signal to the agents of all the schedulable jobs to find the best possible schedule (shortest makespan) with the consideration of scheduled jobs (the set of scheduled jobs is empty at $t=0$). The Gantt charts of schedules obtained by individual agents having job in SAJ_0 are provided below.



Agent a_2 is the winner, since his schedule has the shortest makespan.

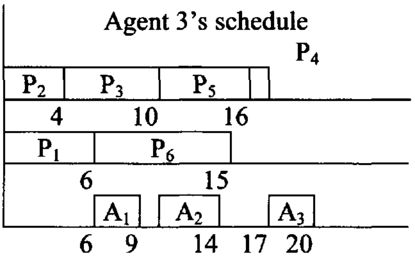
The system is updated as follows:

t=1 $SAJ_1=\{J_3\}$

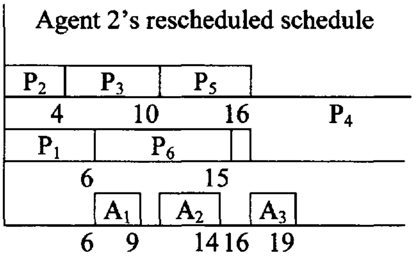
$NSJ_1=\{J_1\}$

$SJ_1=\{J_2\}$

The Gantt chart of schedule obtained by the only agent having job in SAJ_1 is provided below.



Agent a_2 realises that some changes happened to his schedule; the completion times of operations were changed. He reschedules his schedule without changing any assignment made by agent a_3. The final schedule is shown below.



Here it is reasonable to terminate negotiation process, since the total machining time was broken evenly between 2 machines.

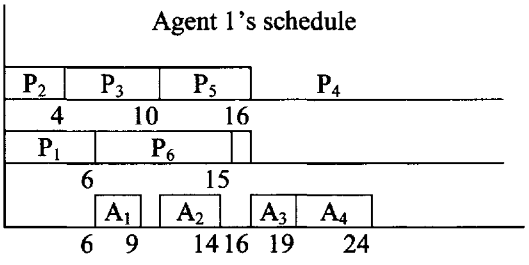
Update the system as follows:

t=2 $SAJ_2=\{J_1\}$

$NSJ_2=\{\varnothing\}$

$SJ_2=\{J_2, J_3\}$

The schedule manager gives a signal to all of the schedulable jobs' agents to find the best possible schedule with the consideration that the jobs in SJ_2 are scheduled. The last scheduling agent a_1 schedules his job and obtains the schedule presented below.



Scheduling agents a_2 and a_3 do not find any change in their scheduled operations completion times, hence there is no need for rescheduling.

Update the system as follows:

t=1 $SAJ_3=\{\varnothing\}$

$NSJ_3=\{\varnothing\}$

$SJ_3=\{J_1, J_2, J_3\}$

Since $SJ_3=J=\{J_1, J_2, J_3\}$, the scheduling process is over.

This page intentionally left blank

Chapter 6

New Product Development within a Concurrent Engineering Environment: Knowledge and Software Tools

Jean-Louis Selves, Eric Sanchis and Zhaoyang Pan
IUT Ponsan, Paul Sabatier University, Toulouse, France

1. Introduction

New product development processes has led us to consider the design project management which is associated with two goals: on one hand the organisational and technical search for solutions, on the other hand the clarification of procedures and necessary means of setting up the project. These goals are to reached within the framework of a process, which can be characterised by two approaches: the traditional sequential approach and the concurrent engineering approach [MID 97]. The traditional approach, in which the project proceeded sequentially from phase to phase [TAK 86] (concept development, feasibility testing, product design, development process, pilot production and final production) and from functional team to functional team, was largely used in manufacturing industry but may conflict with today's competitive requirements: speed and flexibility. The latter approach, where the overlaps extend over several phases, is characterised by a project team that tries to develop new products using co-operative work and can respond more quickly to changing market conditions [CHA 97, TAK 86, CAR 91, SAR 97].

This processes can be described in two ways:
- modelling the whole process using the perspectives offered by the domain of problem solving and knowledge needs [SIM 91, NEW 72],
- concentrating on the identification and the analysis of the tracks and the evolution of the intermediate design objects (IDO) following A. Jeantet and J.F. Boutut [JEA 98]. They noticed that "the designers spend most of their time to create, to manipulate, to discuss, to interpret, to estimate, to transform, ... texts, graphs, calculations, computer models, drawings, physical mock-up ... so much that, to understand the design process, it seemed to us necessary to give to these objects, in our approach of analysis, a place as central as the one that they occupy in the activity of the designers. That is why, we decided to create a generic category: *intermediate design objects*. It is about objects produced or used during the action of designing, getting in touch tools, procedures and actors. We make the hypothesis that these objects are tracks of the design hypothesis, the realised compromises and the decisions made during the design process".

Following these two descriptions, the performance of the team project to develop new products in the CE environment will depend essentially on the speed with which its members are going to acquire, to look for or to share knowledge needed in design problem solving, taking into account the strategy of the company. It will be objective to favour the quick emergence of collective solutions by the means of the traffic of the intermediate design objects (IDO) related to the explicit knowledge of the team. Consequently, the goal of our research will be, first, to examine a company using CE to define knowledge needed in design problem-solving and the methods used. Then we will classify problems and describe the different collective approaches to solving them. Finally, taking into account the nature of problems and mechanisms to solve them, we will propose software tools based on software agents

[FER 95] and a model of document linked to the IDO to help in designing in a context of a team working within the CE environment. This is qualified by Darse [DAR 97] vis a vis co-design process and distributed design process.

2. Design problems within CE environment

Within a CE environment a multidisciplinary project team works to reach goals like quality, low price and maximum speed [GAU 97]. This working framework is particularly associated with the car industry. Thus, to understand a CE development process it seems natural to consider the TOYOTA company because it is considered to be a model for the concurrent engineering [WAR 95].

The vehicle development cycle is organised around key milestones [WAR 95]. The architecture is modular and each module corresponds to a subsystem studied by a functional group. That allows the members of the team project to work under a matrix organisation. For each module, the designers make their search for several solutions, which are explored and experimented first at subsystem level, and if needed, at the system level. The coherence of interfaces is ensured with a checklist of the constraints, which indicates limits not to be exceeded.

In this process we find the five phases defined by I. Nonaka and H. Takeuchi [NON 97]: the sharing of tacit knowledge, the creation of concepts, the justification of concepts, the construction of an archetype, the extension of the knowledge in the various levels of the organisation and we can distinguish four types of knowledge, which move during the project (targets which become technical specifications, constraints, graphical representations or solutions, results of tests or solution evaluations).

A schema which summarises the process of industrial problem solving [SEL 00, SEL 97, BUR 97, BUR 95] is shown in Figure 1. This schema takes into account knowledge used and dynamics of problem solving; difference between objectives and results producing a new cycle. It agrees with the classical modelling of problem solving [NEW 72, DAR 97] and the four types of knowledge that we have identified previously.

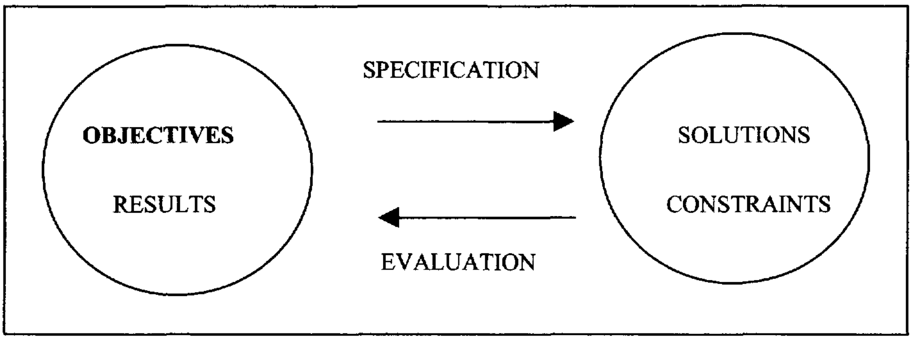


Figure 1. *Industrial problem solving*

Usually one expresses objectives in a static way; these are references from which knowledge making in industrial problems solving progresses. In design problems, objectives are not known exactly at the beginning [DAR 97] and are going to evolve throughout the design process. Thus, when it is a question of modifying the initial data or state, i.e. functional or structural specifications, it obviously involves considerable disorders.

In CE environment all the phases of specifications can be discussed at any moment. Search for a problem solution can proceed forward from functional specifications to physical specifications or backward from physical specifications to functional specifications. This leads us to propose a schema (Figure 2) with a spiral development to take into account two main factors:
- The permanent contribution of four types of knowledge: objectives, results, solutions and constraints.
- The development of the project around the initial purpose (which is described usually by basic concepts) by applying both the following operators: specification (translation of needs in technology) and evaluation (check of the translation). Specifications becoming globally more and more precise to end as the realisation of prototypes and then the object.

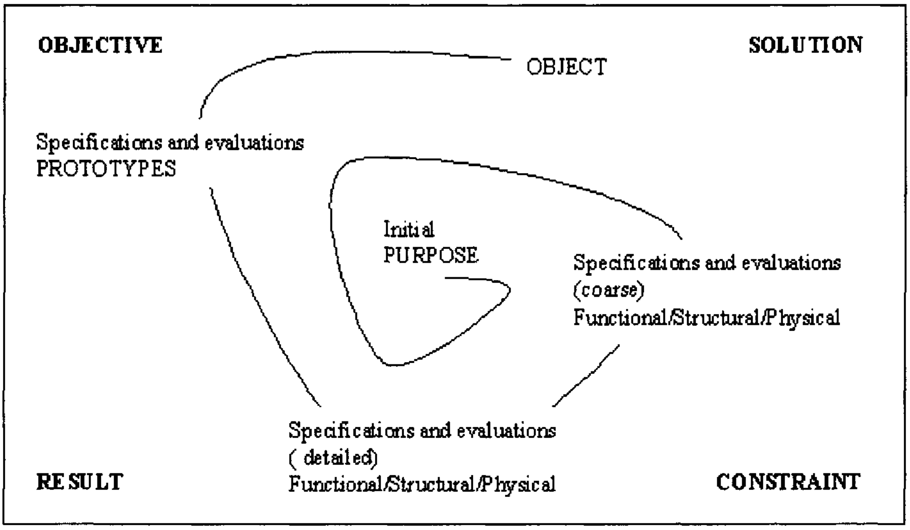

OBJECTIVE
OBJECT
SOLUTION
Specifications and evaluations
PROTOTYPES
Initial
PURPOSE
Specifications and evaluations
(coarse)
Functional/Structural/Physical
Specifications and evaluations
(detailed)
Functional/Structural/Physical
RESULT
CONSTRAINT

Figure 2. *Design problem solving in CE environment*

Let us now examine different procedures used in design problem solving. Although numerous industrial problem solving methods are associated with planned and progressive "paths" towards solutions [SEL 97], design problems cannot be treated like this because objectives are ill-defined and the number of alternative solutions is enormous. Darse [DAR 97] speaks about opportunist solution-making which explains itself by proceeding in two directions: forwards (objectives to solutions) or backwards (modifying objectives according to solutions). Indeed, human beings often perform heuristic search [SMI 95]. Instead of solving problems by exhaustive search of all possible paths, people consider only a small number of alternatives which correspond to known and practicable solutions. Expertise consists in acquiring knowledge which limit in this way the space of search. This approach is largely used and used to understand how human beings usually solve problems. However, these methods do not explain all the procedures for finding a solution and particularly the solutions that may be achieved suddenly [SMI 95, HOL 89, HOLY 96a]. In that case restructuring and parallelism, comparison with sub-structures, are more important than the classical sequential processing or an analytical method. These solutions of emergent type are very important in the current industrial context because they are present in most scientific innovations [HOLY 96a]. Thus, it will be necessary to favour them with procedures or suited methods. For that reason, we are going now to study methods at group level to propose software tools and an organisation of knowledge to help members of the team to find solutions and particularly emerging solutions.

3. The collective aspect and intermediate design objects (IDO)

In a CE environment, solutions can be achieved by creating and manipulating ODI with methods which favour teamwork. But it is very difficult to describe these methods and it is for that reason that we had, in a previous article [SEL 00], identified some domains as the rugby game [JEA 98] and the social insects behaviour builder [THE 97a, THE 97b] to explain some mechanisms used to solve problems collectively. However, there is a fundamental difference between animals and humans. Holyoak and Thagard [HOLY 96b] state that "The ability to form concepts and think about them, which is present in human and to a lesser degree in other mammalian species, marked a fundamental evolutionary advance in intelligence. The advance is closely related to the difference between two types of knowledge, implicit and explicit, that is, between the ability to react to something and the ability to think about it". Thus, knowledge used to solve problems can be implicit and involve a reaction explicit; thinking then provides representations and modelling, and produces a great number of solutions.

At this level and for design activities we might take into account the notion of *coherence* [THA 00]. Actions coming from tacit knowledge or reactions, will lead to a coherent pattern if they are activated by a "stigmergic" logic [SEL 00]. That is, the actors will have common basic principles or long training. Social insects constitute a borderline case because they have all in a given context the same type of reaction. For actions coming from explicit knowledge, a coherent pattern will occur if they are related to object databases [ALK 00] or if constraints (spatial etc) are very severe and thus restrict strongly the search space [WAR 95].This last procedure can be associated with the mechanism called "gabarit" [SEL 00]. Thus a coherent pattern is obtained by reproducing behaviour, actions, objects, but it is necessary to propose new solutions. Social insects find them "at random", that is [SEL 00] they attempt to investigate all possible paths. It is an impossible task for human beings and we have to notice that many advances in scientists' thinking involve mechanisms like analogy, parallel constraint satisfaction, restructuring etc [SMI 95, HOLY 96a]. Solutions are achieved suddenly at an individual level and then might be accepted by the other members of the team. In that case also social insects supply us with an example of the "autocatalytic" process [SEL 00] which can be considered a borderline case.

In the working context of the team project in CE environment, Darse [DAR 97] distinguishes 2 types of design process: co-design and distributed design, and states: "In the situations of distributed design, the actors of the process are simultaneously but not collectively engaged to collaborate; they carry out tasks in advance assigned and try to attempt goals (or at least sub-goals) which they are own. Although, obviously, they had to work for a collective solution of the problem".

In the situations of co-design, "the team members develop solution collectively: they share an identical goal and contribute to the final solution in accord with their specific competence, this with very strong constraints of direct co-operation to guarantee the success of problem solving".

Both types of approach can coexist. In the initial state of the project, it seems better to work with a co-design approach to define the architecture of the system, modules and interfaces. Then these modules and interfaces, so defined, can be studied using distributed conception.

Indeed, taking into account the collective aspect previously developed, both approaches may be used following the nature of the problem and the state of the project:
- Problems mainly related to the specifications (spatial configuration, perceptive aspect etc): it will be necessary to design a structure, an interface, a combination of objects or person, ... and the solution will be a reasonable solution that achieves the goal and not the best solution. Objectives are ill defined and the team members work with "a stigmertgic logic", follow a "gabarit" or use mechanisms like parallelism, restructuring and analogy; it is the co-design approach that seems the most efficient to find solutions and favour emergent solutions.
- Problems mainly related to evaluations (or quantitative characteristics): these are often problems of optimisation usually solved in using deductive reasoning (equations), instruments or simulators. It is rather the distributed design approach which will be used because results will depend on individual competence, software packages or experiments.

Consequently, the IDO used within the framework of design problem-solving should be either distributed (*distributed design*), either visible (or available) by all the members of the team (*co-design*). It is to facilitate these two working approaches which should be available during the whole project for all the team members that we are going now to study a system based on software agents. We shall deal with coherence and associated mechanisms in a future paper.

4. A tool to help in the organisation and the production of knowledge

Thus, in the aim to facilitate the availability of intermediate design objects and to manage the organisation and the production of multilevel knowledge we propose a support suited system which is built around:
- a generic modelling of the IDO or document,
- the sites to store them or resource centres,
- the software entities capable of handling them or mobile software agents [SAN 99, CHE 00, HAR 00, WHI 00].

4.1. *Modelling of an element of knowledge: the document*

We chose to use a uniform model of representation of a knowledge element: the document. This one represents and models in an abstracted way knowledge concerning goals, constraints, solutions and evaluations. An initial document is generated in the first phase with corresponding knowledge. Then from this document a first generation of documents will be made by inheriting knowledge of the first document and so until the generation n, which will correspond to the most specialised knowledge. Every document will keep links with the document *father* and the documents *children* to be able to control the three and regroup documents by function, by component or other characteristics. The knowledge elements so defined will be able to be enriched by the members of the team, the specialists or the subcontractors. For this, in the modelling that we are going to develop, two parts will be considered:
- The element of knowledge such as it appeared during its creation or definition, that is, *the original element of knowledge* and we shall note it IKE (INITIAL KNOWLEDGE ELEMENT),
- The set of the enrichments which were brought to it, the history of the copies of the document, its movements, that is, in a global way, the tracks of the interactions of the document within the project. We shall call *history* this set of information (History).Thus, we shall consider a document as being composed of two elements: the IKE and H (Figure 3).

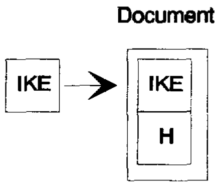


Figure 3. *A document*

4.2. *The resource centres*

Knowledge elements are available in resource centres and the members of the project can use them according to their cognitive gaps or their specific needs to build solutions. Interactions between the first three constituents of the support system are illustrated in the following paragraph and the use of the software agents will be clarified in the following section.

A resource centre RC keeps for the project actors one or several documents. If these documents have been created locally (or in a more precise way the IKE of the document) in a RC, we qualify this RC as *document manager* (DM). In the example chosen, the IKE of the document D was created on RCa and consequently, this last one is DM for D. One supposes that this document was enriched locally and therefore contains a *history* H. A request is engaged in the following way: a user U dependent on another RC (RCb) asks for a copy of the document D (*get request*). Having transferred the wanted document, RCa updates the historic H' of D. According to the policy of a RC for a document, it can send the document in its entirety, that is, the IKE provided with a complete H part, either for reasons of confidentiality, the IKE and part of H. The duplicated document on RCb can also receive requests: the H part of this copy grows then in an independent way of the document D present on RCa. It has the main consequence that a DM cannot know the location of all the copies of the document that it administers. To be able to collect the various H parts of D, scattered within the organisation, the DM which is associated with it (in our example, it is RCa), uses software agents.

4.3. *Mobile software agents*

4.3.1. *Software agents*

Software agents have constituted an active research field since the middle of the 90s. Although one is not capable of defining in a precise way what is an agent, a rather wide consensus exists for the main characteristics that have to possess such software entities. Among aspects usually used to classify the agents, two are used particularly: the *properties* which owns or should possess an agent and the *field of activity*.

Firstly, the main properties usually considered to qualify an agent as a computer entity are: autonomy, mobility, proactivity, adaptivity, cooperation, continuous execution [ETZ 95, JEN 98b]. We can also add: sensing its environment (a tacit property for each agent), *replication* and *learning*. This last property plays an important role in building intelligent software agents [NWA 96]. We consider that these properties are not of the same type: *autonomy*, *proactivity* and *intelligence* are ***qualities*** or aspects of an agent which cannot be described easily, whereas *mobility*, *continuous execution*, *replication* are properties which can be reduced to a mechanism. We call these properties ***attributes***. The distinction between *quality and attribute* is a very important factor in building software agents.

Secondly, an agent is built to perform specialised tasks in several fields [JEN 98a]. and the main themes studied until now in building software agents are: electronic commerce, intelligent user interface, information retrieval, information filtering, entertainment systems, dynamic network routing and concurrent engineering.

The two aspects, properties and fields of activity, were often analysed and combined, leading simultaneously to several definitions of an agent and to several typologies [BRA 97], [NWA 96].

4.3.2. *Mobility*

Mobility is a property which was particularly studied and used since the middle of the 90s [WHI 00, CHE 00, HAR 00]. It was very early associated with software agents to build new applications using internet because models of communication were not appropriate. Indeed, applications were structured on model client/server and generated lots of traffic (Figure 4). In this model, the client part of the application has a dialogue in a synchronous way with the distant service, requiring a permanent connection between both hosts A and B during the whole communication.

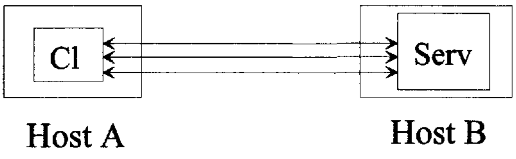


Figure 4. *Client/server model*

To decrease the number of messages and to improve performance and dynamics of applications, White [WHI 00] showed that it was preferable that the software agent moves from the client to the server. The agent or deported client has a local dialogue with the server, and then, when the process is finished, goes back to its site of origin. The traffic now is only due to the transfer of the agent's software, the response delay and the security of the system are seriously improved. Moreover, the user (client) can perform another task during this process (Figure 5).

The working steps of a mobile agent are:

- At time t, the client system based on the Host A create a mobile agent (Ag). After its creation, the agent moves automatically towards the Host B.

- At time t+1, Host B system receives the mobile agent (Ag), restores its state of execution and executes it. The agent opens then a local dialogue with the server. At the end of the dialogue, the agent comes back to the site A or moves towards another host.

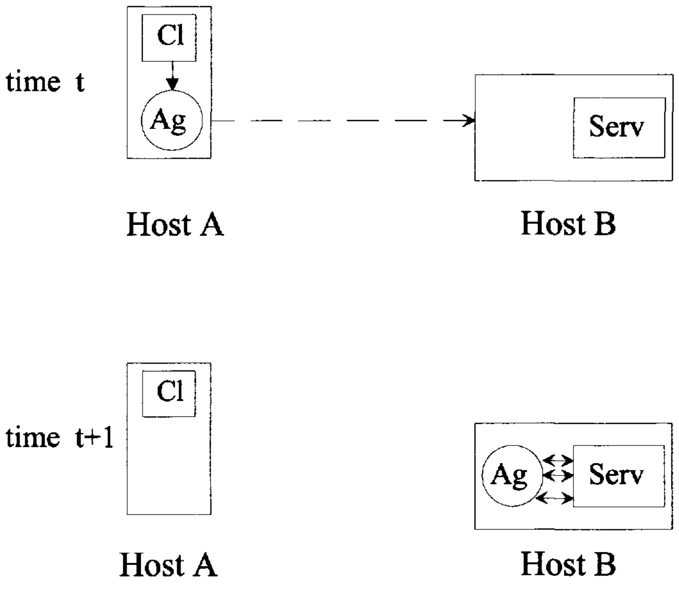


Figure 5. *Modelling mobile agent*

4.3.3. *Modelling systemion software agent*

Our modelling of agent is called *systemion* [SAN 99]. It possesses the two classes of properties which were previously described, that is *qualities and attributes.* Thus, software agents built to collect the history part possess a dual structure (Figure 6):

- A *sub-system* which implants what concerns the fulfilment of the function assigned to the agent. In our application, it is a question of discovering the presence of a document copy, of analysing the historic part and of transporting a copy of this last one towards the DM of the document. The figure below indicates the analysis made by the agent.

- A *sub-system* (or attributes) that implants properties independent of the function allocated to the agent, the mobility and the faculty to communicate.

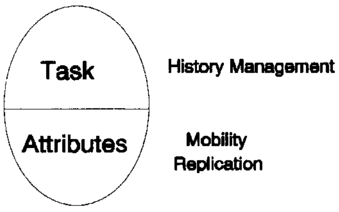


Figure 6. *A software agent*

We can now clear up the work of the software agents (Figure 7). According to a policy previously defined and determined by: events, a regular interval of time or key milestones, the DM of the document D, that is RCa, creates a software agent A1 to analyse the history of the local document D. This contains, in the historic part, the set of the RC having requested a copy of D. In our example, RCa emitted an unique copy of D to RCb. Thus, the software agent duplicates and goes towards the requester RC. When the clone reaches the RC, it analyses the history of the copy (H3) and determines if it has to duplicate itself. Two RC (RCc and RCd) acquired a copy of D, thus two clones A2 and A3 are created and go respectively to RCc and RCd. The agent A1 carries with it a copy of the historic part H3 and goes to RCa, DM of the document D. When the agent A2 reaches the RCc, the document D does not exist any more (for example, the document not having been considered interesting and was erased), it dies, having no historic part to be returned to RCa. The agent A3 discovers on RCd that no new copy was made, it then returns towards RCa with the historic part H4. The processing and the registration of the various histories are made on the site which is DM of the document.

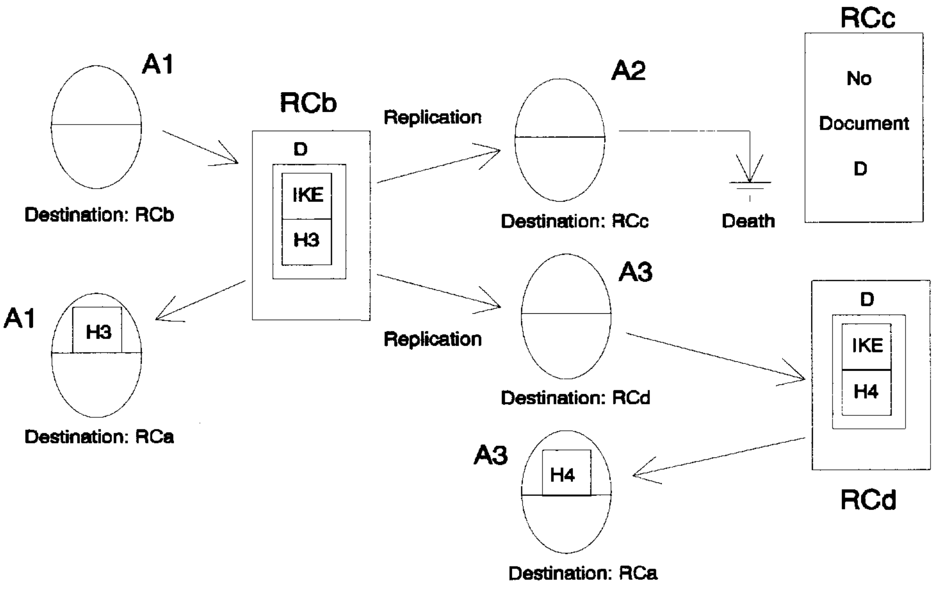


Figure 7. *Support system and software agents*

4.4. *Distributed design process and co-design process*

The registration of the traffic of a document, without or with modifications, inside and (if needed) outside the project team will be associated with:

- *distributed design process* - the traffic of a document (without modification) is registered. The main interest to register the traffic of a document is to understand how it is delivered to the members of the team project. Registration is started by the document manager. This can, after analysis of the historic part, know the location of the document and distribute in a suitable way several copies of the document if needed.
- the backward delivery of the modified document will be associated with a *co-design process*. Now modified copies go back to their owner (backward delivery); that is software agents are going to look for and collect the modified documents. The aim of such an operation is to estimate the state of the project and also, for each member of the team, to favor at the same time its own cognitive improvement and to participate in the *"cognitive synchronisation"* [DAR 97].

4.5. *Control of the design process and concurrent engineering*

According to the structure and the contents of the historic part of a document, numerous applications could be made. Indeed, this support system can be used following two dimensions that allow at any time to establish knowledge mapping and to control the state of the project:
- The vertical dimension from links between the documents, which allows control of the three and to regroup documents by function or by component. The way of handling documents and the links between documents authorises the simultaneous study of several solutions, the overlapping phases and the possibility of studying a function completely or a module or a component independently.
- The horizontal dimension which corresponds to the recording of the interactions between the members of the project, the specialists or the subcontractors in the part "H" of the document. This dimension should allow decrease in the risk of error and to confirm an innovative idea by favouring its emergence. It can lead to a better control of mechanisms ("gabarit", "stigmergy", emergence and "auto-organisation") related to collective working and authorises the creation of project maps composed of chosen elements that have been requested. They will be associated with information concerning the number of requests, the tracks of the interactions, the degree of enrichment and the existence of similar elements.

5. Implementation of software agents based modelling

The first phase of our development was to implement a small prototype to confirm the global architecture of our application. We are going now to describe the main technical characteristics of this prototype by clarifying the various choices which were made.

5.1. *Architecture of the application*

Our support system includes three parts:
- mobile software agents,
- the server of mobile software agents: this receives the agents' new-comers and then starts their execution,
- the user interface which allows the document manager to run a mobile agent.

It is this first software agent that analyses the historic part of the initial document, processes the registered requests GET and begins migration. To do this task, the software agent runs the information which is attached to each request GET, mainly the user's name and the IP address of the machine.

5.2. *Language*

The prototype was written in Perl language. Let us clarify this choice. Two classes of languages are mainly used to implant platforms to use mobile software agents: java and script languages like Unix shell, Perl, Tcl, Python. Perl was chosen because it allows one to write portable and fast programs. It also facilitates the implementation of the actual mobility.

5.3. *Mobility*

Two paradigms structure the architecture of an agent application: *actual mobility* and *virtual mobility*. In actual mobility, all parts of the mobile agent (code, data and perhaps the context of execution) are sent to the target host and then destroyed in the broadcasting host. In virtual mobility, all parts of the agent are also sent but not completely destroyed in the broadcasting host; applets implement the paradigm of the virtual mobility. The advantage of actual mobility, our choice, is that no specific agent components stay on a visited host. But what are the parts of the agent which are included in the migration package?

Usually two possibilities are examined, they are called *weak mobility* and *strong mobility*. Our mobile agents are structured according to the model of the weak mobility. It means that only the code and data of the agent are transferred on the target host. A part of these data is used to restart the execution of the agent at a particular point of its code.

In strong mobility, the context to run an agent, that is mainly the stack and the program counter, must be captured and then sent to the target host. The latter then uses this context of execution to restart the execution of the agent at the point where it had stopped on the broadcasting host. The implementation of strong mobility requires the use of non standard computers (i.e. equipped with modified JVM, modified Tcl interpreter etc). Contrary to strong mobility, weak mobility does not require any particular modification of the target host. Thus, in our application, we use a standard Perl interpreter.

5.4. *Evaluation and perspectives*

The proposed methodology must be applied to real cases concerning design and the making of scientific instruments [ROM 95, ROM 98]. It should allow a CE development type much needed by this highly innovative industrial domain. Indeed, the development of instruments is usually carried out by teams composed of members with various competencies, scattered geographically and coming from various entities (enterprise, research centre, universities).

As a first step we were able to verify by means of this prototype the feasibility of such a system and particularly the migration of the agents by means of the historic part of documents. A second step will consist of implanting it on a widely used internet platform accessible with classic browsers. Thus, any user (client) could reach the resource centres if it is authorised.

6. Conclusion

Knowledge-making for design problem solving in CE is not carried out in a linear way but rather under a spiral development which takes place around the initial purpose by using objects which one calls IDO (intermediate design object) [JEA 98]. Solutions are obtained by various types of individual and collective mechanisms according to the nature of the problem to be solved and/or the capacity of the human beings. Among these mechanisms we will mention the behaviour of the social insects which allows us to propose three mechanisms to explain the coherence in builder activity: “Gabarit”, “stigmergic ” logic and actions or solutions of emergent and “auto-organised” type.

Thus, in the framework of design problem-solving a process to organise use and produce multilevel knowledge is defined and revised using software tools and a suitable support system. Knowledge is partitioned into modules or elements. Each of them is composed of a document associated with multilevel knowledge and a software agent that accomplishes tasks like to go from machine to machine, sense the environment, control birth with duplication and death. These elements are available in resource centres or web sites and project members could ask them to build solutions for problem solving. Each request produces a duplicate element. The software agents attached to this element carries the document, gives resource centre information about the requester environment and take into account a supplement of information from the requester if needed.

All these operations and resources constitute a knowledge management system. It can provide project maps composed of the requested elements associated with parameters related to the number of request, the identity of the requesters, the quantity of information added etc. The purpose is not only to estimate the state of the project but also, for every member, to participate in the construction of a common working repository. Joint presence in the same local space of several concurrent or complementary IDO favours at the same moment the cognitive enrichment of the actor [DAR 97] and the use of collective mechanisms, facilitating the emergence of new solutions and the development of new products. Although this system seems well suited for CE design projects, it needs to be integrated in the "virtual enterprise". This includes organisational problems that will be discussed in a future paper.

7. References

[ALK 00] A. Al-Khuder, W.A. Gray, J.C. Miles, *Issues in Management of Distributed Concurrent Engineering Design in Object-Oriented Databases,* Proceedings of CE2000, Advances In Concurrent Engineering, 2000, p. 586-595.

[BRA 97] Bradshaw J. M., *An Introduction to Software Agents,* in Software Agents, Bradshaw, J.M. (ed.), Cambridge, MA: MIT Press, 1997.

[BUR 94] Ph. Burg, J.L.Selves, J.P. Colin, *Procédés de détermination des caractéristiques d'un pétrole brut,* Brevet N° 9403188 (1994).

[BUR 95] Ph. Burg, J.L.Selves, J.P. Colin, *Numerical simulation of crude oil behaviour from chromatographic data,* Analytica Chemica Acta, 317(1995) 107-125.

[CAR 91] Donad E. Carter, Barbara Stilwell Baker, *Concurrent Engineering, The Product Develpment Environment for the 1990s,* Addison-Wesley Publishing Company, 1991.

[CHA 97] Florence Charue-Duboc, *Maitrise d'œuvre, maitrise d'ouvrage et direction de projet,* Annales des Mines, September 1997, p 41-48.

[CHE 00] Chess D., Grosof B., Harrison C., Levine D., Parris C., Tsudik G., *Itinerant Agents for Mobile Computing,* IBM Research Report, http://www.research.ibm.com/massive/rc20010.ps

[DAR 97] F. Darses, 'Ingénierie concourante, de la technique au social', P. Bossard et al., ECONOMICA, 1997, Chapter III.

[ETZ 95] Etzioni O., Weld D. S., *Intelligent agents on the Internet: Fact, Fiction, and Forecast,* IEEE Expert 10(4): pp 44-49, 1995, http://ftp.cs.washington.edu/pub/ai/ieee-expert.ps.Z

[FER 95] J. Ferber, Les Systèmes multi-agents. Vers une intelligence collective, 1995, InterEditions, Paris, p. 361-419.

[GAU 97] Frédéric Gautier, *Evaluation économique des activités de conception et de développement des produits nouveaux,* Cahier de Recherche du Gregor, 1997.12 (http://www.univ-paris1.fr/GREGOR/).

[HAR 00] Harrison C., Chess D., Kershenbaum A., '*Mobile Agents: Are they a good idea?*', IBM Research Report, http://www.research.ibm.com/xw-d953-mobag-ps

[HOL 89] JH Holland, KH Holyoak, RE Nisbett, P.R. Thagard, *Induction - Processes of inference, learning and discovery,* MIT Press, 1989, p.336-342.

[HOLY 96a] KH Holyoak, P.R. Thagard., *Mental leaps. Analogy in creative thought,* MIT Press. 1996, p. 101-137.

[HOLY 96b] KH Holyoak, P.R. Thagard., *Mental leaps. Analogy in creative thought,* MIT Press. 1996, p. 21.

[JEA 98] A. Jeantet et J.F. Boujut, 'Conception de produits mécaniques', M. Tollenaeree, HERMES, 1998, Chapter 5.

[JEN 98a] Jennings N., Wooldridge M., *Applications of Intelligent Agents,* in '*Agent Technology: Foundations, Applications, and Markets*' (Edited by N. R. Jennings and M. Wooldridge) Springer Computer Science, 1998, http://www.springer.de/comp/special/ jennings.pdf

[JEN 98b] Jennings N., Sycara K., Wooldridge M., *A Roadmap of Agent Research and Development,* Autonomous Agents and Multi-Agent Systems, 1, pp 275-306 (1998), Kluwer Academic Publishers, Boston, ftp://ftp.elec.qmw.ac.uk/pub/isag/distributed-ai/publications/ aa-mas.ps.gz

[MID 97] Christophe Midler, *Evolution des modèles d'organisation et régulations économiques de la conception,* Annales des Mines, February 1997, p 35-40.

[NEW 72] A. Newel and H.A. Simon, *Human Problem Solving,* Englewood Cliffs, NJ, Prentice Hall, 1972, p.787-868.

[NON 97] I. Nonaka et H. Takeuchi, *La connaissance créatrice, la dynamique de l'entreprise apprenante,* DeBoeck University, 1997, p. 108-114.

[NWA 96] Nwana H. S., *Software Agents: An Overview,* Knowledge Engineering Review, Vol 11, No 3, p. 1-40, September 96, Cambridge University Press, http://www.cs.umbc.edu/ agents/introduction/ao.ps

[ROM 95] Romier J., Selves J.-L., Béteille J.-P., Gastellu-Etchegory J.P., Marty G., *Réalisation et etalonnage d'un spectromètre de terrain visible et Infrarouge pour l'étude de la réflectance de la végétation,* Analusis, 23(1995) p.403-411.

[ROM 98] Romier J., Selves J.-L., Gastellu-Etchegory J.P., *Imaging spectrometer based on an opto acoustic tunable filter,* Review of Scientific Instruments, Vol 6, N°8 (1998) p. 2859-2867.

[SAN 99] E. Sanchis, *Modular Autonomy for Simple Agents,* Third International Conference on Autonomous Agents - Workshop on Autonomy Control Software - May 1-5, 1999, Seattle (WA).

[SAR 97] Jean-Claude Sardas, *Ingénierie Intégrée et mutation des métiers de la conception,* Annales des Mines, February 1997, p 41-48.

[SEL 00] JL Selves, Z.Y. Pan, E. Sanchis, Knowledge needs and new products development wthin a Concurrent Engineering Environment, Proceedings of CE2000, Advances In Concurrent Engineering, 2000, p 635-643.

[SEL 97] J.-L. Selves, Ph. Burg, *Décisions techniques et méthodes numériques dans un processus industriel appartenant à l'industrie pétrolière,* Proceedings of CNRIUT'97, Blagnac, 14-16 May 1997.

[SIM 91] H.A.Simon, 'Sciences des systèmes, sciences de l'artificiel', Dunod, Paris, 1991.

[SMI 95] EE Smith & D.N. Osherson, *Thinking - An Invitation to Cognitive Science,* Second Edition, Vol 3, MIT Press, 1995, p. 267-296.

[TAK 86] H. Takeuchi and I. Nonaka, The new new product development game, Harvard Business Review, January-February 1986, p. 137-146.

[THA 00] P. Thagard, *Coherence in Thought and action,* MIT Press, 2000, p. 223-245.

[THE 97a] G. Theraulaz, F. Spitz, *Auto-organisation et comportement*, Hermés, Paris, 1997, p 80-83.

[THE 97b] G. Theraulaz, E. Bonabeau, *La modelisation du comportement bâtisseur des insectes sociaux*, paper in book reference [21], p 210-234.

[WAR 95] Allen Ward, Jeffrey K. Liker, John J. Cristiano, Duward K. Sobek, II, The Second Toyota Paradox: How Delaying Decision Can Make Better Car Faster, Sloan Management Review, Spring 1995, p 43-41.

[WHI 00] White J. E., *Telescript Technology: The Foundation for the Electronic Marketplace,* General Magic White Paper, General Magic, Inc., Sunnyvale, CA 94088.

Chapter 7

An IEC 61499-based Model for Reconfiguration of Real-time Distributed Control Systems

R.W. Brennan and D.H. Norrie
Dept of Mechanical and Manufacturing Engineering, University of Calgary, Canada

M. Fletcher
Agent Oriented Software, Cambridge, UK

1. Introduction

Today's manufacturing systems must be capable of quickly responding to change while maintaining stable and efficient operation. Although manufacturing technology has become increasingly sophisticated to deal with this (e.g., through advanced robotics and computer numerical control), without adequate control the result is often a collection of "islands of automation" that lack the necessary integration for truly responsive behaviour [UPT 95]. As a result, new control software and hardware approaches are required to realise a system that is flexible (i.e., capable of reconfiguration) and responsive (i.e., capable of recovering from disturbances).

In this contribution, we report on the development of a distributed intelligent control solution that is inherently adaptable and dynamically re-configurable, that takes advantage of distributed artificial intelligence at the planning and control levels to achieve significantly shorter up-front commissioning times as well as significantly more responsiveness to change. These potential benefits, in combination with the current trend towards low-cost, distributed computing platforms will result in a much more attractive, low-cost automation solution for future manufacturers than current centralised solutions.

We begin with some background on holonic and agent-based approaches in manufacturing as well as recent work on real-time distributed control. Then, in Section 4, we describe two approaches that can be used to achieve dynamic and intelligent reconfiguration in this environment. In Section 4 we describe a simple prototype system that is used to support function block configuration and reconfiguration. Finally, we provide a brief summary and discussion of our current work in Section 5.

2. Background

Distributed intelligent control involves matching the control model more closely with the physical system. This is particularly relevant to manufacturing control systems that are required to control widely distributed devices in an environment that is prone to disruption. With this model, control is achieved by the emergent behaviour of many simple, autonomous and co-operative entities (i.e., agents) that "decide locally not only how to act (as subroutines do), and what actions to take (as objects do), but also when to initiate their own activity" [PAR 93]. In the following sub-sections we look at how this distributed approach has been applied in the manufacturing domain and in particular, to the problem of manufacturing systems control.

2.1. *Agent-based manufacturing*

The natural fit of multi-agent systems technology to manufacturing problems has resulted in many applications in this domain. In particular, autonomous agents or multi-agent systems (MAS) are an attractive software engineering tool for the development of systems in which "data, control, expertise, or resources are distributed; agents provide a natural metaphor for delivering system functionality; or a number of legacy systems must be made to interwork," [WOO 99]. Manufacturing applications are characteristically "modular, decentralised, changeable, ill-structured, and complex" [PAR 99], and as a result supply a host of problems that are well-suited to agent technology.

Research into the application of multi-agent systems and distributed artificial intelligence in the manufacturing domain has been steadily growing over the last ten years and has focused on all areas of the manufacturing enterprise ranging from product design to real time control. One of the first applications of agent-technology in the manufacturing domain was Parunak's YAMS ("Yet Another Manufacturing System") factory control system [PAR 87]. This system took advantage of the contract net protocol [SMI 82] for inter-agent negotiation to flexibly assign resources to production tasks in a changing environment (i.e., machine failures). Various other applications of agent technology followed in this domain that include product design, enterprise integration and supply chain management, planning and scheduling, and real-time control. For a comprehensive overview of agent-based systems in manufacturing, see Shen and Norrie [SHE 99] and for multi-agent systems in concurrent design and manufacturing Shen, Norrie and Barthes [SHE 00] can be consulted.

2.2. *Holonic manufacturing systems*

Holonic manufacturing systems are manufacturing-specific applications of the broader multi-agent systems approach and are one of the *intelligent manufacturing systems* (IMS) program's six major projects resulting from a feasibility study conducted in the beginning of the 1990s [HMS 01]. The objective of the work of the HMS consortium is to "attain in manufacturing the benefits that holonic organisation provides to living organisms and societies, e.g., stability in the face of disturbances, adaptability and flexibility in the face of change, and efficient use of available resources" [HMS 01]. The term "holon" was coined by Arthur Koestler [KOE 67] who observed a dichotomy of wholeness and partness in living organisms and social organisations and stated "wholes and parts in the absolute sense do not exist anywhere."

Like an HMS, a multi-agent manufacturing system also consists of co-operative and autonomous manufacturing units, but unlike an HMS, a MAS can be considered as embedding a "general software technology that was motivated by fundamental research questions" [BUS 98]. Research in MAS, however, has played a key role in the development of holonic manufacturing systems. For example, Figure 1 illustrates how holons can be thought of as physical agents that consist of a software component and a hardware component. Because of the close relationship between holonic systems and MAS concepts, object-oriented and agent-based techniques have played an important role in holonic systems research in areas such as material handling, production planning and scheduling, real-time control and holonic systems architectures. An extensive overview of the holonic systems approach to production planning and control is provided by McFarlane and Bussmann [MCF 00]. As well, the proceedings of IMS'99 provide numerous papers on various aspects of holonic manufacturing systems research [VAN 99].

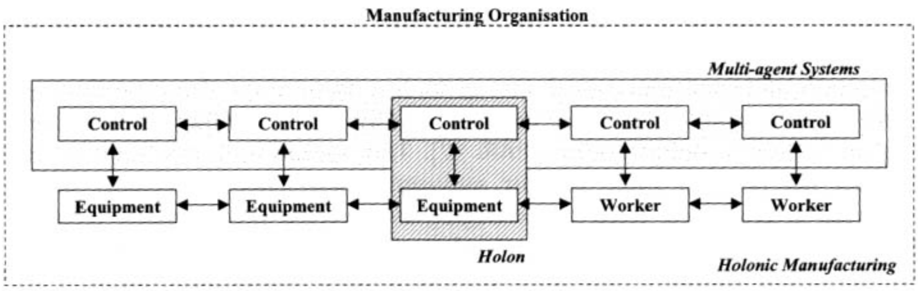


Figure 1. *Agents and holons [BUS 98]*

2.3. *Real-time distributed control*

Although there has been a considerable amount of work on agent-based approaches to the upper, planning and scheduling level of control very little work has been done on applying these techniques to the lower, real-time control level. The main barriers at the real-time control level result from the difficulty of implementing MAS concepts in a stochastic environment where hard real-time constraints must be met to achieve safe system operation.

The primary distinction between non-real time and real time systems is that real time systems tightly link correctness with timeliness. In other words, deadlines must be met under hard real time (i.e., tasks must finish by a specified time) and soft real time (i.e., tasks must meet deadlines on average) constraints [DOU 99]. As well, real time systems are typically safety-critical systems (i.e., the system should not incur too much risk to persons or equipment), and as a result, characteristics such as timeliness, responsiveness, predictability, correctness and robustness are of fundamental importance. In summary, the step from the non-real time or soft real time domain is a large one that requires new models and methodologies for distributed control.

2.4. *The function block architecture*

Recently, there have been a number of advances in real-time distributed control that provide the tools to move away from the traditional centralised, scan-based *programmable logic controller* (PLC) architecture towards a new architecture for real time distributed intelligent control. In particular, there have been a number of advances recently in programming languages [LEW 96], models for distributed control [IEC 00] and software methodologies [LYO 98].

The International Electro-technical Commission (IEC) 61499 standard is one example of this new trend [IEC 00]. This standard addresses the need for modular software that can be used for distributed industrial process control. In particular, this standard builds on the function block portion of the IEC 61131-3 standard for PLC languages [LEW 96] and extends the function block (FB) language to more adequately meet the requirements of distributed control in a format that is independent of implementation.

In developing control applications with this model, the IEC 61499 function block can be thought of in terms of an "enhanced" object. Like recent object-oriented and agent-based models for manufacturing system control, the IEC 61499 function block shares many of the characteristics of the traditional objects and agents used to develop these applications (e.g., a traditional object focuses on data abstraction, encapsulation, modularity, and inheritance) [BRE 01].

An example of an IEC 61499 basic function block is shown on the left side of Figure 2(a). The function block is enhanced through its recognition of two very specific kinds of messages: data messages (which one would expect of a traditional object) and event messages (which are used to schedule the execution of an algorithm). The resulting focus on process abstraction and synchronisation makes this approach particularly suitable for control of an environment that is concurrent, asynchronous and distributed. Figure 2(b) shows how function blocks can be represented recursively; the composite function block shown here consists of multiple basic function blocks and/or composite function blocks.

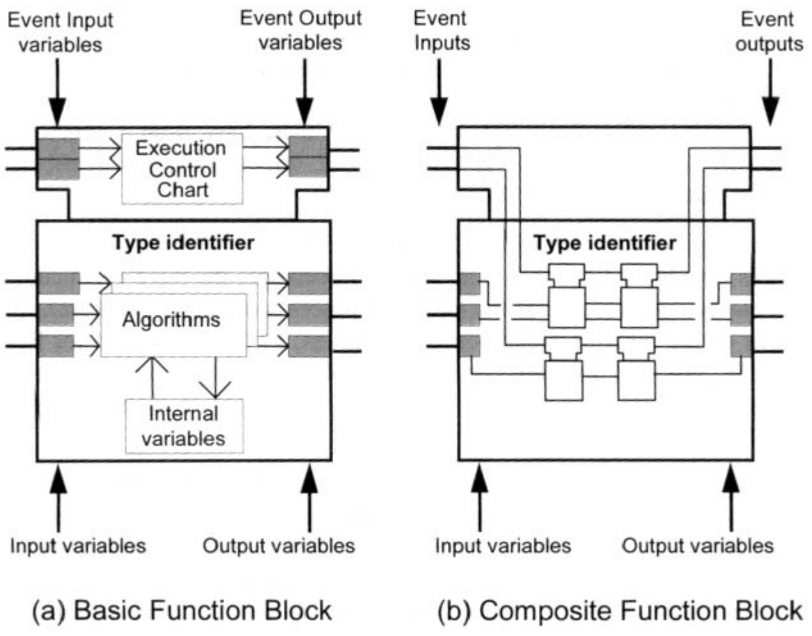


Figure 2. *The IEC 61499 function block model [IEC 00]*

Figure 3 illustrates the IEC 61499 system model. The "devices" shown in this figure are, for example, machine controllers with I/O interfaces to the physical environment. As can be seen in this figure, applications (modelled by IEC 61499 function blocks) can reside on a single device, or be distributed across multiple devices.

The device is shown in more detail in Figure 4. As can be seen in this figure, the device is a container for resources, where the resource provides the execution environment for the application's function blocks [IEC 00]. As well, both the communications and processing interfaces are shown explicitly in this model, allowing physical devices to be mapped to the resources and the various resources to communicate with each other respectively.

The resource is responsible for scheduling and executing function block algorithms and can be thought of as a logical subdivision within the software (or hardware) of a device, which has independent control of its execution [IEC 00].

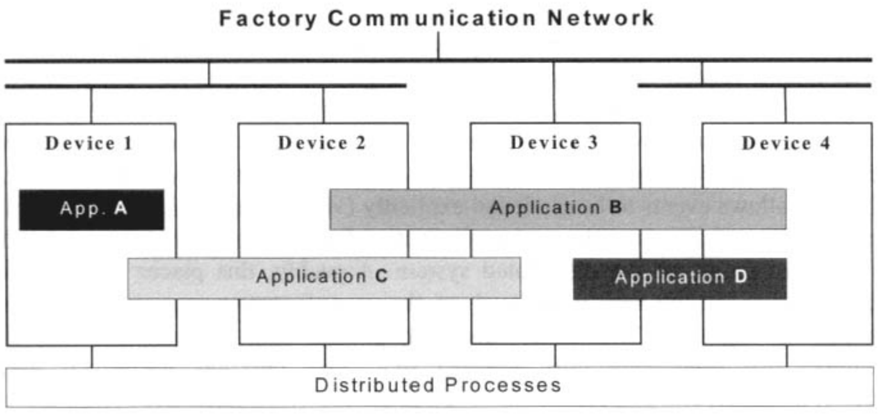


Figure 3. *The IEC 61499 system model [IEC 00]*

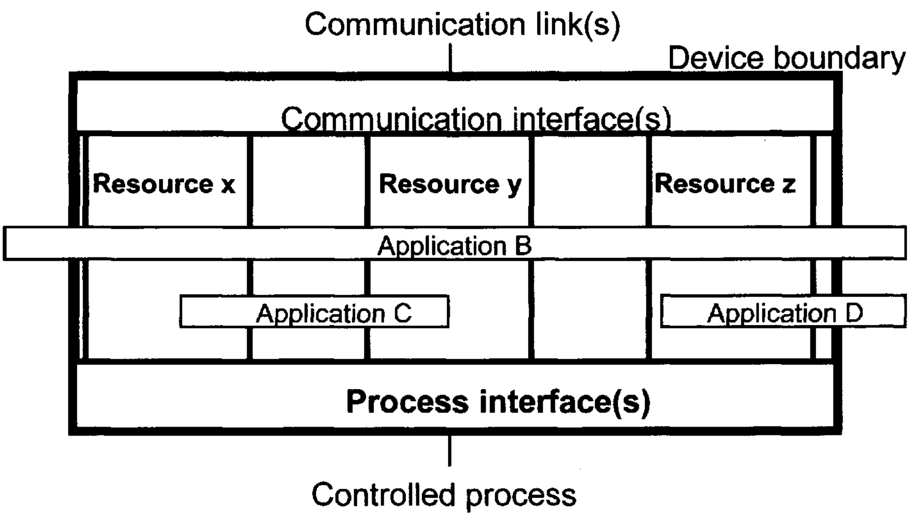


Figure 4. *The IEC 61499 device model [IEC 00]*

The applications illustrated in Figures 3 and 4 consist of a network where nodes are function blocks and branches are data and event connections [IEC 00]. It should be noted that an important characteristic of this model is that the application can be distributed among several resources in the same or different devices.

Figure 5 extends Figure 1 to show how each of the models discussed in this section fit into a holonic manufacturing system. Fundamentally, all are the same kind of model, though each is specialised for a given need: it is the stance that is taken toward the problem that will influence the model that is used. For example, general agent-based approaches are suited for non-real time environments whereas the IEC 61499 model is suited for real time, event-based environments. In particular, IEC 61499 allows events to be modelled explicitly (via the application and function block models) and also provides a clear distinction between the software model and the hardware model of the distributed system. Arguably, this places IEC 61499 function blocks at the real time level of the manufacturing organisation as is illustrated in Figure 5. IEC 61499 does not provide any guidance on how to implement real time constraints however (e.g., task duration, allowance, and deadline) however, and as a result, this is left as an implementation issue (e.g., using a real time operating system as with our prototype system described in Section 4).

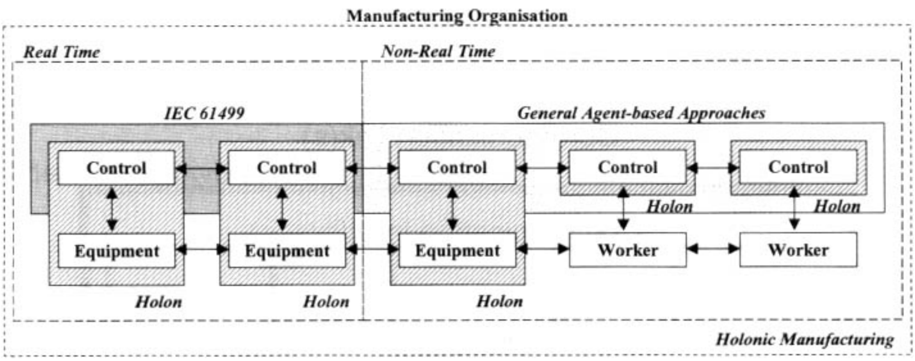


Figure 5. *The relationship between the models*

3. Reconfiguration of real-time distributed control systems

The primary objective of the research reported in this paper is to develop techniques to achieve automatic reconfiguration that results in predictable and stable system behaviour in a real-time environment. In conventional PLC systems, reconfiguration involves a process of first editing the control software offline while the system is running, then committing the change to the running control program. When the change is committed, severe disruptions and instability can occur as a result of high coupling between elements of the control software and inconsistent real time synchronisation. For example, a change to an output statement can cause a chain of unanticipated events to occur throughout a ladder logic program as a result of high coupling between various rungs in the program; a change to a PID function block can result in instability when process or control values are not properly synchronised.

In order to develop appropriate methodologies, reconfiguration can be viewed in three levels of sophistication: simple, dynamic and intelligent reconfiguration. These three types of reconfiguration can be summarised as follows: (i) simple configuration utilises the IEC 61499 model to avoid software coupling issues during reconfiguration, (ii) dynamic reconfiguration uses techniques to properly synchronise software during reconfiguration, and (iii) intelligent reconfiguration exploits multi-agent techniques to allow the system to reconfigure automatically in response to change.

Figure 6 provides a high-level overview of the general model for reconfiguration that we are currently developing. With this model, function block ports (i.e., event and data connections) are objects that register with the *resource manager* (RM) associated with the function block. The resource manager looks after the interconnection of function block ports (i.e., as is specified by the application) and maintains a record of all function block ports in a FB Port table.

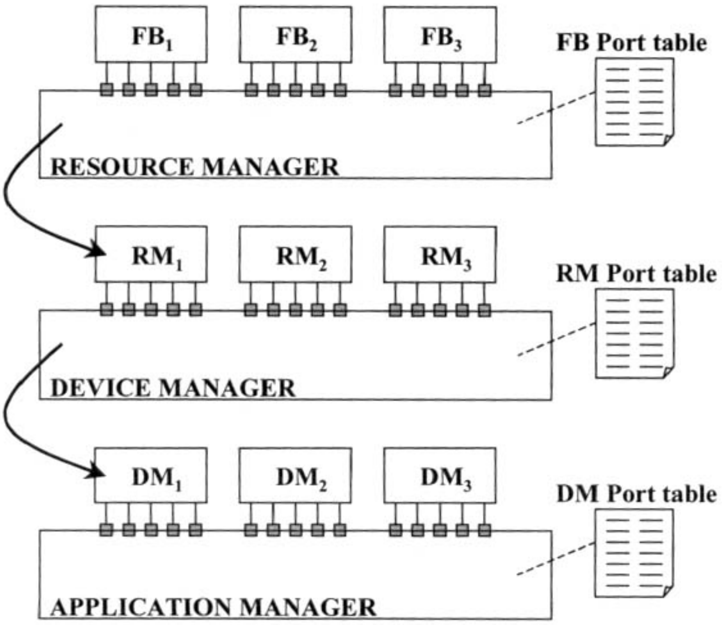


Figure 6. *The reconfiguration model*

As is illustrated in Figure 4, a device consists of one or more resources. Consequently, the relationship between resources and devices can be thought of in similar terms to the relationship between function blocks and resources: i.e., the *device manager* (DM) looks after the interconnection of the RMs function block ports and stores this information in an RM Port table. Similarly, the *application manager* (AM) looks after the interconnection of the DM's function block ports and stores this information in a DM Port table.

Using this model, a device manager, for example, may reconfigure the ports of its RMs to whatever new configuration is desired. The advantage of this approach is that reconfiguration can be managed at various levels (i.e., function block, resource, device, application). For example, at the most basic level all that is required is a "map" of the new configuration (i.e., based on the FB, RM, and DM Port tables). This approach will be discussed further in Section 3.1.

The approach described thus far allows for the "simple reconfiguration" discussed previously, but does not yet address how dynamic and intelligent reconfiguration is performed. The fundamental difference between basic and dynamic reconfiguration is the latter's recognition of timeliness as a critical aspect of correctness. The goal is to develop techniques to allow the user to change a portion of the program (e.g., basic FB, composite FB, or sub-application), while maintaining the timeliness requirements of the application. This will require freezing the state of the current FB when the change is committed, saving this state information, automatically initialising the new FB to the correct state, replacing the current FB with the new FB, and then starting the new FB. The main difference here is that matching of state information is performed automatically (rather than by the user). This is intended to result in a transparent change to the application (i.e., other FB's should see no change at the FB interface) which is intended to result in much more stable system operation than can be realised by current systems.

Intelligent reconfiguration builds on dynamic reconfiguration (i.e., timeliness constraints) by focusing on multi-agent techniques to allow the system to reconfigure automatically in response to change. For example, as part of a fault recovery strategy, higher-level agents will manage the reconfiguration process using diverse or homogeneous redundancy. In the following sub-sections, we describe two approaches to achieve these more advanced forms of reconfiguration: (1) a pre-programmed or "contingencies" approach, and (2) a soft-wiring approach.

3.1. *Contingencies approach to reconfiguration*

With this form of reconfiguration control, contingencies are made for all possible changes that may occur. In other words, alternate configurations are pre-programmed based on the system designer's understanding of the current configuration, possible faults that may occur, and possible means of recovery.

This approach uses pre-defined reconfiguration tables that make use of the FB, RM and DM Port tables described above. For example, in the event of a device failure, the affected portions of an application could be moved to different devices by selecting an appropriate reconfiguration table. As well, this detailed representation of the function block interconnections would allow higher-level agents to access the information required to make a smooth transition from one configuration to another, thus enabling dynamic reconfiguration.

The main disadvantage of this approach is that it is inflexible, particularly with respect to the handling of unanticipated changes. As well, this approach would require constant maintenance in order to keep the reconfiguration tables current: i.e., each change would require a change to the reconfiguration tables.

3.2. *Soft-wiring approach to reconfiguration*

The basic idea behind this approach to reconfiguration is to enable higher layers (e.g., RM, DM, AM) to use higher-level reasoning to analyse the current configuration and plan for reconfiguration when required. Ideally, we are striving for an "integrated circuit" approach where fine grain components can be "plugged-in". Similar to Sun's Jini approach, this approach uses the directory services of the FB, RM, and DM Port tables as well as underlying *configuration agents* (CA) that handle the "wiring" between components. For example, function blocks will have information on how they can be connected (i.e., their interfaces) that is stored by CAs. The CAs will use this information, for example, to connect a new function block with an existing function block or to replace an existing function block with a new one.

The primary advantage of this approach is its potential to overcome the inflexibility of the contingencies approach as well as its potential to realise intelligent reconfiguration. For example, configuration agents will be primarily reactive in nature, responding quickly to changes in the physical environment. These agents however, through emergent behaviour and interaction with higher-level cognitive agents, will allow sophisticated decision-making to occur in the real-time system. An example of this of this type of multi-level behaviour is safety management. The agents involved in achieving this functionality will vary from very simple physical agents that are limited to changing to a fail-safe state and reporting a fault, to higher-reasoning functional agents capable of collaborating to achieve a sophisticated recovery plan.

4. Prototype implementation

In order to develop real-time distributed control systems that are capable of dynamic, and eventually, intelligent reconfiguration, an appropriate software infrastructure is required that supports approaches such as the contingencies approach or the soft-wiring approach. In this section we describe the implementation of a simple prototype system that is implemented in real-time Java and is based on

IEC 61499. For this implementation we focus on function block reconfiguration (i.e., the top part of Figure 6), which relies on a FB Port Table.

We start with a brief description of the general architecture that has been developed to support control application configuration and reconfiguration then describe our recent work on an approach to achieve dynamic "plug-and-play" (or, in the manufacturing context, "plug-and-produce") capabilities. In the sub-sections that follow, we use the term "configuration" to imply both initial control application development (i.e., configuration) as well as control application modification (i.e., reconfiguration).

4.1. *Configuration control services and support architecture*

The architecture for configuration control, shown in Figure 7, consists of three basic modules to enable control application configuration and reconfiguration: (i) a *configuration management application*, (ii) a *configurations services* module, and (iii) a *configuration control application*. As well, Figure 7 shows the *local control application* and a *remote device*, where parts of the control application may be distributed or other control applications may be running (i.e., as illustrated in Figures 3 and 4). In the remainder of this sub-section, we will look at each of these aspects of the configuration control architecture.

The *local control application*, shown in the lower right of Figure 7 is intended to represent an IEC 61499 function block application. IEC 61499 function blocks (e.g., Figure 2) are modelled by configuration agents (CA) and execution agents (EA). First, the function block's control functionality (i.e., how it is expected to behave in the control application) is encapsulated in execution agents (*EA*), which have a direct correspondence with the basic function block model shown in Figure 2(a). For example, the IEC 61499 ECC (execution control chart), algorithms and internal data are encapsulated in EA task agents. One form of representation for the execution agent that has been suggested by the authors [FLE-01] is to use the real-time unified modelling language (RT-UML) "capsule" stereotype [SEL 98]. This model allows state machines to be modelled explicitly and also supports a recursive structure.

As noted previously, configuration agents manage function block configuration. Figure 7 illustrates this graphically by the function block configuration agents (*CA*). In order to manage which agent (EA or CA) is executing, or in other words, whether the function block is in the execution or configuration flow path, an additional state machine is used in our function block model: the basic function block configuration control chart (not shown in Figure 7). This state machine can be in one of three states: configuration ("C"), execution ("E"), or stopped ("S"). In the "configuration" state, the function block CA is running and performing configuration commands that are based on messages received from the CCEE (described below). In the "execution" state, the function block EAs run, allowing the function block to perform its control application functionality. Finally, the function block can be placed in a stopped state if required (e.g., prior to a configuration change).

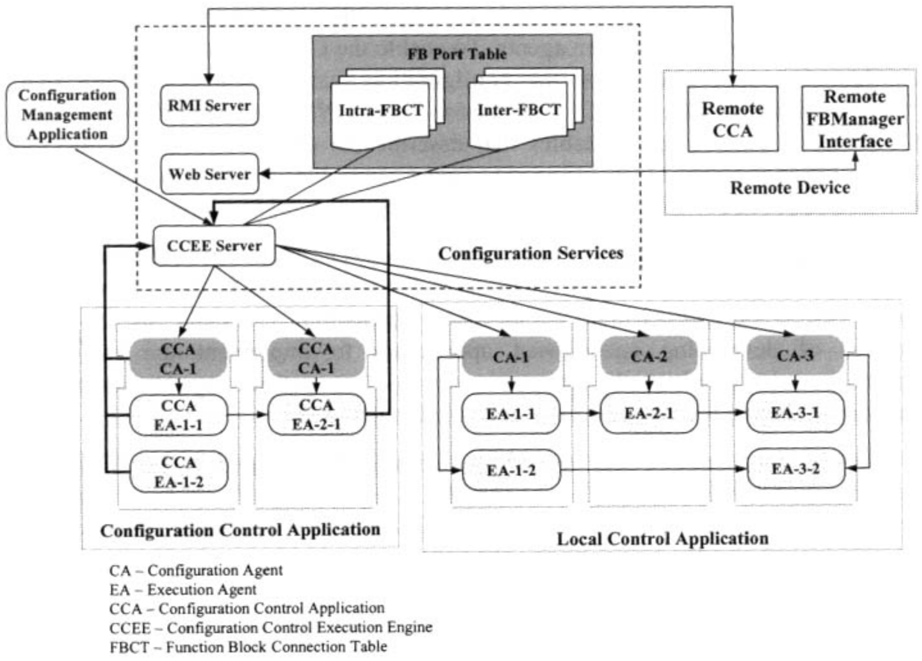

FB Port Table
Configuration Management Application
RMI Server
Intra-FBCT
Inter-FBCT
Remote CCA
Remote FBManager Interface
Web Server
Remote Device
CCEE Server
Configuration Services
CCA CA-1
CCA CA-1
CA-1
CA-2
CA-3
CCA EA-1-1
CCA EA-2-1
EA-1-1
EA-2-1
EA-3-1
CCA EA-1-2
EA-1-2
EA-3-2
Configuration Control Application
Local Control Application
CA – Configuration Agent
EA – Execution Agent
CCA – Configuration Control Application
CCEE – Configuration Control Execution Engine
FBCT – Function Block Connection Table

Figure 7. *Configuration control services and support architecture*

Similar to local control applications, the *configuration control application* (CCA) is also modelled by function blocks in our model. The main difference here is that the CCA is a special type of control application that executes a pre-determined configuration that is provided by the configuration execution engine (described next). Just as a control application controls the behaviour of physical devices (e.g., robots, CNC machines), the CCA control the behaviour of the control application. In other words, the CCA can be thought of as a meta-control application that is responsible, for example, for how function blocks in the local control application are interconnected.

The *configuration services module* shown in Figure 7 acts as an interface between this meta-control application and the higher-level configuration management application, where configuration and reconfiguration plans are developed. As well, since control applications exist in a distributed environment, it also serves as an interface to other devices.

The key elements of the configuration services module are the *configuration control execution engine* (CCEE) and the *FB Port Table*. The CCEE basically completes the configuration control loop: i.e., it monitors the status of execution agents (and reports this information to the configuration management application) and relays the configuration commands (from the configuration management application) to the configuration agents. To enable the CCEE to convert higher-level configuration commands from the configuration management application (e.g., "connect function block *ADC1* to function block *PID3* on *RESOURCE2*"), the CCEE relies on two sets of tables that describe the connections within function blocks (*intra-function block connection tables* (intra-FBCT)) and the connections between function blocks (*inter-function block connection tables* (inter-FBCT)).

The key to achieving intelligent reconfiguration (i.e., multi-agent techniques that enable the system to reconfigure automatically in response to change) with this model lies in the *configuration management application*. Currently, we are in the process of developing agent-based approaches to implement the two basic approaches described in the previous section. Our preliminary architecture to support intelligent reconfiguration is described in [ZHA 00].

Finally, configuration and reconfiguration can be managed across distributed devices either automatically (i.e., invoked by a remote device's CCA) or manually through a *remote function block (FB) manager interface*. The configuration services provided to support these two modes of configuration are remote method invocation (RMI) and web services respectively.

4.2. *Implementing "plug-and-produce" capabilities*

A key requirement of this research is the development of systems that can dynamically form coordination domains, or teams of holons in real-time. This requirement is tightly linked with the central HMS ideal of developing manufacturing systems that can automatically and dynamically adapt to change. In this sub-section, we describe one approach to achieving this basic requirement that builds on the idea of "plug-and-play" introduced in Section 3.2. First, we will look at a simple example (Figure 8) of dynamic team formation.

In this figure, robot R_1 is initially responsible for unloading parts from machines M_1 and M_2 (i.e., shown at the top of Figure 8). At some point a second robot, R_2, is made available and is added to the work cell. Given the current state of cell controller technology, the following steps would typically be required to put this second robot on-line: (i) manually reconfigure the cell controller software off-line, (ii) shut down the work cell, (iii) download the new cell controller software, (iv) bring the work cell back on-line, and (v) debug the cell controller software (this may require several iterations of steps (i) to (v)). This approach is not only time consuming and error-prone, but also results in considerable added costs to manufacturers (both in terms of system down time and programming costs). As a result, there is often a reluctance to reconfigure systems when the opportunity arises (e.g., the reconfiguration costs may outweigh the savings resulting from increased productivity).

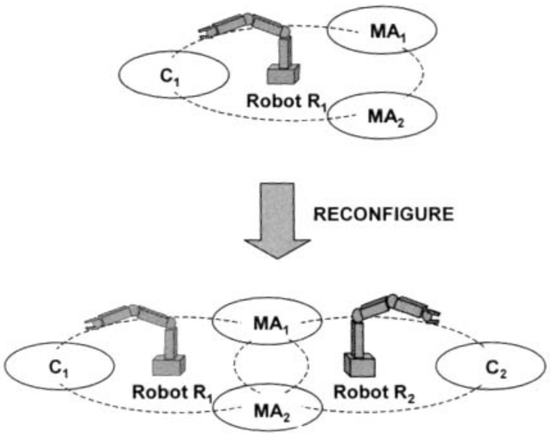


Figure 8. *A simple work cell*

In order to achieve the dynamically reconfigurable environment envisioned by the HMS community, we have been investigating extending the "plug and play" capabilities of Jini [NEW 01] to the manufacturing domain to achieve a "plug and produce" environment. For example, as is illustrated in Figure 9(a), Jini allows a federation of clients and services to be established that is facilitated by one or more look up services [NEW 01]. The basic idea is that services can be dynamically registered in a look up service; the look up service then acts as a broker/trader/locator between services and clients. An example of this is shown in Figure 9(b). In this case, robot R_2 registers its services (e.g., robot control) when it is plugged into the work cell. Clients (e.g., the cell controller) can then discover these services and use them to implement their plans. As is shown in this figure, the service object is really a proxy that communicates back to the service provider using *remote method invocation* (RMI) or some other equivalent protocol.

Unfortunately, the limitations of Jini in the real-time control domain are its memory requirements (typically > 1 Mbyte) and requirement of support for dynamic class loading and deserialisation (which some Java-based microcontrollers do not support). For our research, we have been investigating a small-footprint (i.e., < 100 Kbyte) version of Jini that has been developed by PsiNaptic Inc. [PSI 02] for use on real-time Java platforms such as Jbed [ESM 01] and TINI [LOO 01] that addresses both of these issues.

An example of the system described in this section is provided in Figure 10. In this case, we show the terminal interface to a PowerPC 823e (running RPX Lite) and our function block manager interface implemented on a web browser. The interface allows function blocks to be manually configured (i.e., both intra- and inter-function block connections) and the event and data connections to be tested. The focus of our current work is on extending this manual configuration capability to dynamic configuration using the RMI-based services described above. Our tests have shown that this approach allows small function block applications like the one described in this section to be quickly and easily configured and reconfigured at run time. Of course, to achieve a truly holonic system, as described in Section 1, integration with a configuration management application is the next step in this research.

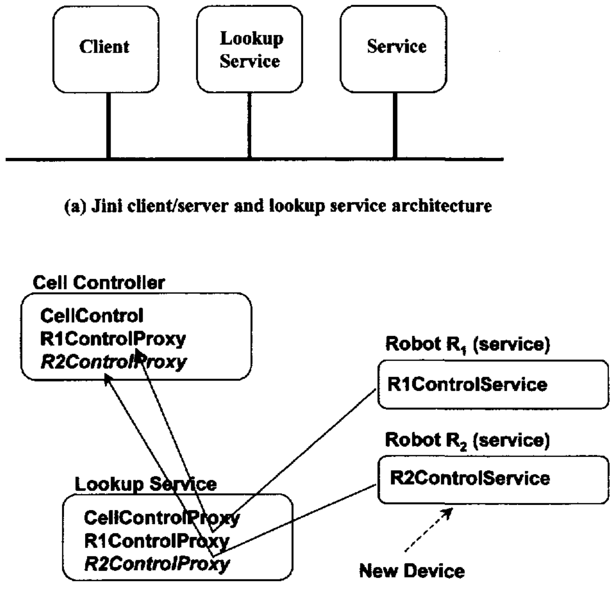
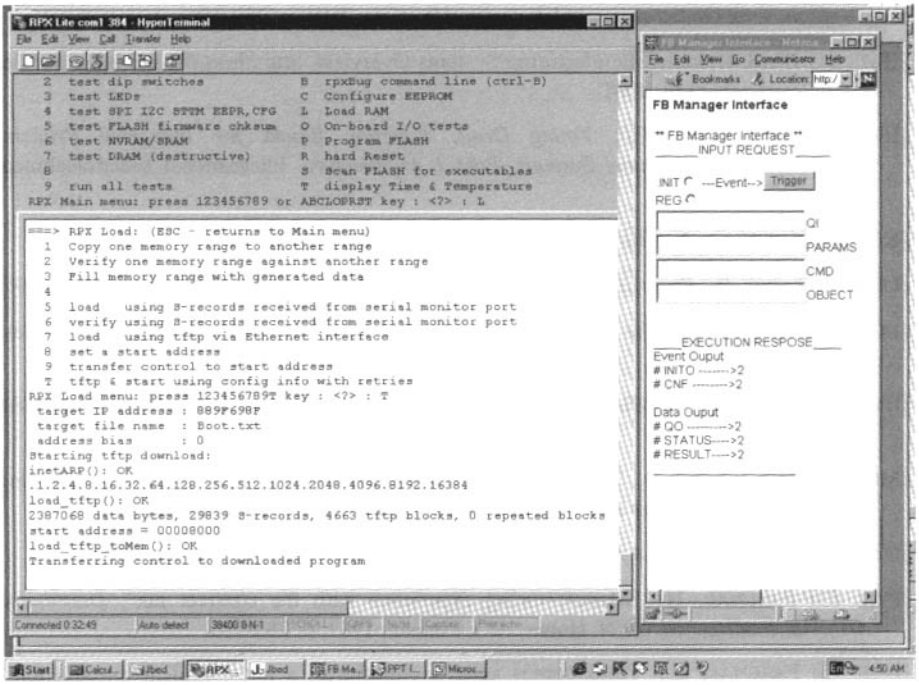


Figure 9. *Using Jini to enable dynamic coordination domains*

5. Summary and current work

In this paper we described two general approaches for dynamic and intelligent reconfiguration that are based on the IEC 61499 model for distributed intelligent control and report on our progress with an experimental prototype system. An important feature of these approaches is that they take advantage of the holonic characteristics (i.e., modular, recursive nature) of the IEC 61499 model to allow reconfiguration to be managed at the most appropriate level. As well, the general reconfiguration model proposed here can be applied using a layered, agent-based approach to allow fault detection and recovery to occur dynamically and automatically.

Our current work in this area is focused on the development of a conceptual architecture for configuration management. In particular, we are looking at how basic system functionality such as control application management and safety management (i.e., fault detection and recovery) is decomposed in a multi-layered, multi-agent environment as well as determining the nature of the agents used in this system. Work in this area will eventually lead to the development of a "holonic controller" that will take advantage of distributed artificial intelligence at the planning and control levels to achieve significantly shorter up-front commissioning times as well as significantly more responsiveness to change than current industrial control solutions.


RPX Lite com1 384 - HyperTerminal
File Edit View Call Transfer Help
Connected 0:32:49
Auto detect
38400 8-N-1
FB Manager Interface - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http:/
Trigger
Start

```
  2  test dip switches              B  rpxBug command line (ctrl-B)
  3  test LEDs                      C  Configure EEPROM
  4  test SPI I2C STTM EEPR,CFG     L  Load RAM
  5  test FLASH firmware chksum     O  On-board I/O tests
  6  test NVRAM/SRAM                P  Program FLASH
  7  test DRAM (destructive)        R  hard Reset
  8                                 S  Scan FLASH for executables
  9  run all tests                  T  display Time & Temperature
RPX Main menu: press 123456789 or ABCLOPRST key : <?> : L

===> RPX Load: (ESC - returns to Main menu)
  1  Copy one memory range to another range
  2  Verify one memory range against another range
  3  Fill memory range with generated data
  4
  5  load   using S-records received from serial monitor port
  6  verify using S-records received from serial monitor port
  7  load   using tftp via Ethernet interface
  8  set a start address
  9  transfer control to start address
  T  tftp & start using config info with retries
RPX Load menu: press 123456789T key : <?> : T
 target IP address : 889F698F
 target file name  : Boot.txt
 address bias      : 0
Starting tftp download:
inetARP(): OK
.1.2.4.8.16.32.64.128.256.512.1024.2048.4096.8192.16384
load_tftp(): OK
2387068 data bytes, 29839 S-records, 4663 tftp blocks, 0 repeated blocks
start address = 00008000
load_tftp_toMem(): OK
Transferring control to downloaded program
```

FB Manager Interface

** FB Manager Interface **
_______INPUT REQUEST_____

INIT ○ ---Event--> Trigger
REG ○

QI
PARAMS
CMD
OBJECT

____EXECUTION RESPOSE____
Event Ouput
INITO -------->2
CNF --------->2

Data Ouput
QO ---------->2
STATUS----->2
RESULT----->2

Figure 10. *The function block manager interface*

6. References

[BRE 01] BRENNAN R., NORRIE D., "Agents, holons and function blocks: distributed intelligent control in manufacturing", *Journal of Applied Systems Studies Special Issue on Industrial Applications of Multi-Agent and Holonic Systems*, vol. 2 no. 1, 2001, p. 1-19.

[BUS 98] BUSSMANN, S., "An Agent-Oriented Architecture for Holonic Manufacturing Control", *the First Open Workshop on Intelligent Manufacturing Systems Europe*, 1998, p. 1-12.

[DOU 99] DOUGLASS, B., *Doing Hard Time: Developing Real-time Systems with UML, Objects, Frameworks, and Patterns*, Addison-Wesley, 1999.

[ESM 01] ESMERTEC, "Jbed product line 'the only real choice for Java technology – small, fast and hard real-time'", *Technical report*, http://www.esmertec.com/, 2001.

[FLE 01] FLETCHER, M., BRENNAN, R.W., NORRIE, D.H., "Design and evaluation of real-time distributed manufacturing control systems using UML Capsules", *7th International Conference on Object-oriented Information Systems*, 2001, p. 382-386.

[HMS 01] HMS, Holonic Manufacturing Systems Overview. http://hms.ifw.uni-hannover.de/public/overview.html, 2001.

[IEC 00] IEC TC65/WG6, *Voting Draft: Function Blocks for Industrial Process-Measurement and Control Systems, Part 1 Architecture*. International Electrotechnical Commission, 2000.

[KOE 67] KOESTLER, A., *The Ghost in the Machine*. Arkana, 1967.

[LEW 96] LEWIS, R., *Programming Industrial Control Systems Using IEC 1131-3*, IEE, 1996.

[LOO 01] LOOMIS, D., *The TINI Specification and Developer's Guide*, Addison-Wesley, 2001.

[LYO 98] LYONS, A., "UML for real-time overview", *Technical Report of ObjecTime Ltd*, 1998.

[MCF 00] MCFARLANE, D., BUSSMANN, S., "Developments in holonic production planning and control", *Production Planning and Control*, vol. 11 no. 6, 2000, p. 522-536.

[NEW 01] NEWMARCH, J., *A Programmer's Guide to Jini Technology*, Apress, 2001.

[PAR 87] PARUNAK, H., "Manufacturing experience with the contract net", *Distributed Artificial Intelligence*, (M.N. Huhns, ed.), Pittman, 1987, p. 285-310.

[PAR 93] PARUNAK, H., "Autonomous agent architectures: a non-technical introduction". *Industrial Technology Institute Report*, 1993.

[PAR 99] PARUNAK, H., "Industrial and Practical Applications of DAI", *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, (G. Weiss, ed.), The MIT Press, 1999, p. 377-424.

[PSI 02] PSINAPTIC INC., *Jmatos Software User Manual for TINI 1.02d*, http://www.psinaptic.com/, 2002.

[SEL 98] SELIC, B., RUMBAUGH, J., "Using UML for modeling complex real-time systems", *Technical Report of ObjectTime Ltd and Rational Software Corporation*, 1998.

[SHE 99] SHEN, W., NORRIE, D., "Agent-based systems for intelligent manufacturing: a state-of-the-art survey", *Knowledge and Information Systems*, vol. 1, 1999, p. 129-156.

[SHE 00] SHEN, W., NORRIE, D., BARTHES, J., *MultiAgent Systems for Concurrent Design and Manufacturing*. Taylor and Francis, 2000.

[SMI 82] SMITH, R., "The contract net protocol: high-level communication and control in a distributed problem solver", *Defence Research Establishment Atlantic D.R.E.A. Report 80/1*, 1982.

[UPT 95] UPTON, D., "What really makes factories flexible?", *Harvard Business Review*, vol. 73, no. 4, 1995, p. 74-84.

[VAN 99] VAN BRUSSEL, H., VALCKENAERS, P., *Proceedings of the 2nd International Workshop on Intelligent Manufacturing Systems*. K.U. Leuven, 1999.

[WOO 99] WOOLDRIDGE, M., JENNINGS, N., "Software engineering with agents: pitfalls and pratfalls", *IEEE Internet Computing*, 1999, p. 20-27.

[ZHA 00] ZHANG, X., BALASUBRAMANIAN, S., BRENNAN, R., NORRIE, D., "Design and implementation of a real-time holonic control system", *Information Science Special Issue on Computational Intelligence for Manufacturing*, vol. 27 no. 1-2, 2000, p. 23-44.

This page intentionally left blank

Chapter 8

Intelligent Agents for Production Systems

Jihad Reaidy, Yingjiu Liu and Daniel Diep
Research Centre, Ecole des Mines d'Alès, Nimes, France

Pierre Massotte
Research Centre, Ecole des Mines d'Alès, Nimes, France, and IBM Academy of Technology

1. Introduction – main challenges in future production systems

This part details some trends and characteristics required for future and advanced manufacturing systems. They briefly lead manufacturing organisations to be more flexible in management and labor practices as well as to be able to develop, produce and re-manufacture virtually defect-free products and services quickly in response to opportunities and needs of customers. They can be identified as follows:

Engineering. With the advent of computing and information technologies and telecommunications, design, manufacturing process and the use of products and services have evolved. Also, in the recent years, the impact of computer-integrated manufacturing (CIM) has shifted from the traditional factory integration philosophy to a virtual factory management philosophy (Lee, 1996). These technological advances enable one to achieve a fast and highly collaborative environment based on i) multi-media software engineering tools and highly reliable communication systems for facilitation of distributed procedures of concurrent design and development of products and services, and ii) operations indistributed production systems.

Environment and ecology. Recycling involves the settling and reusing of materials. They can be provided from returned products (new, repaired, restored or rebuilt, etc) This implies introduction of a fashionable concept around environmentally conscious design. Compared with the well known approach called *design for manufacturing* it covers many more constraints since it means design for re-use of assemblies and parts, for maintenance, for easy dismantling and/or disassembly, and for longevity (with new concepts such as self-maintenance, self reparability, error recovery, etc.). These items are all included in the GNOSIS program (GNOSIS-VF, 1998).

Process and supply chain management. Industry is now faced with a global and distributed economy. Worldwide competition and mergers reduce margins. As might be expected this has had a great impact, not only on the design of product and processes but also on social and logistic systems:

– how to improve the procurement and the circulation of materials and parts and how to share the manufacturing of complex end products;
– how to organise the scrap, reclaims, wasting and recycling of products and goods;
– how to develop new opportunities of jobs and enhanced conditions of working;
– how to take into account economical added value of this new logistic and to find the most economical balance to control and manage the resulting disturbances in terms of flow of products.

Evolution of the production requirements. Reduction of manufacturing costs and improvement in customer satisfaction led to a paradigm shift from a focus on "quality of product design and manufacturing" to a focus on "quality of service" with decreasing delivery time. As a result, reactivity and flexibility of production systems become of key importance. Thus, research focus on "service" issue is better addressed in the design and manufacturing enterprise. This approach is tightly related to "lean production" which is the minimisation of costs resulting from product variability, in particular the times required to change dies and the size of inventories (just-in-time (Flik, 1995)). Companies adopting lean production approaches (more suited than the 7-zero's approaches) have achieved substantial reduction in costs resulting from product variability or specific demand changes. In most cases, it was demonstrated that lot sizes or batches had to be reduced as much as possible, i.e., to increase the number of product variants to be able to meet the customer requirements in every segment of the market. Now, the trend to one-of-a-kind-production (OKP) has been termed "agile manufacturing"; it is much more constraining than "lean manufacturing" and requires the implementation of new paradigms in terms of reactivity and flexibility.

E-Business. E-business, irrespective of disturbances presently observed at stock exchanges, is evolving towards more activity. The net-economy is characterised by "speed". Each time demand occurs, it is necessary to quickly specify when, at what cost, in which quantity the order will be delivered. Here, response time is the key. Moreover, considering the high volume of individual and specific demands, with many product variations, mass customisation is required for such production systems. These characteristics combined imply a great reactivity. At last, because of logistics reasons associated with these above constraints, the role of the customer is becoming pervasive. He is progressively involved in any decision-making along the manufacturing value added chain.

In this paper we focus on the development of new approaches devoted to the monitoring and control of complex production systems submitted to these constraints. They are based on interactions between intelligent agents operating at product and process level in the distributed production system. We will try to define the right coupling, adequate task assignments and the best fitted adjustment of product and process parameters to perform the dynamic reconfiguration of the production system.

2. Inverse solutioning approaches

2.1. *Main concept*

As a consequence of the aforementioned, and taking into account the actual difficulties encountered in managing such distributed production system (DPS), a new approach has been studied. The behaviour of a production system evolves from its structure with knowledge, or functional mechanisms, embedded in the interactions between resources (modelled by so-called "agents"), rather than as the direct result of a predefined and complex given function; thus, the way to manage a production system is totally different of what we usually consider in industry. As seen in Figure

1, we can compare the two functional approaches, based on the example of the PABADIS model, which is detailed in section 5.

On the left hand side of Figure 1: we can observe a deterministic and conventional organisation of the work; the knowledge /know-how and associated processing are hierarchically structured in specific modules and layers. Here, we notice enterprise resource planning (ERP), manufacturing execution system (MES), supervisory control and data acquisition (SCADA), man-machine interface (MMI) and programmable logic controllers (PLC's). The demand and/or customer orders have a single entry point at the top of the graph and the production is organised in a top-down way.

At the right hand side of Figure 1: there is a de-coupling between ERP and the operational level including the same SCADA and PLC modules. The links between the modules are suppressed since they are much more autonomous; each one is communicating with each other. However, for practical purposes we can define a neighbor local exchange. Here, the operating model is different; cooperation, competition and/or collaborative works are the basic principles.

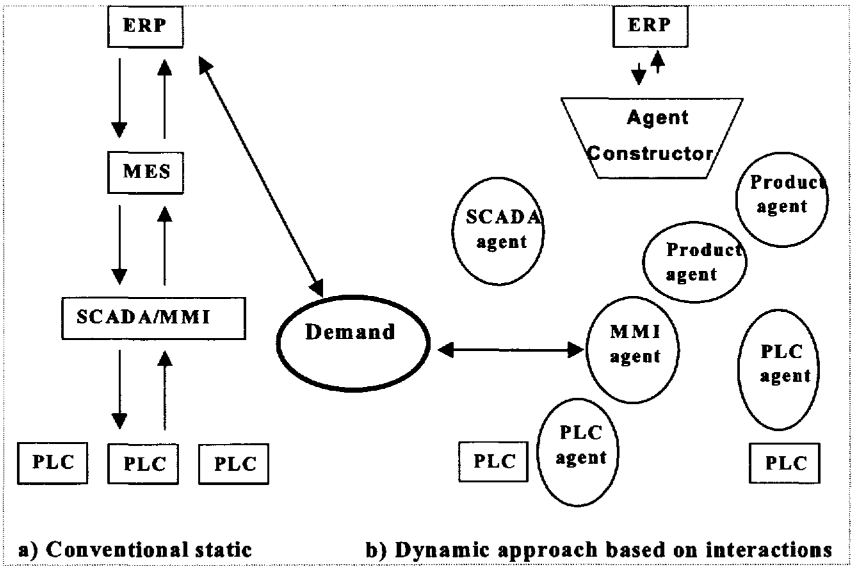


Figure 1. *Basic approaches involved in production systems*

If we now address some basic logic mechanisms, as required by the previous approaches, we may consider the two situations (see Figure 2), where the interactions between the different items play a very specific role. Again, as per a logic point of view, the way of reasoning is quite different:

– The conventional approach is based on static functions: it consists of analyzing the complete system, to detail its main tasks (decomposition principle), to simplify the process and to automate it with the help of computers, and to execute these functions in parallel if possible to gain some performance. The chaining of the functions is organised in a top-down way.

– Dynamic approach; autonomous and communicating multi-agents are considered. This concept consists of generating global functions from interactions existing in interconnected networks. Such a concept is totally different and can be expressed in a bottom-up way. This is why a different graph has been designed. Because, for example, auctions and negotiations between agents directly generate the detailed scheduling, a scheduler is no more necessary. Also, in terms of resources assignment, a production configurator as usually designed is no longer required, except for presizing of the production system; moreover the intelligence in any conventional approach is quite localised at process level. Now, with this extension the intelligence can be distributed at product and process level or distributed at process and logistic/interaction level.

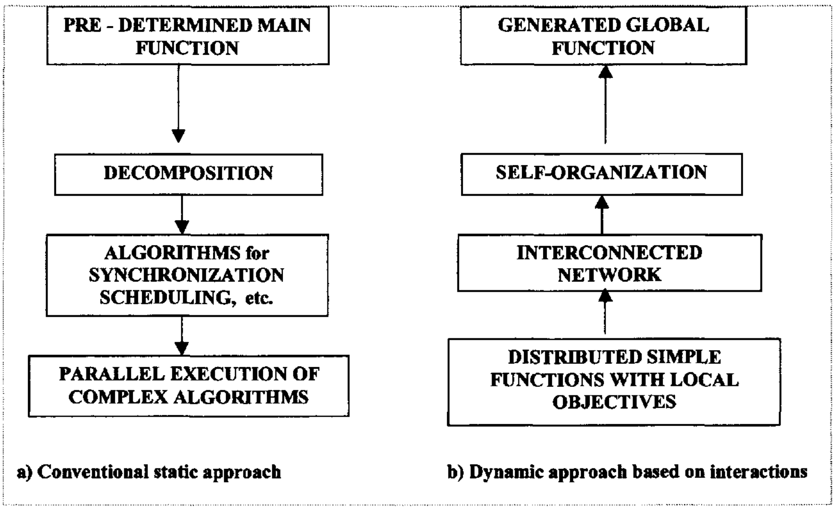

PRE - DETERMINED MAIN FUNCTION
DECOMPOSITION
ALGORITHMS for SYNCHRONIZATION SCHEDULING, etc.
PARALLEL EXECUTION OF COMPLEX ALGORITHMS
a) Conventional static approach
GENERATED GLOBAL FUNCTION
SELF-ORGANIZATION
INTERCONNECTED NETWORK
DISTRIBUTED SIMPLE FUNCTIONS WITH LOCAL OBJECTIVES
b) Dynamic approach based on interactions

Figure 2. *Mechanisms involved in production systems management*

2.2. *What is the scheduling problem?: capacities optimisation or configuration/ reconfiguration?*

In the area of monitoring and management of complex and distributed production systems, the specialists often address one problem: "How to perform a detailed scheduling with limited capacities?". Within this context, we will focus our attention and skills on the scheduling problem and we will develop complex solutions, which will give obsolete results when it will become necessary to implement them. Another reasoning way consists of focusing on the capacities: "if we are able to reconfigure permanently the resources and to adapt them to the demand, there is no scheduling problem!" Thus, we have decided to address this second problem since the scheduling problem is in fact a configuration/reconfiguration problem.

Configuration and reconfiguration are now considered as a major improvement to be implemented in the area of organisation and setting-up of a production system. Some approaches have been proposed in the framework of plant automation based on distributed systems (PABADIS). PABADIS is an European IST- 60016 project, it is intended to implement innovative techniques in distributed manufacturing systems, as with flexible manufacturing systems (FMS) (PABADIS, 2001). The approach consists of dynamically assigning resources and organises production in order to have the "best" flow of parts/products (physical flow).

This approach is interesting since it is aimed at focusing and finding dynamically the best configuration, structure and environment to face up to the problems related to distributed manufacturing, scheduling and production on demand. A well-known technique used for that kind of problem solving is based on simulation. PABADIS is still in progress and will include these above concepts, based on intelligent agents.

3. Design and development of the new approach

3.1. *Problem assessment*

Traditional production systems models are strongly centralised, which results in large and complex software, difficult to upgrade and maintain. Often, a detailed schedule is generated over a long time horizon. Also, planning, scheduling and execution are carried out sequentially. Such systems are not able to adapt to changing circumstances over time (such as machine breakdown, rush-orders etc.). The model to be used today must be able to support a flexible, reactive and economic manufacturing process. Indeed, in a facility, the flexibility is the result of several readjustments and reassignments of manufacturing resources, which vary in time. Thus, the corresponding production systems require adaptable and autonomous control systems, which can support, in a dynamic way, a reconfiguration or a readjustment of operational parameters.

Using multi-agents system (MAS) is a solution for the problems as expressed above. Such systems aid working closer to reality while using mechanisms of artificial life (co-operation, bidding, game theory, etc.); this gives more autonomy to the agent (decentralisation of the decision system, etc). The objective of our works is to design and develop such MAS, to define operating modes and adapted algorithms (as defined in the previous section), finally to implement "intelligent agents" (Liu and Zhong, 1999) to take up the challenges of the industrial world of tomorrow.

3.2. *Generalities on multi-agents system*

An agent is a software entity that represents one or more (functional or physical) components within a production system (such as a machine or customer order). An agent is authorised to carry out local activities and to make plans for the components that it represents. Agents have the ability to observe their environment and to communicate with other agents in order to resolve conflicts and make agreements. An agent should be seen as an autonomous intelligent controller/planner for a unit within a dynamic system with the ability to coordinate its actions with those of other agents. A system that consists of more than one agent is called a multi-agents system.

In contrast to traditional software systems, MAS are distributed, parallel and autonomous. The basic approach behind MAS is to decompose a complex problem into a number of (less complex) sub-problem. Each sub-problem falls under the responsibility of an agent. Since sub-problems are interrelated, a co-ordination mechanism is applied to ensure that the local decisions lead to a globally desirable result. This approach leads to a modular and flexible software solution that is capable of reacting to disturbances while keeping a view of longer-term goals. According to agent characteristics required in the area under study, we have to consider cognitive or reactive agents and, therefore, cognitive or reactive systems; often, a reactive system includes a great number of agents of low granularity (moreover low level) which have a protocol and a reduced communication language; their capacities respond to the stimulus/action law.

Generally speaking a cognitive system is composed of a small number of "intelligent" agents having a base of knowledge or experiments, including understanding the whole of information and know-how necessary to the realisation of its task; it will also cover the management of the interactions between the agents and their environment (Ferber, 1999).

Such an agent can be considered as a "situated" agent. It is means that it is able, starting from its own experiment, to determine what is significant for it while interacting with the current situation. It does not require elaborated internal models of its world to plan its actions, since the real world is a part of the agent knowledge. From this point of view, the agent and its world are specified mutually ("its" and not "the", because the world of an agent is the world of its experiment) (Drogoul and Meyer, 1999).

3.3. *Example of a distributed manufacturing management system*

Here, we briefly address the design and the development of a prototype that is able to integrate some of the above detailed concepts. This tool called the *virtual factory dynamics configuration system* has been developed by the LGI2P to anticipate and to solve some predefined problems in supply chain management. (See GNOSIS project (GNOSIS-VF, 1998). In this tool, intelligence is distributed at product and process level. Its implementation only concerns software tools for improving the management and the monitoring of a DPS (see Figure 3).

This approach, based on agents, enables one to improve the configuration and reconfiguration of a production system, and the product flow-accordingly, in assigning the resources through self-organisation mechanisms.

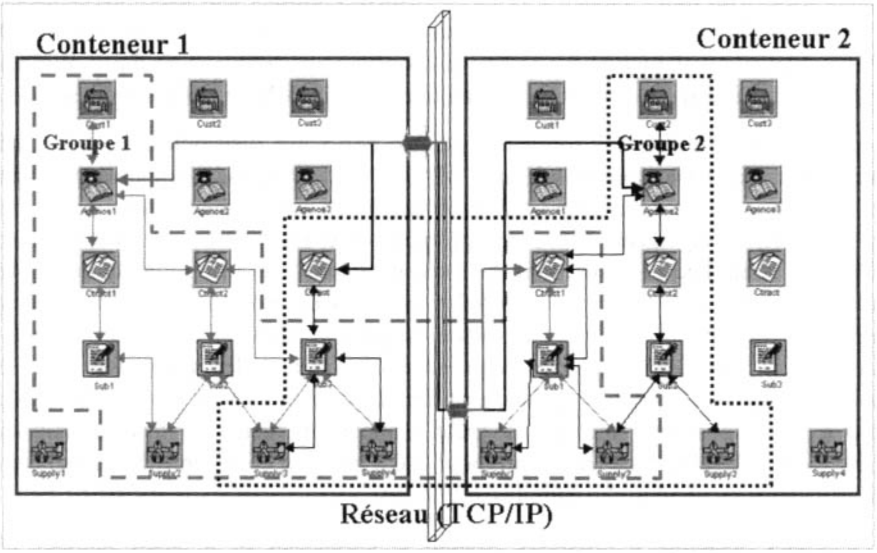


Figure 3. *Configuration of a virtual factory with VFDCS*

3.3.1. *Characteristics and specificity of VFDCS*

The design and architecture of VFDCS (Liu and Massotte, 1999) will not be described in this paper since the focus is brought on to upgrading VFDCS though the implementation of "intelligent agents". The VFDCS model is an innovative workbench, derived from HIVE (Minar et al., 1999) and JAFMAS (Deepika, 1997), integrating several promising design techniques: supply chain management, pull production control technology, internet-based e-commerce, auction based on reasoning with rules and coordination based on contact net protocol.

The entities (components) of a virtual factory are usually working with pre-qualified suppliers and do not rely on auctions to get the commodities or services they need. So, at the same time, we adopt auction and coordination techniques. Auction mechanism is a promising method to resolve distributed resource allocation problems characterised by self-interested agents and scarce resources. Many different types of auctions are in common use: English open-outcry auction to sell art and other collectibles, Dutch auction to sell perishables, first-price sealed bid

(FPSB) and Vickrey auction for procurement situations and continuous double auctions (CADs) for trading securities and financial instruments (Friedman and Rust, 1993; McAfee and McMillan, 1987; Milgrom, 1987).

One of the most difficult problems an agent faces in dealing with negotiation auction over complex plans, is the problem of evaluating bids. The agent must solve both bid-allocation and temporal feasibility constraints, while attempting to minimise cost and risk. Here, we have implemented a pair of buyer-seller agent that uses both forward-chaining and backward-chaining algorithms, based on if-then rules, to process the auction. It is key to find an appropriate tradeoff between systematic optimisation and random exploratory behaviour. No individual agent has sufficient competence, expertise, resources, or information to solve the entire problem in a multi-agents universe (Jennings, 1995).

The contract net protocol is a negotiation protocol proposed by Smith (Smith, 1980). This protocol facilitates distributing subtasks among various agents. The agent which wants to solve a problem broadcasts a request for bids, waits for an answer for a certain length of time, and then awards a contract to the best offer(s) according to its selection criteria. VFDCS can provide decision support, from simple buying and selling of goods and services to complex multi-agents contract negotiations. VFDCS is designed to negotiate contracts based on temporal, quantity, price and precedence constraints, and includes facilities for dealing time-based contingencies.

3.3.2. *Features included within VFDCS*

The VFDCS workbench has the following advantages:

- VFDCS is associated with high flexibility and scalability concerning the configuration and organisation.

- VFDCS is easy to be maintained because of its modular scaleable architecture and design patterns programming mechanism.

- Integration of electronic commerce and internet-based technology because of implementing Java's connect remote service (RMI) and multi-threads application.

- VFDCS is suited for customer-oriented and market-driven new organisation paradigm because of integrating "PULL" control technique just in time.

- VFDCS is easy to integrate existing software and hardware to resolve the legacy problem because of a separate user-friendship interface package layer with debugging capabilities.

Several aspects of our workbench warrant further investigation. Our current work directions include:

- Implementation of a real VFDCS integrating e-commerce and web-based internet standard in a real manufacturing factory.

- Development of process to simulate continuous manufacturing and trading.

Incorporation of more adaptive factory agents that are capable of modifying their control policies during simulation based on evolving circumstances. Generally speaking, this conducts to in-depth study the implementation of "intelligent agents".

4. Proposed hybrid architecture for this new approach

In this section we describe improvements under implementation in VFDCS for PABADIS. We focus on the design and architecture of the so-called "intelligent agents".

4.1. *Main concepts at agent level*

Information to be processed in a distributed dynamic environment such as supply chain management is often heterogeneous (quantitative /qualitative); it requires some co-operative, differentiated and complex data processing. Suitable agents become more intelligent and require the implementation of hybrid technologies to integrate enhanced capabilities. For that, a new distributed architecture is developed in our study and will integrate several concepts (like case based reasoning, self organisation algorithms, neuronal/ genetic classifiers), it is aimed at designing more evolutionary, adaptive, flexible and reactive agents with the use of a very simple and more rapid algorithms.

Indeed, the main objective of our architecture is to demonstrate the interests of new paradigms and the efficiency of self-organisation mechanisms as well. These agents will evolve overtime to ensure more autonomy and to cover more functionality. Several schemes, and architectures, can be proposed to design an intelligent agent (see Figure 4). Three different and independent cases have been considered:

Case (C): Here, the agent is mainly a cognitive one and treats qualitative information using mechanisms such as case based reasoning (CBR), knowledge base system (KBS), game theory (GT), etc. with qualitative reasoning. Considering the nature of knowledge to be processed and the time required to collect and formalise the expertise relevant to the decision making process, a CBR seems to be more suited to model the IADSS than a KBS. During the operational activity of the CBR, different cases will be experienced and the "prototypes base" will be enriched and validated progressively. Very quickly, genetic/neuronal classifiers tools (GC/NC on top of the CBR) will be able to be correctly trained (Shaw and Whinston, 1989); our past experience showed that learning by reinforcement is possible (reward/penalty mode). Moreover we can include some negotiation strategies like game theory (GT) for revolving problems of negotiation and communication between agents (buyer/seller). These different concepts can be modelled by genetic/neuronal classifiers (GC/NC).

Case (A) and Case (B) address a reactive agent, specially built with some basic artificial neural networks (ANN) or genetic algorithm / swarm algorithm (GA/ swarm) to model the functioning of a given entity (Bonabeau et al., 1999). Generally, reactive agents treat quantitative information using mechanisms such as stimulus-response, self-reinforcement, etc., with computational heuristics and optimisation algorithms. So, depending on the problem to be solved, one of these three architectures will be selected to set up the agent. As a result, this intelligent agent will be more reactive and able to process hybrid information.

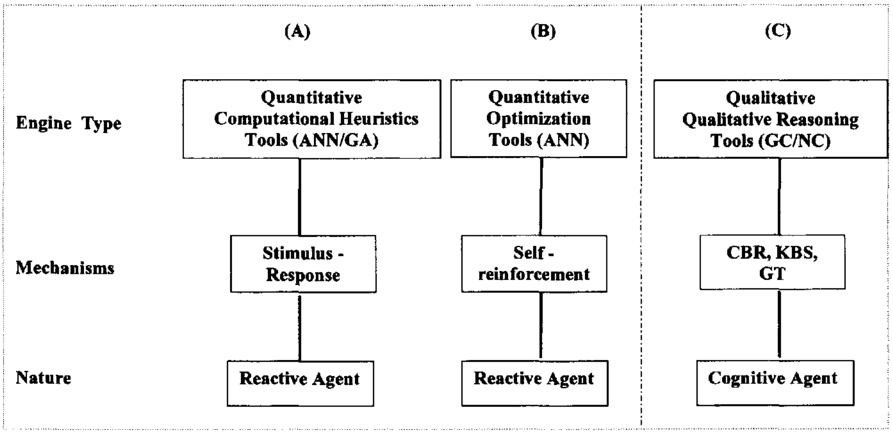

(A)
(B)
(C)
Engine Type
Quantitative Computational Heuristics Tools (ANN/GA)
Quantitative Optimization Tools (ANN)
Qualitative Qualitative Reasoning Tools (GC/NC)
Mechanisms
Stimulus - Response
Self - reinforcement
CBR, KBS, GT
Nature
Reactive Agent
Reactive Agent
Cognitive Agent

Figure 4. *Logic structure of three kinds of intelligent agents decision support system (IADSS)*

4.2. *Implementation approach of intelligent agents*

In a MAS environment, different types of cognitive and reactive agents, as described above, can be used separately or at the same time and many communicate with each other. This depends both on the application and the problem to be solved. In applications where problems seem uncomplicated such as travelling sales-man problems or others, the use of reactive agents alone using a self-organisation algorithm can solve the problem, quickly and easily. On the other hand, resolving complex problems which imply some intelligence represented by knowledge about the functioning of a particular application acquired during some time or existence of constraints during the process between the different entities require the use of a cognitive agent. In this case, for instance, the use of the two types of reactive and cognitive agents at the same time seems to be useful and could represent a good solution.

Such agents are differentiated by their identity and the task to be achieved; they distribute tasks between them according to their capacity and the complexity of the action to be carried out. For example, in a production system we can represent the requests from the customers, and the product, by reactive agents; the resources may be cognitive agents. Then reactive agents can communicate with cognitive agents and other reactive agents for simple tasks (to accept / refuse) manufactured by one of these resources or for exchanging information between them (see Figure 4). The resources take in charge all problems related to communication with another one to comply with the various constraints of manufacturing of the product and with the generation of the number of reactive agent for the achievement of certain aims, and then using reinforcement algorithms and classifier systems for learning (Shen et al., 1998) and game theory or local rules (CBR, KBS) for strategy determination for communication between the agents.

Figure 5 details the solution we are implementing in the European project called PABADIS. The graph represents the overall architecture and functioning mode of the MAS (e.g. VFDCS), used for reconfiguring and assigning tasks in the distributed production system. Each product and/or resources is an agent. The relationships between the agents are communication links devoted to messages and information exchanges. They are already defined and implemented in VFDCS. The agents included in this workbench are now simple ones. The more sophisticated ones to be embedded are those described in this paper.

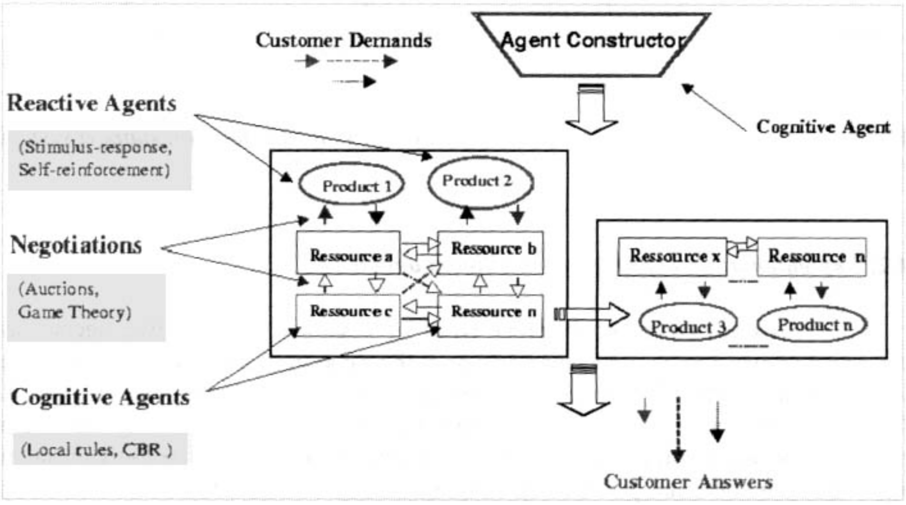


Figure 5. *Dynamic approach based on interactions between reactive/cognitive agents*

5. Industrial application: the PABADIS model

5.1. *Main concepts*

The aim of the PABADIS IST- 60016 project is to overcome the problems of centralisation by distributing as many functions as possible within the plant automation system at the operation level. Two principles are combined for achieving this goal:

- decentralisation with agents,
- dynamic reconfiguration.

In the following, we will not describe the rationale behind the implementation of the multi-agents system in PABADIS. We will only focus our attention on the architecture of the kernel to be implemented in order to get agents working properly together. More details can be found in (PABADIS, 2001), (Sauter and Massotte, 2001).

Several industrial partners are involved in this European project, which started at the end of year 2000. Some parts of concepts and tools developed in this paper are planned to be implemented in PABADIS in 2002.

5.1.1. *Decentralisation with agents*

Figure 6 shows how MES (manufacturing execution system) and SCADA (supervisory control and data acquisition) are linked to the ERP (enterprise resource planning) and the controls layers in the conventional and in the PABADIS model. Decentralised and autonomous MES and SCADA functions will be realised by a population of agents, the synchronisation of processes like allocation, routing and scheduling being performed by means of communication acts between agents.

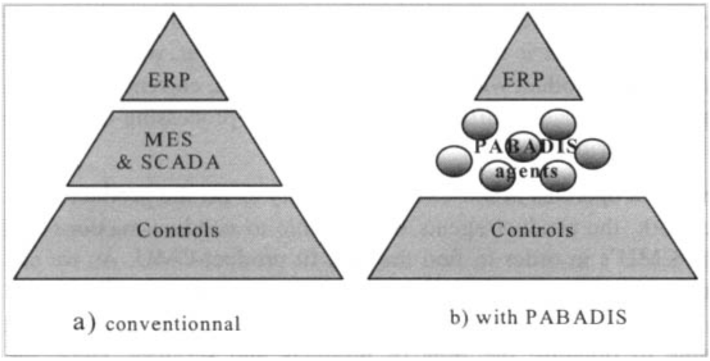


Figure 6. *CIM pyramid of automation*

5.1.2. *Dynamic reconfiguration*

Within the PABADIS terminology, the actual production plant is an abstract set of so-called CMUs (cooperative manufacturing units) that are linked by a communication network and that offer certain services and resources (Figure 7). CMUs could be anything from a single tooling machine to a complete production line, but also a simple computer offering special computational services. At any time a new CMU may join or leave the community without any change for others, according to a plug-and-participate mechanism.

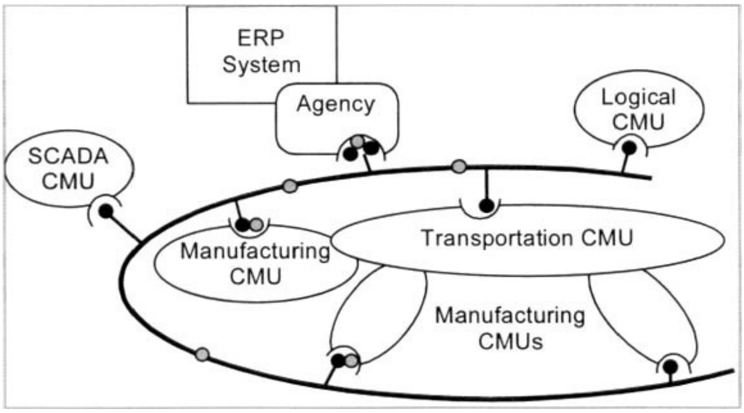


Figure 7. *Community of CMUs*

5.2. *Operations*

The general operational scenario of the PABADIS plant is roughly as follows (see Figure 8): The ERP system creates a production order with information about the product to be made. This order is converted into a product agent, which contains a technical description of the product with processing recipe data, current processing state data, done processing data, further schedule data, processing-dependent machine data and a task description.

The product agent then operates in the CMU community to get the product done. It is a collaborative work: the product agents will be able to conduct auctions and negotiations with the CMU's in order to find the best fit product-CMU. As we can be seen, the conventional MES part of the process, which operates the distributed production system, is replaced by a network of autonomous and communicating agents which are able to organise the flow of products and services. Here, the resources and the products as well are involved in the self-organisation mechanisms. A similar reverse scheme will be used to collect data and send feed-back information to the ERP function.

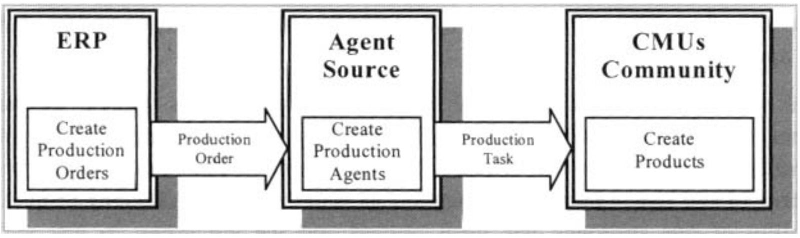


Figure 8. *Production process in PABADIS*

5.3. *Agents in PABADIS*

Two types of agents are present in the PABADIS system:

A *product agent* (PA) is a mobile agent created by the agent source and based on a production order issued from the centralised ERP system. Each product agent is a piece of code consisting of an execution program (tasks calculation, negotiation mechanism, safety, security, etc.) and the specific product data, operating procedures, bill of materials, etc. It migrates in the network independently to solve its task (this is, to "create" the product, and to add some value to it). The task for each product agent may be drawn from several subtasks with all possible mutual dependencies (time dependence, sequence, priority, etc.).

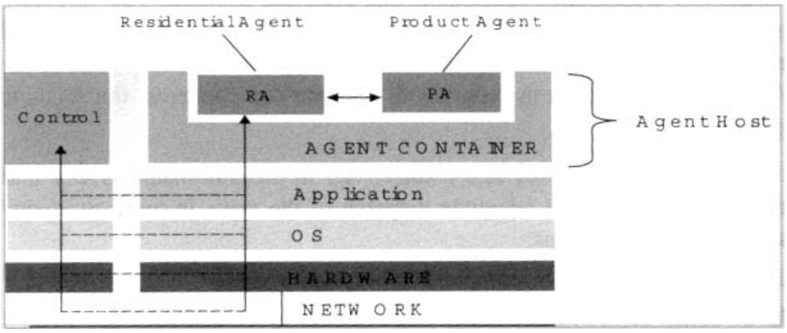


Figure 9. *Structure of a CMU*

A *residential agent* (RA) is a part of every CMU. The main function of the immobile residential agent is to provide the connection between the plant network (including other agents) and the particular resources offered by the CMU (such as a PLC – programmable logic controller, CNC – computer numerical control, or an automation controller of any kind). The residential agent acts as a gateway between the agent community on one hand and a large variety of concrete production facilities on the other. Figure 9 shows how a CMU is structured to host mobile and residential agents.

This architecture which is based on multi-agents system was subject to the development of several prototypes which have been applied to the improvement of the demonstrator available at the LGI2P Research Center. This demonstrator is representative of the real industrial application as stated in the PABADIS project. The feasibility of our approach has been proved and we are going to test and evaluate them deeply on a real production system. In our point of view, implementation of these concepts must be associated with a methodology and reorganisation of operating procedure; this work is in progress. This aspect is very important since we implement advanced technology without considering their organisational and social impact in the enterprise.

6. Conclusions

In this paper we have described a new approach devoted to the management and control of distributed manufacturing systems, and based on self-configuration mechanisms (VFDCS). To improve this approach we carried out some work in designing and developing an hybrid architecture for implementing "intelligent agents" as a smart solution for the reconfiguration problem raised in a supply chain management. This architecture is composed of reactive and cognitive agents modelled through self-organisation algorithms and classifier systems. This structure is implemented in a workbench, called VFDCS, to improve its capabilities and performance, and, to enhance the autonomy and the ability (intelligence) itself of the agent. Development and tests are in progress to validate our concepts; the focus is specifically put on:

- the optimum level of autonomy and connectivity in programmable graphs and therefore the determination of the adequate dispatch of the tasks in multi-products/ multi-processes production systems,

- the control, cooperation and/or competition between agents,

- the role of some centralised control mechanisms to consolidate the overall performance of the new MES.

Finally, we have to mention that fundamental changes have to be introduced into our industry, into our culture and way of working to integrate these paradigms into our management and decision making processes. Some of these concepts are being introduced in the PABADIS project which is intended to cover such new functional needs, and can be considered as a heart of a logistics composite modelling system.

7. References

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to artificial Systems*. Oxford University Press.

Deepika, C. (1997). JAFMAS: A Java-based Agent Framework for Multiagent Systems Development and Implementation. ECECS Department Thesis, University of Cincinnati, Cincinnati, OH.

Drogoul, A. and Meyer, J. (1999). *Intelligence artificielle située*. Hermès Paris Science Publications.

Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Pub Co.

Flik, M. I. (1995). Optimization of Products Variants – Lean Standardization at a Radiator Supplier. Internal Report, Behr Automotive.

Friedman, D. and Rust, J. (1993). *The Double Auction Market: Institutions, Theories, and Evidence*. Addison-Wesley Publishing, Reading, MA.

GNOSIS-VF, (1998). *Virtual Factory*, http://www.cordis.lu/esprit/src/28448.htm

Jennings, N. R. (1995). Chapter 6: *Coordination Techniques for Distributed AI, Foundations of Distributed Artificial Intelligence*. G.M.P.O'Hare and N. R. Jennings (Eds), John Wiley & Sons, pages 187-210.

Lee, j. (1996). Overview of Manufacturing Strategy, Production Practices, Emerging Technologies, and Education System in Japan. NSF/STA Study Report.

Liu, J. and Zhong, N. (1999). *intelligent agent Technology: Systems, Methodologies, and Tool.*, World Scientific Publishing.

Liu, Y. J. and Massotte, P. (1999). Self-adaptation and Reconfiguration of an Agent-Based Production System: Virtual Factory. *IAT'99: Asia Pacific Conference on intelligent agent Technology*, Honk-Hong, Chine.

Massotte, P. (1997a). *Analysis and Approaches for the Management of Complex Production Systems.* The Planning and Scheduling of Production Systems, Edit by Artiba A. and Elmaghraby S.E., Chapman & Hall.

Massotte, P. (1997b). Application of Self-Organization Principles to System Control. IFACS Conference, Grenoble, France.

McAfee, R. P. and McMillan, J. (1987). Auctions and bidding. *Journal of Economic Literature*, 25: pages 699-738.

Milgrom, P. (1987). *Auction theory*. In Dewley, T. F., editor, Advances in Economic Theory: Fifth World Congress, Cambridge University Press.

Minar, N., Gray, M., Roup, O., Krikorian, R. and Maes, P. (1999). Hive: Distributed Agents for Networking Things, *Proceedings of. ASA/MA '99.*

PABADIS, (2001). Plant Automation BAsed on Distributed Systems, http://www.pabadis.org/.

Sauter T., Massotte P., Enhancement of distributed production systems through the Attachment of Agents to Fieldbus Networks, *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Nice, France.

Shaw, M. and Whinston, A. (1989). *learning and adaptation in DAI systems*. Distributed Artificial Intelligence, volume 2, pages 413-429, Pittman Publishing/Morgan Kauffmann Publishers.

Shen, W., Maturana, F. and Norrie, D. (1998). Learning in Agent-based Manufacturing Systems. *AAAI* Press, pp.177-183.

Smith, R. G. (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, C-29 (12), pages 1104-1113.

Index

agent-based
  manufacturing 129
  scheduling
    agile system 87 et seq
    algorithm 93, 105
      negotiation 97
    framework 91
    literature 90
    negotiation model 93
agents based modelling, software 121
assembly 88
  sequence, complex 88
  simple 88
attribute layer 10
attributes 117

basic configuration problem 54
building
  context, data exchange 73
  product models 1 et seq
    procedure 4

CAD tools, communication and 74
cell structure 34
central configuration problem 55
class and object layer 9
co-design process, distributed design process and 120
communication, CAD tools and 74
complex assembly sequence 88
component
  locations, layout with
    absolute 62
    relative 63
  quantity 60
  relative locations, layout with 63
concurrent engineering environment
  new product development 109 et seq
configuration
  control
    application 139
    services 138
  definition 51
  problem
    basic 54
    central 55
  product generic modelling for
    requirement analysis,
      modelling elements and 49 et seq
configurator 50, 51
  presentation 51
constraint satisfaction problem (CSP) 50, 53
  approach 53
contingencies approach, reconfiguration to 136
continuous variable, discrete and 60
  numerical constraints 60
customised products 88
custom storage system 50

data exchange, building context 73
decision support system, help interface 76
DeSAP
  interface, e-OCEA environment

scheduling problems 27 et seq
module, global view 28
design for manufacturing 148
digraph 88, 89
discrete variable 60
continuous and
numerical constraints 60
distributed
artificial intelligence (DAI) 90
design process and
co-design process 120
manufacturing management 153

e-business 149
ecology, environment and 148
element of knowledge, modelling of 116
environment, ecology and 148
e-OCEA 28
environment, DeSAP interface
scheduling problems 27 et seq

function block architecture 131
future production systems 148

generic 49 et seq, 52, 54
model
expressivity 52
testability 52
modelling
elements, product modelling requirements and 54
product, configuration for requirement analysis, modelling elements and 49 et seq

help interface, decision support system 76
hierarchical bill-of-materials 65
holonic manufacturing systems 129
holons 130

IEC 61499-based model 127 et seq
real-time distributed control systems, reconfiguration of 127 et seq
implementation approach, intelligent agents of 157
intelligent agents
implementation approach 157
manufacturing systems (IMS) 129
production systems for 147 et seq
inverse solutioning approaches 149

knowledge integrated product 3

layer
attribute 10
class and object 9
method 10
structure 10
subject 9
layout 62
component locations
absolute 62
relative 63
line structure 34

manufacturing
agent-based 129
management, distributed 153
system, {m,q} 88
systems
holonic 129
intelligent (IMS) 129
method layer 10
mobile software agents 117
mobility 118
model
expressivity, generic 52
IEC 61499-based
real-time distributed control systems model, reconfiguration of 127 et seq
negotiation, agent-based scheduling 93
PABADIS 159
testability, generic 52

modelling
  element of knowledge 116
  elements
    generic, product modelling, requirements and 54
    requirement analysis and 49 et seq
  language, virtual reality (VRML) 80
  product generic, configuration for
    requirement analysis, modelling elements and 49 et seq
  requirements, product
    generic modelling elements 54
  software agents based 121
models
  building product 1 et seq
    procedure 4
  product related 3
{m,q} manufacturing system 88

negotiation
  agent-based scheduling algorithm 97
  model, agent-based scheduling 93
new product development
  concurrent engineering environment 109 et seq,
numerical constraints
  discrete variable
    continuous and 60

object
  layer, class and 9
  oriented
    analysis (OOOA) 9
    design 12

PABADIS model 159
"plug-and-produce" capabilities 140
problem assessment 152
process
  analysis 6
  distributed design and co-design 120
  supply chain management 148
product
  analysis 7
  development, new
    concurrent engineering environment 109 et seq
  generic modelling, configuration for
    requirement analysis, modelling elements and 49 et seq
  knowledge integrated 3
  master 8
  modelling requirements
    generic modelling elements 54
  models
    building 1 et seq
    related 3
production
  systems
    future 148
    intelligent agents for 147 et seq
    management 71 et seq
products, producing customised 88
programmable logic controller (PLC) 131
prototype implementation 137

qualities 17

real-time distributed control
  systems, reconfiguration of 134
    IEC 61499-based model 127 et seq
reconfiguration
  approach
    contingencies 136
    soft-wiring 137
  real-time distributed control systems of 134
    IEC 61499-based model 127 et seq

remote function block (FB) manager
  interface 140
requirement analysis, modelling
  elements and 49 et seq
  product generic modelling,
    configuration for 49 et seq
resource centres 116

scheduling
  agent-based
    agile system 87 et seq
    algorithm 93, 105
      negotiation 97
    framework 91
    literature 90
    negotiation model 93
  problems
    classification of 39
    DeSAP interface
      e-OCEA environment
        27
    identification of 37
security manager 30
simple assembly 88
software agents
  based modelling 121
  mobile 117
soft-wiring approach, reconfiguration
  to 137
specification process 2
structure layer 10
subject layer 9
supply chain management, process
  and 148
systemion 119

VFDCS 154
virtual reality modelling language
  (VRML) 80