

**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DO PARANÁ (CEFET-PR)**  
**Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial (CPGED)**

**UNIVERSITÉ HENRI POINCARÉ (UHP)**  
**Centre de Recherche en Automatique de Nancy (CRAN)**

---

**A CONTRIBUTION TO THE DEVELOPMENT OF A  
HMS SIMULATION TOOL AND PROPOSITION OF A  
META-MODEL FOR HOLONIC CONTROL**

---

**Doctoral Thesis**

**by**

**Jean Marcelo SIMÃO**

Examining Board:

Advisors:

Prof. Paulo César STADZISZ

*Centro Federal de Educação Tecnológica do Paraná (CEFET-PR)*

Prof. Gérard MOREL

*Université Henri Poincaré (UHP)*

President:

Prof. Luis Allan KÜNZLE

*Universidade Federal do Paraná (UFPR)*

Reporters:

Prof. Bernard GRABOT

*École Nationale d'Ingénieurs de Tarbes (ENIT)*

Prof. José Eduardo Ribeiro CURY

*Universidade Federal de Santa Catarina (UFSC)*

Examiners:

Prof. Carlos Eduardo Trabuco DÓREA

*Universidade Federal da Bahia (UFBA)*

Prof. César Augusto TACLA

*Centro Federal de Educação Tecnológica do Paraná*

Curitiba, Paraná, Brazil

June 13, 2005



**J E A N M A R C E L O S I M Ã O**

---

A CONTRIBUTION TO THE DEVELOPMENT OF A  
HMS SIMULATION TOOL AND PROPOSITION OF A  
META-MODEL FOR HOLONIC CONTROL

---

Doctoral Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Science (DSc) at the Graduate School in Electrical Engineering and Industrial Computer Science (CPGEI), The Federal Center of Education in Technology of Paraná (CEFET-PR), emphasis on Industrial Computer Science.

Advisor: Prof. Dr. Paulo César Stadzisz.

Doctoral Thesis equally submitted in partial fulfillment of the requirements for the degree of Doctor (Dr.) of Henri Poincaré University (UHP) at UHP, Research Center for Automatic Control of Nancy (CRAN), emphasis on Automatics, Signal Treatment, and Computer Engineering.

Advisor: Prof. Dr. Gérard Morel

Curitiba, Paraná, Brazil.

June 13, 2005

Ficha catalográfica elaborada pela Biblioteca do CEFET-PR – Unidade Curitiba

S588c Simão, Jean Marcelo

A Contribution to the development of a HMS simulation tool and proposition of a meta-model for holonic control / Jean Marcelo Simão – Curitiba : CEFET-PR, 2005.

222 f. : il. ; 30 cm





Orientador: Prof. Dr. Paulo Cezar Stadzisz

Tese (Doutorado) – CEFET-PR. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. Curitiba, 2005

Bibliografia: f. 157-166

1. Análise de sistemas. 2. Sistemas de manufatura holônicos. 3. Meta-modelo Para controle holônico. 4. Controle orientado a regras e agentes. 5. Controle Orientado ao produto. 6. Ferramenta de simulação. I. Stadzisz, Paulo Cezar. II. Centro Federal de Educação Tecnológica do Paraná. Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. III. Título.

CDD: 620.7

  <p>CNRS UMR 7039</p>	 
<p>Faculté des Sciences UFR Sciences Techniques Mathématiques Informatique Automatique</p> <p>Ecole Doctorale IAEM Lorraine DFD Automatique</p> <p><b>Université Henri Poincaré (UHP) – Nancy I France</b></p>	<p>Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial (CPGEI)</p> <p><b>Centro Federal de Educação Tecnológica do Paraná (CEFET-PR) – Brasil</b></p>

**THÈSE en co-tutelle entre l'UHP et le CEFET-PR**  
présentée en vue de l'obtention des titres de

**Docteur de l'Université Henri Poincaré, Nancy I**  
en Automatique, Traitement du Signal, Génie Informatique

et

**Doutor em Ciências pelo Centro Federal de Educação Tecnológica do Paraná**  
na área de Engenharia Elétrica e Informática Industrial

par

**Jean Marcelo SIMÃO**

---

**A Contribution to the Development of a HMS Simulation Tool and  
Proposition of a Meta-Model for Holonic Control**

---

**Contribution au Développement d'un Outil de Simulation de Systèmes Holoniques  
de Production et Proposition d'un Meta-Modèle de Contrôle Holonique**

---

**Uma Contribuição ao Desenvolvimento de uma Ferramenta de Simulação de Sistemas de  
Manufatura Holônicos e Proposição de um Meta-Modelo para o Controle Holônico.**

---

Thèse soutenue le 13 juin 2005 à Curitiba (Paraná) Brésil – CEFET-PR

Membres du Jury:

Président et Rapporteur :	Prof. Bernard GRABOT	École Nationale d'Ingénieurs de Tarbes
Rapporteur :	Prof. José Eduardo Ribeiro CURY	Universidade Federal de Santa Catarina
Directeur de Thèse :	Prof. Gérard MOREL	Université Henri Poincaré – Nancy I
Directeur de Thèse :	Prof. Paulo César STADZISZ	Centro Fed. de Educ. Tecnol. do Paraná
Examineur Invité :	Prof. Carlos Eduardo Trabuco DÓREA	Universidade Federal da Bahia
Examineur Invité :	Prof. Luis Allan KÜNZLE	Universidade Federal do Paraná
Examineur Invité :	Prof. César Augusto TACLA	Centro Fed. de Educ. Tecnol. do Paraná

---

Centre de Recherche en Automatique de Nancy (CRAN) – UHP  
Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial (CPGEI) – CEFET-PR



# Acknowledgements

I would like to express my sincere and deep gratitude to Mr. Stadzisz for intensely guiding and advising me during the development of my research efforts from the master thesis to the current doctoral thesis. I would also like to thank Mr. Stadzisz for his more general advice, as well as for his sincere friendship.

In the same way, I would like to demonstrate my sincere and deep gratitude to Mr. Morel, for having invited me to do a part of my doctoral project in France, where he has given me a lot of advice, orientation, and support. My gratitude to Mr. Morel extends to his friendship and his teaching about the fascinating French culture.

Other acknowledgements go in particular to Mr. Grabot and to Mr. Cury. I am profoundly grateful for their acceptance and for the attention they have devoted to correcting this doctoral thesis. I am equally thankful to Mr. Dórea, Mr. Künzle, and Mr. Tacla for their acceptance and for their attention in correcting this doctoral thesis.

Finally, I would like give a special thanks to Mr. Künzle for his advice and friendship during these past six years and to Mr. Tacla for his advice and friendship this past year. Similarly, I would like to mention my laboratory friends, both in France and Brazil, for seemingly unlimited professional help and friendship. More specifically, I express my friendship to L.F.F. Rosinha (*in memoriam*) and A. Koscianski, who have developed works on which the current project is based.

In general, I would like to express all my indebtedness to the people that have personally or professionally helped me along my way. It would be, perhaps, impractical to mention each one here; however each one indubitably has been very present in my thoughts and prayers. Specifically, I would like to thank my family, my very close friends, and S. L. for all the sentimental and material help that they have provided me.

I would also like to express my appreciation to the Brazilian and French governments for the material and financial support they have provided me. In particular, I would like to thank CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*) for the national and international scholarship covering the period between May of 2001 and April of 2005. In addition, I am very thankful to CPGEI/CEFET-PR, CRAN-UHP, and other concerned institutions, for all the support they have provided me.

Finally, I express my infinite thankfulness to the Supreme Being.

Sincerely,

Jean Marcelo Simão.

# Remerciements

Je voudrais exprimer ma sincère et profonde gratitude à M. Stadzisz pour intensivement diriger et conseiller le développement de mes travaux de recherche depuis du master jusqu'au présent travail de doctorat. Je voudrais encore remercier M. Stadzisz pour les conseils dans un sens large et aussi pour l'amitié sincère.

Également, je tiens à démontrer mon sincère et profonde gratitude au M. Morel. M. Morel m'a invité à son laboratoire pour réaliser une partie du doctorat en France en m'apportant beaucoup de conseils, de support et de l'orientation. Ma gratitude à M. Morel concerne tout son amitié et ses enseignements sur la fascinante culture française.

Mes remerciements sont aussi particulièrement adressés à M. Grabot et à M. Cury à qui je remercie pour avoir accepté de reporter ce travail de thèse. Pareillement, mes remerciements sont présentés à M. Dórea, M. Künzle et M. Tacla pour ses acceptations et pour ses attentions pour examiner ce travail de thèse.

En outre, je voudrais faire une mention spéciale à M. Künzle pour ses conseils et pour son amitié pendant les six dernières années et à M. Tacla pour ses conseils et pour son amitié dans la dernière année. Similairement, je voudrais mentionner les amis des laboratoires, en France et au Brésil, pour leurs innombrables aides professionnelles et leur amitié. Spécialement, j'exprime mon amitié à L. F. F. Rosinha (*in memoriam*) et à A. Koscianski qui ont développé des travaux de base en permettant le présent travail de recherche.

En général, je voudrais exprimer toute ma reconnaissance aux personnes qui m'ont aidé, personnellement ou professionnellement, dans mon chemin. Il serait, peut-être, impraticable de les mentionner tous, cependant chacun a certainement beaucoup d'attention dans mes pensées et dans mes prières. Spécifiquement, je voudrais remercier ma famille, mes amis et S.L. de toute leur aide sentimentale et matérielle.

L'appréciation relative au support matériel et financier est aussi présentée aux gouvernements brésilien et français. Particulièrement, je voudrais remercier la CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*) pour les bourses des études (national et international) entre mars 2001 et avril 2005. Je remercie encore et beaucoup, le CPGEI/CEFET-PR, le CRAN-UHP et les institutions concernés pour tout le support.

Finalement, j'exprime toute mon infinité gratitude à l'Être Suprême

Sincèrement.

Jean Marcelo Simão



# Agradecimentos

Eu gostaria de expressar minha sincera e profunda gratidão ao Sr. Stadzisz por intensivamente orientar e aconselhar o desenvolvimento de meus trabalhos de pesquisa a partir do trabalho de mestrado até o presente trabalho de doutorado. Ainda, eu gostaria de agradecer o Sr. Stadzisz pelos conselhos num amplo sentido bem como pela amizade sincera.

Igualmente, eu gostaria de demonstrar minha também sincera e profunda gratidão ao Sr. Morel que me convidou para fazer parte do trabalho de doutorado na França, onde ele me concedeu vários conselhos, orientações e suporte. Minha gratidão ao Sr. Morel também inclui toda sua amizade e o seu ensinar sobre a fascinante cultura francesa.

Outros agradecimentos são particularmente feitos ao Sr. Grabot e ao Sr. Cury, eu estou profundamente agradecido pelas suas aceitações e atenções para relatar este trabalho de tese. De uma mesma maneira, meus agradecimentos são apresentados ao Sr. Dórea, ao Sr. Künzle e ao Sr. Tacla pelas suas aceitações e atenções para examinar este trabalho de tese.

Ainda, eu gostaria de fazer uma menção especial de agradecimento ao Sr. Künzle pelos seus conselhos e amizade durante estes últimos seis anos e ao Sr. Tacla pelos seus conselhos e amizade neste último ano. Similarmente, eu gostaria de mencionar os amigos dos laboratórios, na França e no Brasil, pelas incontáveis ajudas profissionais e amizade. Especialmente, eu expresso minha amizade por L.F.F. Rosinha (*in memoriam*) e por A. Koscianski que desenvolveram trabalhos de base, permitindo o presente trabalho de pesquisa.

Em geral, eu gostaria de expressar todo meu reconhecimento para com as pessoas que têm, de forma pessoal ou profissional, ajudado no meu caminho. Seria, talvez, impraticável mencionar cada um aqui, entretanto cada um tem indubitavelmente muita atenção nos meus pensamentos e nas minhas orações. Especificamente, eu gostaria de agradecer por toda a ajuda sentimental e material que minha família, meus amigos próximos e S.L. têm me proporcionado.

A apreciação relativa ao suporte material e financeiro é também apresentado aos governos brasileiro e francês. Particularmente, eu gostaria de agradecer à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelas bolsas de estudo (nacional e internacional) entre março de 2001 e abril de 2005. Eu ainda agradeço, em muito, o CPGEI/CEFET-PR, o CRAN-UHP e as instituições correlatas por todo o suporte.

De uma maneira final, eu expresso minha infinita gratidão ao Ser Supremo.

Sinceramente

Jean Marcelo Simão



# Abstract

The present context and tendencies in modern production system, as mass customization, requires improvements with respect to the agility of the production organizations. In this sense, agile approaches have been proposed, such as the holonic approach. In Holonic Manufacturing System (HMS) the production entities, as resources and products, are envisaged with a type of intelligence. These smart-entities are called holons (HLs) whose intelligence is related to their autonomy and collaboration skills.

The HMS also comprises a Holonic Control (HC) that must properly organize holon collaborations in order to become agile. Actually, HMS development requires engineering tools for design and testing. In this doctoral thesis, a meta-model for HC is proposed, whose instances are simulated within a particular tool called ANALYTICE II. This tool presents a clear separation between high-level control and emulated resources.

Firstly, before the proposition of the HC meta-model, the resource *holonification* is proposed in this environment. Each Resource-HL is obtained by means of a virtual resource that provides data and services of an emulated-resource at a high level of control. Subsequently, the meta-model for HC over Resource-HLs following a process-driven production approach is proposed.

The essence of the solution is based on Rule Base System (RBS) concepts being the causal relations of control dealt with by entities called Rules. The inference process in this RBS is realized through collaborations based upon notifications. The Resource-HLs notify the Rules about factual knowledge with respect to their states. Each Rule that is notified deliberates about the proper moment to execute some control action, as the coordination of a set of Resource-HLs, using causal knowledge.

The inference occurs within a notification chain enabled by a group of Resource-HL agents and Rule agents. This kind of inference can be expected to provide advantages for the HC, such as high reactivity and entity decoupling. Furthermore, it allows for the creation of co-operative mechanisms for dealing with determinism and conflict issues. Moreover, this approach of rule-oriented control allows for coherent control implementation and expression.

The control mechanisms emerge based on causal control knowledge expressed by experts in the Rules. Experts are exclusively concerned with the proper control knowledge needed for exploiting system flexibilities in order to increase system agility. Furthermore, some experts could even be artificial agents automatically dealing with knowledge of the Rules. Briefly, this process-driven HC solution concomitantly treats a set of control issues while also being a self-contained and open solution.

Indeed, the solution openness allows its interpretation as a product-driven solution. The product-driven control is a tendency to reach agility by the decoupling of production demands and execution via entities like Smart-Product-HLs. Each Smart-Product-HL is concerned with a specific customized production order. The Smart-Product-HLs, with certain autonomy, use Resource-HLs to reach their production goals.

In the meta-model interpretation, their interactions are organized by Rules for Resource-HL cooperation that avoids inappropriate system behavior. In this context, the execution of Rules depends upon the explicit Smart-Product-HL interest in their utilization. In some manner, each Smart-Product-HL deals with Rules as a kind of expert agent. The solution has been applied in a set of examples in ANALYTICE II presenting some simulation independence because each control instance is not aware that Resource-HLs and Smart-Product-HLs are simulated.

# Résumé

Le présent contexte et les tendances autour des systèmes de production modernes, comme la personnalisation de masse, conduisent à des besoins d'améliorations en ce qui concerne l'agilité des organisations de production. Ainsi, des approches agiles ont été proposées telle que l'approche holonique. Dans des Systèmes Manufacturiers Holoniques (*HMS*) les entités de production, par exemple les ressources et les produits, sont envisagées avec un certain degré d'expertise. Ces entités expertes sont appelées *holons* (*HLs*) et leur expertise concerne les habiletés d'autonomie et de collaboration.

L'*HMS* contient aussi le Contrôle Holonique (*HC*) qui doit organiser proprement les collaborations des *holons* pour atteindre de l'agilité. En effet, le développement des *HMS* demande des outils d'ingénierie d'aide au projet et aux tests. Dans cette thèse, il est proposé un meta-modèle pour *HC* dont les systèmes dérivés sont simulés dans un outil appelé ANALYTICE II. Cet outil présente une séparation précise entre les entités de contrôle du haut niveau et les ressources émulées.

Premièrement, avant de proposer le meta-modèle pour l'*HC*, l'*holonification* de ressource est proposée dans cet environnement. Chaque *Resource-HL* est obtenue à l'aide d'une ressource virtuelle qui permet d'accéder des données et des services d'une ressource émulée au haut niveau de contrôle. Par la suite, il est proposé le meta-modèle pour l'*HC*, sur les *Resource-HLs*, dans une orientation au processus.

L'essence de la solution est inspirée des concepts des Systèmes à Base de Règles (*RBS*) où les relations causales du contrôle sont traitées par des entités appelées Rules. Le processus d'inférence dans ce genre de *RBS* a été obtenu grâce à des collaborations basées sur notifications. Les *Resource-HLs* notifient les *Rules* par milieu de la connaissance factuelle, comme leurs états. Chaque *Rule* notifié délibère au moment approprié sur l'exécution d'une certaine action de contrôle.

L'inférence se passe dans une chaîne de notifications grâce à une composition de *Resource-HLs* et de *Rules* basées sur agents. Ce type d'inférence apporte des avantages pour l'*HC* tels que la haute réactivité et le découplage des éléments. Il permet aussi la création de mécanismes coopératifs pour répondre aux besoins du contrôle comme le déterminisme et la résolution de conflits. De plus, cette approche de contrôle orientée aux règles permet d'obtenir une implémentation et une expression cohérentes du contrôle.

Les mécanismes de contrôle sont émergés à partir de la connaissance causale de contrôle exprimée par des experts dans les *Rules*. Des experts sont exclusivement impliqués dans la connaissance de contrôle appropriée pour exploiter les flexibilités du système en cherchant de l'agilité. En outre, certains experts pourraient être des agents artificiels pour traiter de façon automatique la connaissance des *Rules*. En résumé, cette solution de *HC* orientée au processus traite simultanément un ensemble de sujets de contrôle encore en s'agitant d'une solution indépendante et aussi ouverte.

En fait, l'ouverture de la solution permet son interprétation comme une solution orientée au produit. Le contrôle orienté au produit est une tendance pour trouver de l'agilité via le découplage des demandes de production et ses exécutions en utilisant des entités comme les *Smart-Product-HLs*. Chaque *Smart-Product-HL* concerne un ordre de production spécifique et personnalisée. Les *Smart-Product-HLs*, avec une certaine autonomie, utilisent les *Resource-HLs* pour répondre à ses besoins de production.

Dans l'interprétation du meta-modèle, leurs interactions sont organisées en utilisant les *Rules* pour la coopération des *Resource-HLs* qu'empêchent des comportements impropres du système. Dans ce contexte, l'exécution des *Rules* dépend de l'intérêt explicite des *Smart-Product-HLs* dans leurs utilisations. En quelque sorte, chaque *Smart-Product-HL* utilise des *Rules* comme un genre d'expert. La solution a été appliquée dans un ensemble d'exemples en ANALYTICE II qui ont présenté une certaine indépendance de la simulation, celle-ci parce que chaque système de contrôle n'est pas conscient que les *Resource-HLs* et les *Smart-Product-HLs* sont simulés.

# Resumo

O atual contexto e tendências relacionadas com os sistemas modernos de produção, como a personalização de massa, reclamam aperfeiçoamentos relativos à agilidade nas organizações de produção. Neste sentido, abordagens ágeis têm sido propostas como, por exemplo, a abordagem holônica. Nos Sistemas de Manufatura Holônicos (*HMS*) as entidades de produção, tais quais os recursos e os produtos, são vislumbrados com uma certa inteligência. Estas entidades “inteligentes” são chamadas de *holons* (*HLs*) cuja “inteligência” é relacionada às habilidades de autonomia e colaboração.

O *HMS* também contém o Controle Holônico (*HC*) que deve organizar apropriadamente as colaborações dos *holons* para alcançar agilidade. De fato, o desenvolvimento de *HMS* necessita de ferramentas de engenharia para projeto e teste. Nesta tese é proposto um meta-modelo para *HC* cujos sistemas derivados são simulados em uma ferramenta chamada ANALYTICE II. Esta ferramenta apresenta uma nítida separação entre as entidades do controle de alto nível e os recursos emulados da planta.

Primeiramente, antes da proposição do meta-modelo para *HC*, a *holonificação* de recurso é proposta neste ambiente. Cada *Resource-HL* é obtido por meio de um recurso virtual que permite o acesso a dados e serviços do recurso emulado no alto nível de controle. Subseqüentemente, é proposto o meta-modelo para *HC*, sobre os *Resource-HLs*, orientado ao processo. A essência da solução é inspirada no conceito de Sistemas Baseados em Regras (*RBS*) onde as relações causais do controle são tratadas por entidades chamadas *Rules*. O processo de inferência neste tipo de *RBS* é realizado via colaborações baseadas em notificações. Os *Resource-HLs* notificam conhecimento factual, como seus estados, para as *Rules*. Cada *Rule* notificada delibera a respeito do momento apropriado para a execução de uma certa ação de controle.

A inferência acontece em uma cadeia de notificações possível devido a uma composição dos *Resource-HLs* e das *Rules* baseada em agentes. Este tipo de inferência traz vantagens para o *HC* como a alta reatividade e o desacoplamento dos elementos. Ela também permite a criação de mecanismos cooperativos para questões relativas ao determinismo e ao conflito. Além disso, esta abordagem de controle orientada a regras permite a coerência entre a implementação e a expressão do controle.

Os mecanismos de controle emergem a partir do conhecimento causal de controle expressado por especialistas nas *Rules*. Os especialistas são exclusivamente concentrados no conhecimento apropriado de controle para explorar as flexibilidades do sistema objetivando agilidade. Além do mais, certos especialistas poderiam ser agentes artificiais para tratar, de maneira automática, os conhecimentos das *Rules*. Em suma, esta solução de *HC* orientada ao processo trata simultaneamente de um conjunto de questões de controle ainda sendo uma solução auto-contida e também aberta.

De fato, a abertura da solução permite sua interpretação como uma solução orientada ao produto. O controle orientado ao produto é uma tendência para alcançar agilidade via o desacoplamento dos pedidos de produção e suas execuções utilizando entidades como, por exemplo, os *Smart-Product-HLs*. Cada *Smart-Product-HLs* é relacionado a uma ordem de produção específica e personalizada. Os *Smart-Product-HLs*, com uma certa autonomia, utilizam os *Resource-HLs* para alcançar seus desejos de produção.

Na interpretação do meta-modelo, suas interações são organizadas utilizando *Rules* de cooperação de *Resource-HLs* que impedem comportamentos impróprios do sistema. Neste contexto, a execução das *Rules* depende do interesse explícito dos *Smart-Product-HLs* nas suas utilizações. De uma certa forma, cada *Smart-Product-HL* utiliza as *Rules* agindo como um tipo de especialista. A solução tem sido aplicada num conjunto de exemplos em ANALYTICE II que têm apresentado certa independência da simulação, uma vez que cada sistema de controle não é consciente que os *Resource-HLs* e os *Smart-Product-HLs* são simulados.



---

# Table of Contents

<b>CHAPTER 1 : INTRODUCTION.....</b>	<b>3</b>
1.1. INTRODUCTION.....	3
1.2. INDUSTRIAL CONTEXT .....	3
1.3. AGILE AND HOLONIC MANUFACTURING .....	5
1.4. HOLONIC CONTROL.....	8
1.5. SIMULATION.....	9
1.6. THESIS' OBJECTIVES .....	10
1.7. THESIS' STRUCTURE .....	11
<b>CHAPTER 2 : CONTEXT &amp; OBJECTIVES .....</b>	<b>15</b>
2.1. INTRODUCTION.....	15
2.2. AGILE MANUFACTURING .....	16
2.3. HOLONIC MANUFACTURING.....	17
2.4. LARGE SYSTEM CONTROL.....	20
2.5. CONTROL FORMALIZATION .....	22
2.6. HOLONIC THINKING .....	22
2.7. HOLONIC CONTROL.....	26
2.8. CONTROL COMPLEXITY.....	28
2.9. SIMULATION.....	29
2.10. ANALYTICE II.....	31
2.11. HOLONIC SIMULATOR .....	33
2.12. CONTROL SOLUTION .....	34
<b>CHAPTER 3 : HOLONS IN ANALYTICE II.....</b>	<b>39</b>
3.1. INTRODUCTION.....	39
3.2. SIMULATION ISSUES .....	39
3.3. SIMULATION ENVIRONMENT .....	40
3.4. SIMULATION ARCHITECTURE .....	42
3.5. FUNCTIONAL PROTOTYPE.....	45
3.6. A PROCESS-CELL IN ANALYTICE II.....	46
3.7. COMPUTATIONAL HOLONIFICATION.....	47
3.8. RESOURCE HOLONIFICATION .....	48
3.9. RESOURCE HOLON IN ANALYTICE II.....	49
3.10. A SET OF RESOURCE-HLS .....	50
3.11. SELF-SIMILAR RESOURCE-HLS .....	51
3.12. SELF-SIMILARITY .....	52
3.13. ATTRIBUTE SUBHOLON .....	53
3.14. THE METHOD SUBHOLON.....	54
3.15. RESOURCE-HL FORMALIZATION.....	55
3.16. HOLONIC CONTROL.....	56
3.17. REMARKS .....	57
<b>CHAPTER 4 : RULE AND AGENT-ORIENTED CONTROL.....</b>	<b>61</b>
4.1. INTRODUCTION.....	61
4.2. RULE-ORIENTED CONTROL.....	63
4.3. CONTROL SOLUTION ISSUES.....	64
4.4. RULES AS HOLONS .....	65
4.5. RULE COLLABORATORS .....	66
4.6. RULE STRUCTURE .....	67
4.7. SUMMARY OF INFERENCE PROCESS.....	68
4.8. NOTIFICATION MECHANISM.....	69
4.9. CONTROL INSTANCE.....	70
4.9.1. Introduction .....	70

4.9.2.	<i>Coordination Rules</i> .....	70
4.9.3.	<i>Rules in Conflicts</i> .....	71
4.9.4.	<i>Rule Features</i> .....	72
4.9.5.	<i>Decisional Rules</i> .....	73
4.10.	RULE COMPOSITION .....	73
4.11.	RULES AND AGILITY .....	75
4.12.	CONSIDERATIONS .....	78
<b>CHAPTER 5 : HOLONIC CONTROL .....</b>		<b>83</b>
5.1.	INTRODUCTION .....	83
5.2.	RULE BASED SYSTEMS .....	83
5.2.1.	<i>Introduction</i> .....	83
5.2.2.	<i>Inference Engine</i> .....	84
5.2.3.	<i>Avoiding Searches</i> .....	85
5.2.4.	<i>Notification Principle</i> .....	87
5.3.	COMPUTATIONAL COMPLEXITY .....	89
5.3.1.	<i>Introduction</i> .....	89
5.3.2.	<i>Asymptotic Analysis of the Complexity</i> .....	90
5.3.3.	<i>Complexity and Time</i> .....	91
5.4.	CONTROL CORRECTNESS .....	92
5.4.1.	<i>Introduction</i> .....	92
5.4.2.	<i>Conflict Issues</i> .....	92
5.4.3.	<i>Reactivity and Determinism</i> .....	94
5.4.4.	<i>Robustness Features</i> .....	96
5.5.	FORMALISM FOR CONTROL EXPRESSION .....	97
5.5.1.	<i>Introduction</i> .....	97
5.5.2.	<i>Rules and Petri Nets</i> .....	98
5.5.3.	<i>Petri Net Player</i> .....	99
5.5.4.	<i>Petri Net Playing</i> .....	99
5.5.5.	<i>Petri Nets and Solution Generality</i> .....	101
5.5.6.	<i>The Agility and Petri Nets</i> .....	102
5.5.7.	<i>Player Architecture</i> .....	103
5.6.	SUMMARY .....	105
<b>CHAPTER 6 : PRODUCT DRIVEN CONTROL .....</b>		<b>109</b>
6.1.	INTRODUCTION .....	109
6.2.	SMART-PRODUCT-HOLON .....	110
6.3.	RULES AND SMART-PRODUCT-HOLONS .....	111
6.4.	CONTROL INSTANCE .....	113
6.5.	SMART-PRODUCT-HOLON ISSUES .....	115
6.6.	HOLONS INTERACTION .....	116
6.7.	HOLONIC CONTROL ARCHITECTURE .....	119
6.8.	RULES CREATION .....	120
6.9.	SUMMARY .....	120
<b>CHAPTER 7 : CASE STUDIES .....</b>		<b>125</b>
7.1.	INTRODUCTION .....	125
7.2.	MACHINING SYSTEM .....	126
7.2.1.	<i>Introduction</i> .....	126
7.2.2.	<i>Process-driven Control Simulation</i> .....	127
7.2.3.	<i>Product-driven Control Modeling</i> .....	128
7.2.4.	<i>Summary</i> .....	129
7.3.	FLEXIBLE ASSEMBLING CELL .....	130
7.3.1.	<i>Introduction</i> .....	130
7.3.2.	<i>Holonic FAC</i> .....	131
7.3.3.	<i>Resource Holons Development</i> .....	132



---

7.3.4.	<i>Process Plan</i>	133
7.3.5.	<i>Rules</i>	135
7.3.6.	<i>Smart-Product Holons</i>	136
7.3.7.	<i>Production Flexibility</i>	136
7.3.8.	<i>Selection/Allocation Process</i>	137
7.3.9.	<i>Information for Agility</i>	137
7.3.10.	<i>Summary</i>	138
7.4.	AIPL CASE STUDY	139
7.4.1.	<i>Introduction</i>	139
7.4.2.	<i>Simulation Context</i>	140
7.4.3.	<i>Simulation Issues</i>	141
7.4.4.	<i>Holonic Control</i>	142
7.4.5.	<i>Summary</i>	145
7.5.	CONSIDERATIONS	145
<b>CHAPTER 8 : CONCLUSION AND PERSPECTIVES</b>		<b>149</b>
8.1.	CONCLUSION	149
8.2.	PERSPECTIVES	153
<b>BIBLIOGRAPHY</b>		<b>157</b>
<b>ABBREVIATIONS</b>		<b>167</b>

---

---

## Table of Figures

Figure 1: Illustration of the Technological Evolution. ....	3
Figure 2: Automobile Industry Evolution.....	4
Figure 3: Enterprise Integration.....	5
Figure 4: Smart-Entities in a Flexible Plant. ....	7
Figure 5: Flows consistence. ....	8
Figure 6: Holonic Discrete Control System (HDCS) functions.....	9
Figure 7: ANALYTICE II – An in-house simulation tool.....	10
Figure 8: Object and Classes in UML for model holons. ....	18
Figure 9: Holonic Robotic Assembling Cell (McFarlane et al. 2003). ....	20
Figure 10: UML meta-class diagram. ....	24
Figure 11: Main holonic-classes in PROSA.....	25
Figure 12: Resource-HL specialization. ....	25
Figure 13: Holonic Discrete Control System.....	27
Figure 14: ANALYTICE II structure and its graphics animator module. ....	32
Figure 15: Attributes and Methods for Resource-HLs. ....	34
Figure 16: HMESE framework (Morel et al., 2003).....	36
Figure 17: Conceptual Block Diagram of the ANALYTICE II Simulation Environment. ....	40
Figure 18: Simulator primitives: equipment, pallets, and parts. ....	41
Figure 19: 3D graphical animation of an AGV in ANALYTICE II.....	42
Figure 20: Petri Net of the Simulator Kernel.....	43
Figure 21: 3D graphical interface and user interface in ANALYTICE II. ....	46
Figure 22: Virtual Machining Cell simulated using ANALYTICE II.....	46
Figure 23: Process plans of virtual products.....	47
Figure 24: Virtual resource (agents) to represent and treat each piece of equipment. ....	48
Figure 25: Example of Resource-HL in ANALYTICE II. ....	49
Figure 26: Specialization/generalization tree about Resource-HL. ....	50
Figure 27: Attributes and Methods Holonic-Classes.....	52
Figure 28: Attribute-SHL and Command-SHL. ....	56
Figure 29: Control Entity.....	61
Figure 30: Control Rule expressing a causal relation. ....	63
Figure 31: Rules as soft-holons. ....	65
Figure 32: Control Entity.....	66
Figure 33: Example of Rule knowledge. ....	67
Figure 34: The structure of Rules and their collaborators. ....	67
Figure 35: Notification mechanism. ....	69
Figure 36: Virtual Machining Cell in ANALYTICE II. ....	70
Figure 37: Rules to transport parts from Store for Table.1.....	71
Figure 38: Rules to coordinate Resource-HLs.....	72
Figure 39: Rules for signalize the reposition and removing of parts. ....	73
Figure 40: Rules improved to support a product variant. ....	76
Figure 41: Rules to deal with more a Lathe-HL in the Environment. ....	77
Figure 42: The Cycle Operation of Inference Engine in a Block Diagram. ....	84
Figure 43: Examples of rules with simple and compound premises.....	85

---

Figure 44: Distributed Rules on Control Islands. ....	86
Figure 45: Generic examples of Attributes inside of a FBA. ....	87
Figure 46: Sequence diagram to exemplify a notification chain. ....	88
Figure 47: Main classes in the control architecture. ....	89
Figure 48: Class Diagram for Rule Conflict Identification and Solving. ....	93
Figure 49: An instance of the dynamics during conflict identification and resolution. ....	94
Figure 50: Representation of the Mechanism for Deterministic-Evolution. ....	95
Figure 51: Instance of control entities considering the deterministic-evolution. ....	96
Figure 52: A Rule in Petri net representation. ....	98
Figure 53: A Control designed using a Petri net. ....	100
Figure 54: Complementary subnet. ....	101
Figure 55: An alternative to the subnet in Figure 54. ....	102
Figure 56: Control Improved. ....	103
Figure 57: Feedback control evolution. ....	110
Figure 58: Dynamics concerned to Smart-Product-HLs, Rules and Resource-HLs. ....	111
Figure 59: Rule and Smart-Product Holonic-Classes. ....	112
Figure 60: Rules to transport part from Store3x3.1 to Table2P.1. ....	113
Figure 61: Rules to the co-ordination of Resource-HLs. ....	114
Figure 62: Smart-Product-HL allocating Rule E'. ....	117
Figure 63: Smart-Product-HL allocating Rule B'. ....	118
Figure 64: Holarchy with Smart-Product-HLs, Rules, and Resource-HLs. ....	119
Figure 65: Product produced at AIPL-PRIMECA area. ....	125
Figure 66: A similar AIPL-PRIMECA context. ....	126
Figure 67: Possible production ways. ....	126
Figure 68: Process Plan for the Part from type 01 or 09. ....	127
Figure 69: Similar Rules for a control instance. ....	128
Figure 70: Alternative Rule. ....	129
Figure 71: Picture of Flexible Assembly Cell (FAC). ....	130
Figure 72: HFAC Resource-HLs. ....	131
Figure 73: Segments and positions in the conveyor. ....	132
Figure 74: Distribution Scenario of Passive-Smart-Product-HLs. ....	133
Figure 75: Process Plan to Products of the types A and D. ....	134
Figure 76: The first part of the set of Rules for the HDCS-HFAC. ....	135
Figure 77: The second part of the set of Rules for HDCS-HFAC. ....	135
Figure 78: Flow issues. ....	138
Figure 79: The sketch of the simulated plant in ANALYTICE II. ....	140
Figure 80: Rules for determining the assemblages in the FAC. ....	142
Figure 81: Rules for the starting of the Smart-Product-HLs concerned to base-parts. ....	143
Figure 82: Rules for enabling the rough material and production of base-parts. ....	143
Figure 83: Rules for reaching the finalization of base-parts. ....	144
Figure 84: Rules for enabling base-part in the WSs of FAC. ....	144

---

## Table of Equation

Equation 1: Fusaoka's Equation.....	22
Equation 2: Holonification of Emulated Entities.....	33
Equation 3: Correct and Holonic Control Systems.....	35
Equation 4: Composition of Resource-HL.....	55
Equation 5: Complexity in the Attribute notification.....	91
Equation 6: Complexity in the Attribute notification taking into account the time.....	91
Equation 7: Rule Creation.....	120

---

## List of Tables

Table 1: Classified approaches .....	6
Table 2: Possible requirements related to controller correctness.....	29
Table 3: Properties and particular features of ANALYTICE II .....	31
Table 4: An interpretation of correctness and holonics features.....	35

## **Introduction**

---





# Chapter 1 : Introduction

## 1.1. Introduction

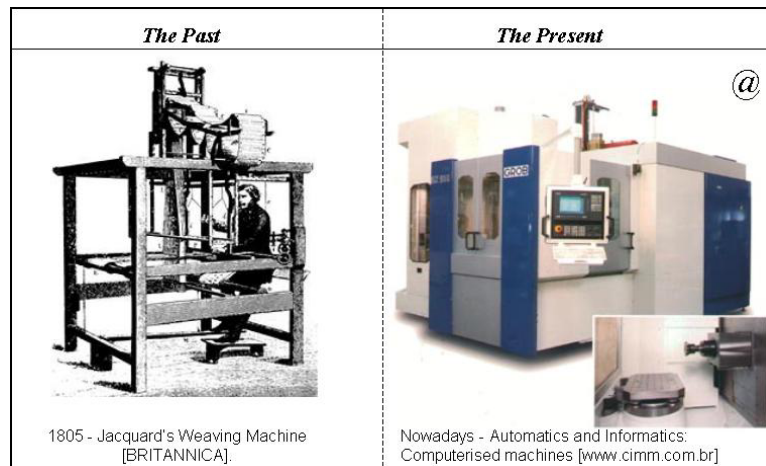
This introductory chapter presents the context, motivations, and objectives of this doctoral thesis. Initially, the industrial context is highlighted summarizing its evolution, tendencies, and challenges. After that, a research approach called agile manufacturing is presented, emphasizing holonic manufacturing as a potential solution to certain challenges in the industrial context.

In the scope of holonic manufacturing, the control system is described due to its importance for attaining the objectives of agile manufacturing. Subsequently, the simulation is highlighted as a means to test new technologies, e.g. holonic manufacturing systems. Also highlighted is a special simulator, called ANALYTICE II, and its role in a research and educational project, since its features may be relevant to research into agile manufacturing.

At the end of this chapter, the objectives of the doctoral thesis are presented. The objectives are concerned with an appropriate simulator and control solution to holonic manufacturing systems. As a final matter, the structure of the doctoral thesis is also outlined.

## 1.2. Industrial Context

There have been some remarkable developments within society within the past century. These include the development of the relationship between production and business. Certainly, this development also results from the evolution of production systems and computational systems, brought about by the introduction of new methods and technologies (Wyns, 1999), as illustrated in Figure 1.



**Figure 1: Illustration of the Technological Evolution.**

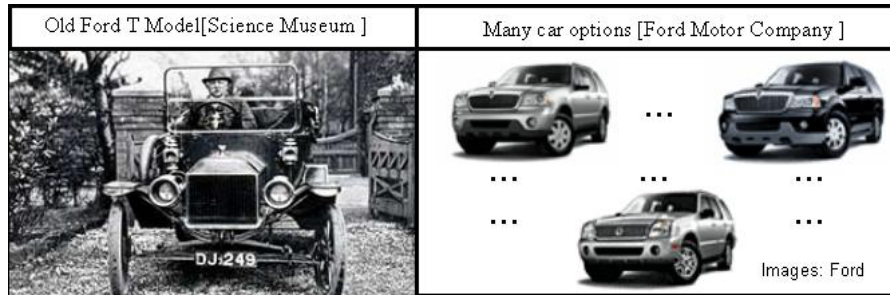
Nowadays, in an inverse way, the current state of society forces the development of more sophisticated production and informational systems, resulting in a dynamic improvement process. Increasing competition among businesses and increasing consumer demands, for example, explain the need for improvements.

The demand for quality, diversity, and time-to-market by the majority of consumer markets has been effectively augmented within the past few decades. In the case of industry, it has been verified that a requirement to assure a good position in the market is the capacity for

producing a large number of product types within a short period of time (Muhl et al., 2003) (Wyns, 1999).

This scenario of production is reinforced by the mass customization tendency, whereby customers have personal choices when they buy products, i.e. they want to buy products that correspond as well as possible to their needs and desires (Da Silveira et al., 2001). This means that production is no longer “pushed” by manufacturers but, rather, “pulled” by customers.

There are a lot of examples of this mass customization tendency in different industrial sectors, e.g. in the clothing, computer, and food industries. A striking example is the automobile industry, whose evolution has allowed consumers to choose among many car models, where each model may have a lot of variants (Muhl, 2002). This evolution is illustrated in Figure 2.



**Figure 2: Automobile Industry Evolution.**

A complementary tendency in developed societies is the on-line mass customization, whereby consumers demand customized products by *non-bureaucratic* means using the present e-technologies, i.e. a type of e-manufacturing (Gouyon et al., 2004)(Da Silveira et al., 2001). This may be called “internet mass customization”, meaning that production is “pulled” online by customers in a context of business to manufacturing (B2M) (Gouyon et al., 2004)(Morel et al., 2003).

A present concern in modern Manufacturing Systems (MS), involved in B2M issues, is the improvement of systemic integration between informatics and automatics subsystems for enabling the e-manufacturing. In this type of MS, another related concern is the improvement of technological support for achieving agility, i.e. competitive delivery response time, in a variable production context (Bongaerts, 1998).

In fact, these improvements aim at supplying the need of the *Internet society*. These concerns are implicated in the so-called enterprise-control integration, whereby high systemic integration and agility are envisaged. Figure 3 illustrates the complexity related to the enterprise-control integration in the internal scope, i.e. from the management system to the plant level, as well as in the external scope, i.e. supply chain management and e-manufacturing<sup>1</sup>.

This expands the traditional setting of Automation Engineering into the Systems Engineering approach<sup>2</sup> (Morel et al., 2003). However, the System Engineering approach is a step toward attaining an agile production environment (i.e. an integrated, automated, efficient, and competitive production environment) established upon an integrated multi-aspect vision of the production system and its environments.

<sup>1</sup> Figure 3 presents the integration beginning at the high-level of ERP (Enterprise Requirements Planning), passing through the integration-level of the SFC/MES (Shop Floor Control – Manufacturing Execution System), and terminating at the low-level of the plant, with SCADA (Supervisory Control and Data Acquisition). Also represented is the horizontal integration with supply chain management as well as the vertical integration with “e- clients”.

<sup>2</sup> See International Council on Systems Engineering, <http://www.incose.org>.

Actually, this type of MS evolution is facing challenges, such as the need for improvements in technologies and paradigms, in order to effectively meet the needs of the next generation of manufacturing systems (Ollero et al., 2003). Therefore, the community related to manufacturing system research and development has invested efforts in MS improvements<sup>3</sup>. For instance, new types of self-organized or agile systems have been studied (Mařík, 2004)(Patrity, 1998)(Tharumarajah et al., 1998).

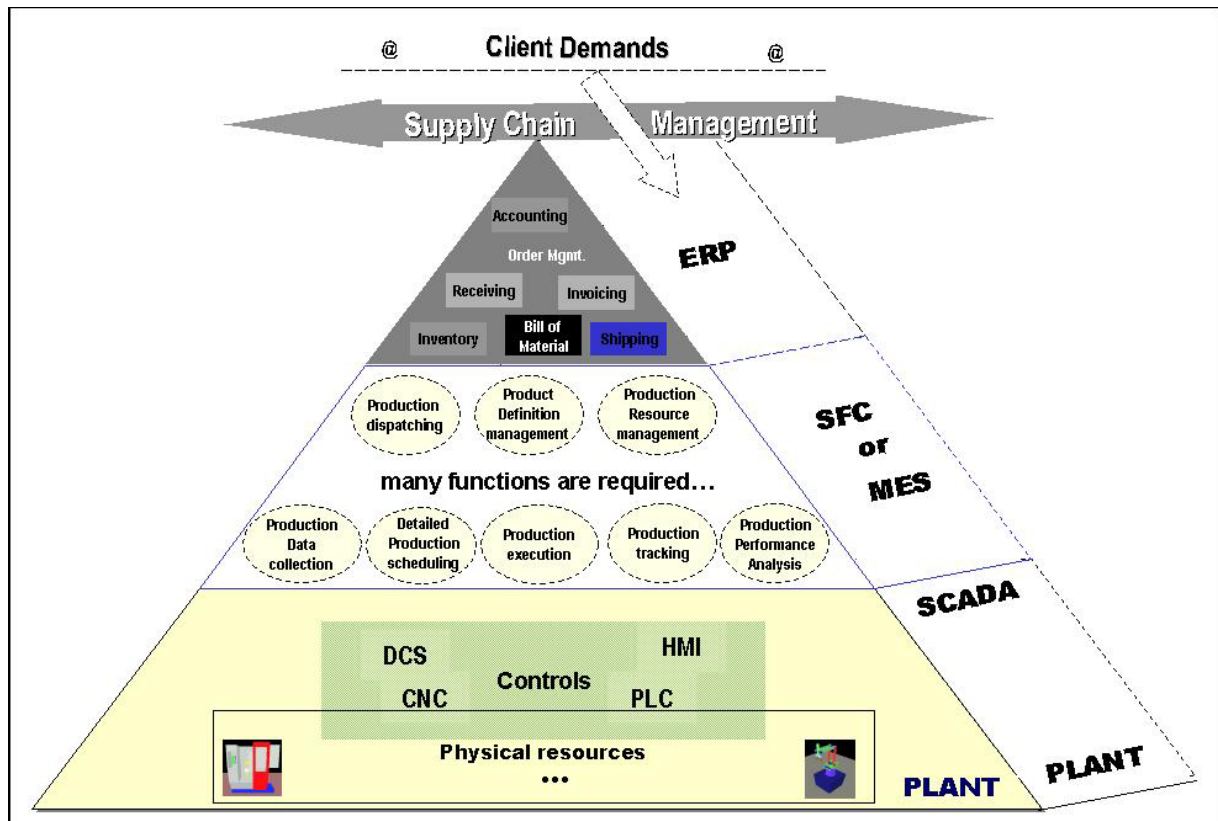


Figure 3: Enterprise Integration<sup>4</sup>.

### 1.3. Agile and Holonic Manufacturing

Agile paradigms have been proposed with respect to the new context of agile production. These paradigms aim at overcoming drawbacks found when non-agile paradigms, currently used in Computer Integrated Manufacturing (CIM), are applied to this new context. Such non-agile paradigms include hierarchical and even heterarchical paradigms (Patrity, 1998).

Hierarchical manufacturing organizations undertake inflexible behavior at lower levels. This imposes a rigid structure that is unable to cope with unforeseen modifications. Heterarchical manufacturing organizations ban all central decision-making. Therefore, it impedes global optimization and prediction of individual orders behavior (Wyns, 1999).

<sup>3</sup> See for example: (a) the broad framework of the industry-led international IMS (Intelligent Manufacturing System [www.ims.org](http://www.ims.org)) discussed by (Yoshikawa, 1995), (b) HMSC (Holonic Manufacturing System Consortium <http://hms.ifw.uni-hannover.de>), (c) IMS-NoE (Intelligent Manufacturing System Network of Excellency <http://www.ims-noe.org>), (d) Auto-ID labs <http://www.autoidlabs.org>, (e) IFIP (International Federation for Information Processing [www.ifip.org](http://www.ifip.org)), (f) FIPA (Foundation for Intelligent Physical Agents <http://www.fipa.org>) and also (g) IEEE Control Systems Society ([www.ieeecs.org](http://www.ieeecs.org)) & IEEE Systems, Man, and Cybernetics Society ([www.ieeesmc.org](http://www.ieeesmc.org)).

<sup>4</sup> Figure adapted from the Siemens' website.

An agile-manufacturing organization must find a balance between hierarchism and heterarchism, endeavoring to maintain the profitable features of each approach, such as behavior prediction in the former approach and flexible strategies in the latter approach (Wyns, 1999).

In Table 1, system architecture approaches are classified, from the primary isolated approach to the advanced agile approach, highlighting some relevant paradigms (Iung et al., 2001)(Morel et al., 2003)(Neunreuther, 1998).

Table 1 presents a classification compatible with the metric EICM - Enterprise Integration Capability Model (Hollocks et al., 1997). This classification facilitates the understanding of the paradigm scopes as well as of the evolution (from 1 to 5) towards the agile paradigms.

**Table 1: Classified approaches.**

System Architecture Approaches	Paradigms	
	Theoretical	Modelling
5. Intelligent, Adaptable or Agile	Fractals, Bionics & Holonics	Multi Agent System (MAS)
4. Heterarchical or Interoperable	Ontologies & Cognitics	Decoupled (Objects) or Distributed System (Agents)
3. Hierarchical Integrated or Visible	Systemics & System Engineering	Computer Integrated Manufacturing (CIM)
2. Hierarchical or Rigid	System Theory	Automatic Control
1. Isolated or Fragmented	Empiricism	Ad hoc approach

Among the agile paradigms, the most emphasized is the holonic paradigm. It originated from a philosophical theory on the creation and evolution of complex adaptive systems in the world, as social systems and evolutionary theory (Bongaerts, 1998)(Deen, 2003).

In holonic manufacturing, the main idea is to make use of the good properties of holonic systems, e.g. agility via adaptability, in the manufacturing context by developing a class of system called Holonic Manufacturing System (HMS) (Morel et Grabot, 2003)(Valckenaers, 2001)(Van Brussel et al., 1998).

The manufacturing entities, such as resources (e.g. equipment, cells and plants) and orders (e.g. rush and normal orders), are improved with some expertise in HMS, as represented in Figure 4. This expertise is added with the goal of providing a more adaptable production environment, by exploiting the existing flexibilities (Wyns, 1999).

The expertise or intelligence can be related, for example, to integration, negotiation, and cooperation capacities, potentially using agents<sup>5</sup> and multi-agents system (MAS)<sup>6</sup> as technological means (Mařík, 2004).

These intelligent entities, each one composed of a manufacturing entity and a related agent, are called “holons”. Holon *etymologically* denotes a kind of equilibrium between autonomy and cooperation skills (Valckenaers et al., 1998).

<sup>5</sup> An agent can be defined as a system (e.g. a software application), or a high-cohesion system module, with well-defined scope, autonomy, and some expertise (e.g. capacity for collaboration, negotiation or even reproduction aiming at goals or desires), which takes part in a certain context wherein it perceives the changes. These perceptions may change the agent behavior, and may promote other changes within the context (Franklin et Graesser, 1996)(Müller, 1998)(Russell et Norvig, 1995)(Schmeil, 1999)(Yufeng et Shuzhen, 1999).

<sup>6</sup> Multi-agent system (MAS) is a community of agents that interact (e.g. through collaboration, negotiation or competition) in order to solve a set of problems. Normally the agents are in a distributed environment and each one has a partial vision of the problem (Franklin et Graesser, 1996)(Müller, 1998)(Russell et Norvig, 1995)(Schmeil, 1999)(Yufeng et Shuzhen, 1999). The use of MAS in research related to the modern manufacturing domain is frequent because of its properties, such as adaptability and reactivity (Mařík, 2004).

Mass customization, for example, has been conceptualized within the scenario of cooperative holons in manufacturing systems with production flexibilities. Each customized production order would be handled by an intelligent-order holon representing a product-instance (Wyns, 1999).

As represented in Figure 4, *smart-order* holons would utilize tasks provided by resource holons and even compete for the opportunity to use these resource holons, based on such factors as the priority assigned to each of the smart-order holons. These tasks would be negotiated between smart-order holons and resource holons, for example based upon customized production needs and present available production skills, with the aim of providing adaptability.

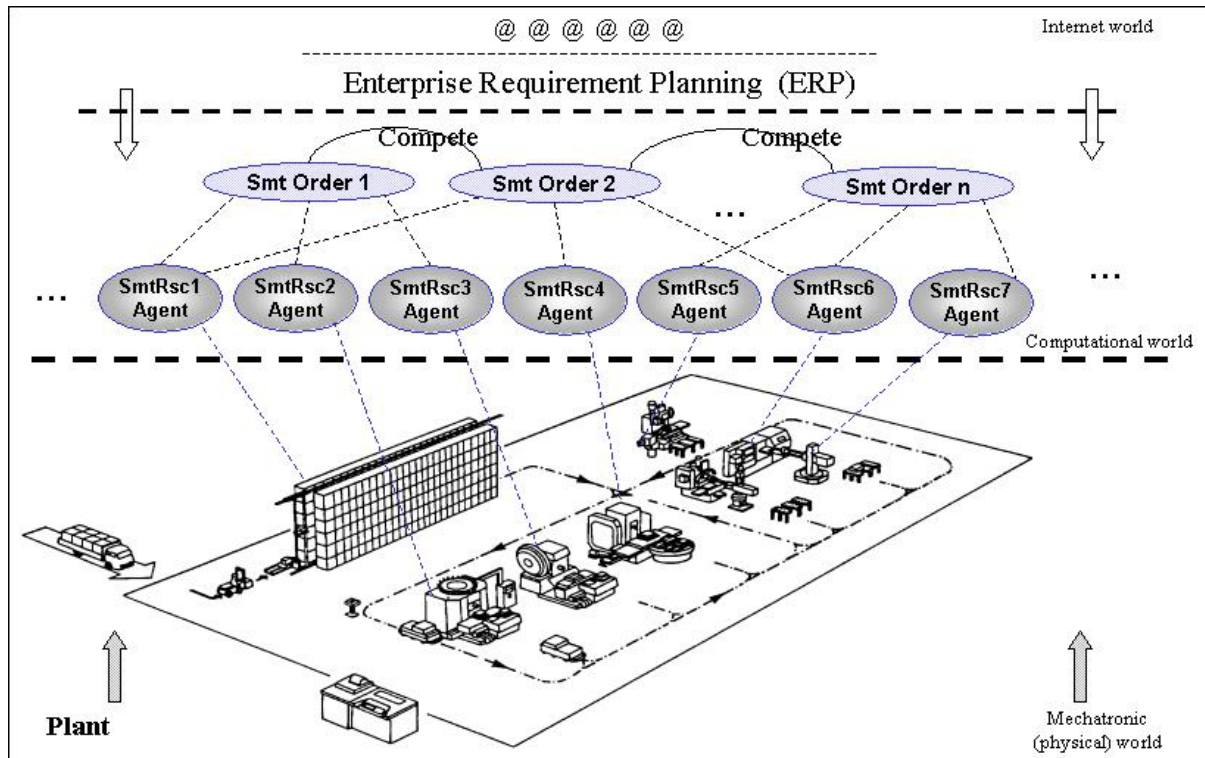


Figure 4: Smart-Entities in a Flexible Plant<sup>7</sup>.

In addition, a new technology has been proposed permitting the smart-order holons to be effectively connected to the corresponding physical product. The technology is called *smart-product*, and it is based on an auto-id tool known as RFID (Radio Frequency Identification)<sup>8</sup> (McFarlane, 2003).

An RFID tag, storing some communicable information, can be physically connected to a product. The illustration in Figure 5 represents the connection between the informational and physical flows by means of information exchanged between “products” and respective agents. Therefore, holons actually related to specific product-instances are envisaged within a holic scenario (Bajic et Chaxel, 1997)(Gouyon et al., 2004).

These smart-product holons would allow informational and material flow consistency, e.g. the information being kept about the physical product state would be effectively synchronized with its spatial position. The flow-consistency helps the agility keeping and the better

<sup>7</sup> Plant drawing adapted from (Stadzisz, 1990), based on the original drawing found in (Hartley, 1984).

<sup>8</sup> See Auto-ID labs <http://www.autoidlabs.org>.

systemic integration, because errors are avoided and accurate production information is enabled (McFarlane, 2003).

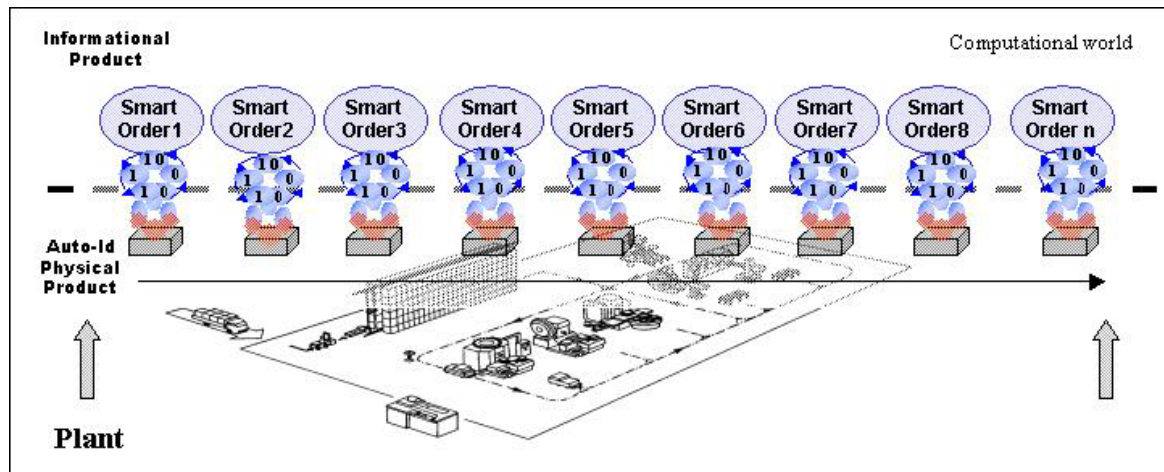


Figure 5: Flows consistence.

### 1.4. Holonic Control

In spite of the agile features in the scenario described in the previous section, only base holons (i.e. smart-resources and smart-products) negotiating in a heterarchical way would be inconsistent with the essence of the holonic approach, which includes a trade-off between hierarchy and heterarchy. In this sense, some flexible rules could be imposed to regulate the base holons, forming a *holarchy* (Wyns, 1999).

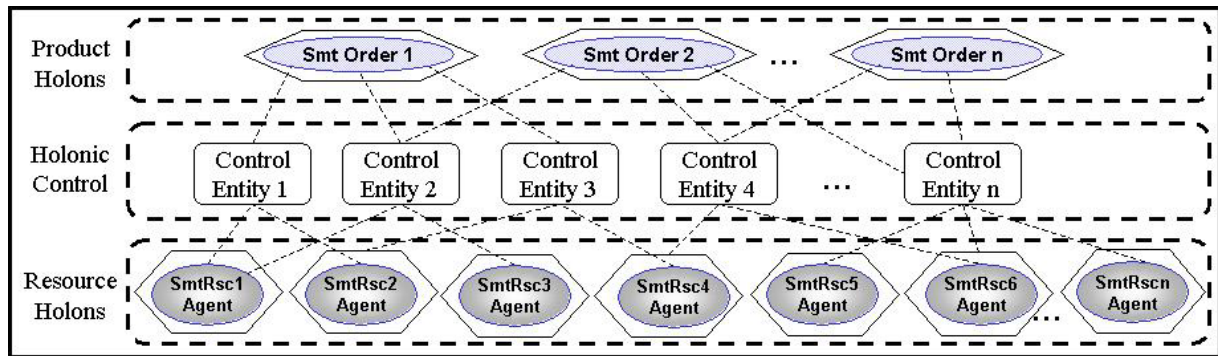
In order to establish flexible rules in the holarchy, decision systems should exist to guide or better regulate the holon society (Bongaerts, 1998). The set of decision systems may be understood as a *Large Holonic Control System (LHCS)* whose subsystems cooperate to maintain a good equilibrium between hierarchy and heterarchy.

A LHCS could be, for example, composed of dynamic planning, an on-line scheduling system, and an agile discrete-event control subsystem. The agile or *Holonic Discrete Control System (HDCS)* could be emphasized, due to its foreseen role as an intermediary control system between discrete base holons and some other subsystems of LHCS, e.g. on-line planner and scheduler subsystems (Bongaerts, 1998).

A system as an HDCS has been proposed as a guider or manager of base holon collaborations, perhaps aided by the other subsystems such as an on-line scheduler (Wyns, 1999). A more specific HDCS function is to mediate among smart-resource holon cooperation (e.g. avoiding deadlock) as well as to mediate the smart-product and smart-resource holon negotiation (e.g. respecting production priorities) when it takes the form of a product-driven control.

HDCS may be conceptualized as an evolution of Shop Floor Control (SFC) or Manufacturing Execution System (MES) into current modern MS (ISA 95 - Part 1)(Qiu et al., 2003). Consequently, HDCS have been called *Holonic SFC (HSFC)* and *Holonic MES (HMES)* (Cheng et al., 2004).

An advanced control system, such as a HSFC/HMES, may be viewed abstractly as a unique entity, but for holonic purposes it would be more appropriate to view it as a composite of decoupled or distributed entities, with each entity regulating some parts of the holarchy (Morel et al., 2003). For instance, the control depicted in Figure 6 would be composed of decoupled entities organizing the resource holon cooperation and their use by smart-product holons.



**Figure 6: Hologic Discrete Control System (HDCS) functions.**

An HDCS distributed in a set of minor entities contributes to the agility of the system, because alternative control entities are possible, parts of the control structure can be functionally independent, errors in control activity can be more easily isolated, loss of a control part does not necessarily cause the breakdown of the entire system, and so forth.

In fact, the essence of HMS is its agile control, and therefore, HMS is sometimes referred to as *control-oriented manufacturing* (McFarlane, 2003). In this sense, research has been proposed presenting advances in control architectures (Bongaerts, 1998)(McFarlane et al. 2003)(Wyns, 1999).

Based on the advances achieved in HMS, the present critical mass of research into advanced manufacturing considers the hologic control paradigm as a promising way to satisfy the requirements for the next generation of manufacturing systems (Deen, 2003)(Morel et Grabot, 2003).

## 1.5. Simulation

A current issue associated with technologies based on hologic paradigms is the testing and comparison of solutions, e.g. control solutions. One of the contexts in which this issue arises is the attempt to deal with the risks and costs associated with implementations within real Manufacturing Systems (MS) (Mařík, 2004). In fact, this is a recurrent issue related to new technologies concerned with MS (Carvalho, 2003)(Tacla et al., 1997)(Tacla et Tazza, 1995).

An alternative widely used for performing tests and comparisons of MS solutions is simulation, which avoids risks and has a lower cost than experiments within real MS (Bako et al., 1990)(Koscianski et al., 1999). In applying this alternative, a potential concern is the availability of a simulation tool comprising a set of concomitant general features (e.g. high quality and low price) and technical features (e.g. customizable application and source code availability) (Koscianski, 2000).

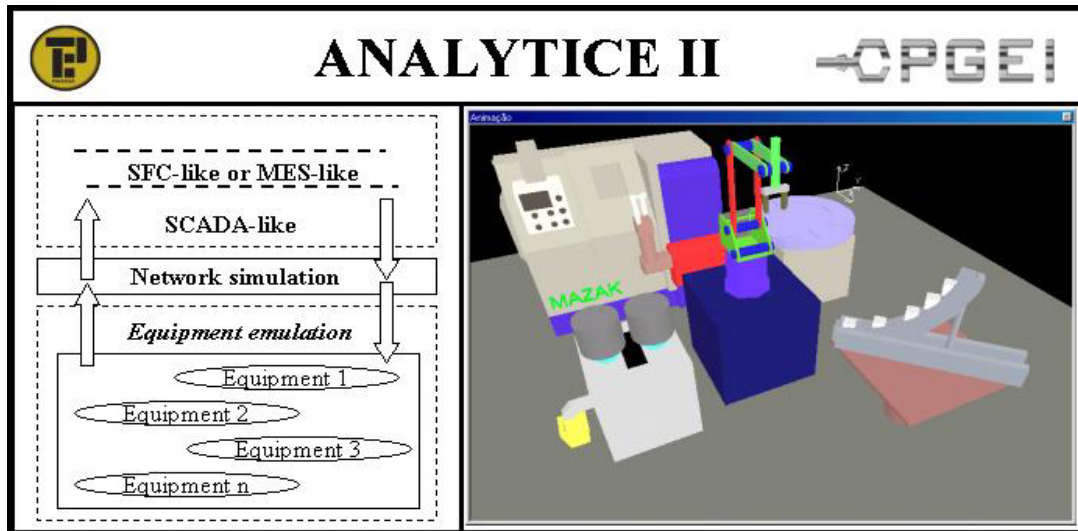
A research effort at LSIP<sup>9</sup> of CPGEI/CEFET-PR is the development of a tool integrating a simulator in a major environment for MS design and analysis, observing systemic application (Koscianski et al., 1999)(Rosinha et al., 2000). This special in-house tool for design and simulation of MS is called ANALYTICE II.

ANALYTICE II is related to one of the educational and research programs at CPGEI/CEFET-PR, namely the program dealing with development of software for industrial applications. In the beginning, CPGEI/CEFET-PR research was concerned with a previous version of the tool called ANALYTICE (Tacla et Tazza, 1995). The present efforts aim at achieving know-how

<sup>9</sup> Laboratório de Sistemas Inteligentes de Produção - Laboratory of Intelligent Production Systems.

in the use and development of this type of tool and related application by means of an in-house solution.

Additionally, it is intended as an alternative design and simulation tool for MS. A special feature of ANALYTICE II is the high degree of modularity and scalability, which contributes to the separation between physical emulation of the plant entities (as mechatronic equipment or work-cells) and control entities (such as SCADA-like and SFC-like) by a virtual communication network.



**Figure 7: ANALYTICE II – An in-house simulation tool.**

Figure 7 illustrates the splitting of entities by the virtual network, on the left side, and the ANALYTICE II graphical module showing a simulated manufacturing cell, on right side. This explicit separation between entities helps simulate MS in a more realistic fashion, because networks separate automatics and informatics subsystems within a real MS. For this reason, the ANALYTICE II could be called a *realistic simulation tool*.

In this sense, a relevant research project would be the development of a realistic holonic simulation tool (Mařík, 2004). With this type of tool, the base holons would be implemented in a realistic way, implying the separation of its software and mechatronic-physical parts by means of a virtual network.

### **1.6. Thesis' Objectives**

A first objective of this doctoral thesis is to propose and to carry out the holonification of ANALYTICE II, an in-house solution and project developed at LSIP/CPGEI/CEFET-PR. ANALYTICE II holonification would fulfill two functions: (1) generating a simulator solution for HMS and (2) synchronizing this tool and related educational and research projects with HMS research.

The ANALYTICE II holonification is similar to real MS holonification, due to its realistic features. The holonification can be achieved by means of the composition of base holons and a related holonic control solution. The composition of base holons could be firstly related to the computational integration of each emulated resource by means of an equivalent and synchronized software agent, i.e. a virtual resource, on the control side of ANALYTICE II.

The virtual resource would entail a degree of *smartness* with respect to the high-level monitoring and service requirements. Briefly, the emulated-resource and the virtual resource would together form the resource holon in ANALYTICE II. Likewise, virtual products could



also be proposed in relation to emulated products, envisaging smart-product holons in ANALYTICE II.

After the composition of base holons, the next step would be the development of a holonic control (HDCS) solution to piloting their cooperation. The HDCS solution must respect the requirements of agile manufacturing, e.g. integration, exploitation of flexibility, and entity decoupling. Moreover, a solution to holonic control implies going beyond process-driven control, to consider product-driven control as well.

Furthermore, a special feature that is aimed at within this control solution is openness to *correctness*. This means that the holonic control solution should consider classical concerns about *correct* controllers or control systems, e.g. determinism, deadlock avoidance, and good performance.

Effectively, this doctoral thesis has as a special objective to define a control solution for carrying out holonic control over base holons, such as those simulated in ANALYTICE II. The targeted control solution is a special objective due to some special features, such as the need to consider all specific features described above, along with the goal of generating a widely applicable solution (i.e. solution *generality*).

The control solution that is aimed at within the research must be generic because it is related to a diversity of production scenarios, such as those that can be created with the base holons simulated in ANALYTICE II. In fact, this generic control solution should be a type of engineering tool to facilitate the composition of agile control over resource holon cooperation.

In short, the control solution must include as a *meta*-feature an appropriate balance of generality and applicability in order to effectively facilitate the development of control systems. The control solution that is aimed at in this thesis is developed as a meta-model<sup>10</sup> of HDCS for HMS. In fact, the main objective of this doctoral thesis is the definition of a particular holonic control meta-model applicable to resource holons, such as those realistically simulated by ANALYTICE II.

Beyond that, an additional and implicit goal is to consider the meta-model of HDCS as a conceptual solution for real HMS, thereby taking advantage of the relation between the real MS and this realistic-simulation tool. In some sense, the main difference between real and simulated holarchies is the way in which the base holons are implemented.

## **1.7. Thesis' Structure**

The second chapter details the doctoral thesis' context, motivations, relevance, and objectives. It is especially important and relevant to gain a deeper multi-aspect understanding of the thesis by means of fitting the above topics within a bibliographical context.

The third chapter presents the ANALYTICE II structure, its resource holonification, and introduces a related holonic modeling approach. The fourth chapter presents a meta-model solution to process-driven HDCS, in which the solution essence is based on the use and development of artificial intelligence techniques.

The subsequent two chapters are concerned, respectively, with correctness and the holonics features of the proposed control solution and with improvements in order to support product-driven control. This is followed by a chapter that presents case studies and by a final chapter that presents conclusions, expresses challenges, and proposes directions for future research.

---

<sup>10</sup> In general, a Meta-Model is on higher level of abstraction (one or more) than the actual model and serves to facilitate the conception of related models (Larsen et al., 2001).



## **Chapter 2**

---

Context and Objectives



# Chapter 2 : Context & Objectives

## 2.1. Introduction

This chapter does not follow the classical approach in which the historical background relating to all aspects of this doctoral thesis is discussed. Instead, the context and objectives of the thesis are dynamically fit within this historical background in order to present the motivations for and relevance of the research project described in this doctoral thesis, as well as the concepts and propositions useful in the development of this research project.

Firstly, section 2.2 briefly highlights important concerns related to agile and holonic manufacturing systems. After that, section 2.3 discusses the shift in paradigms from a standard manufacturing case with “unanimated” entities to an agile manufacturing case carried out by smart-entities called holons.

The discussion about agile or holonic manufacturing facilitates an understanding of the concepts of order-driven and product-driven production by means of holons as well as the need for a Large Holonic Control System (LHCS), constituted by decisional systems. Having discussed the *need* for a LHCS in section 2.3, section 2.4 presents the LHCS itself, while section 2.5 complementarily presents a formalization of the control.

Subsequently, section 2.6 considers one of the properties of holons, their homogeneity, and the implications of this homogeneity for system engineering, focusing on the LHCS design. The section then goes on to present certain formalism regarding holons by means of the Unified Modeling Language (UML), a system-engineering tool. The formalism is applied to present a relevant generic architecture, called PROSA, applicable to LHCS.

Section 2.7 considers the Holonic Control System (HCS) as a kind of decision system within the LHCS. It draws attention to the importance of Control System (CS) across the spectrum from current to holonic manufacturing environments. The subsequent section, 2.8, highlights the complexity of CS and HCS as well as the need for using engineering tools when constructing this kind of system.

Section 2.9 explains the need for using a particular type of engineering tool, a simulation tool, when designing standard and holonic MSs (e.g. in order to test control policies), while section 2.10 explains the essence of the in-house manufacturing simulator, ANALYTICE II, namely its potential utility and suitability when dealing with holonic issues. Subsequently, the last sections present the thesis’ objectives in detail.

Section 2.11 has as subject a brief discussion about the strategy of the ANALYTICE II holonification. Then, section 2.12 relates the challenges involved in developing a generic architecture for holonic control, open to correctness features, in terms of both product-driven and process-driven approaches. This architecture would be both a meta-model focused on applications in ANALYTICE II as well as a conceptual solution to a real MS application.

Splitting the chapter in many sections facilitates the presentation and discussion of the scope and objectives of this work by means of complementary points of view, based on the state of art. In fact, the complexity of the subject matter demands such a multi-aspect presentation and discussion in order to clarify the issues involved.

## 2.2. Agile Manufacturing

Recent research points out that over the next decades there will be developments that may shift the manufacturing scenario to a production model based on smaller units with greater flexibility. Current manufacturing organizations will be replaced by more innovative organic structures whose intention is to offer a very high operational and structural level of flexibility (Tharumarajah et al., 1998).

One example of this shift is already evident, the current focus on agile manufacturing, also called *auto-organized manufacturing*, which will have a deep impact on the design and operation of future manufacturing systems. The auto-organized paradigms can be distinguished by the discipline in which they originate, e.g. mathematics for the fractal factory, nature for bionic and genetic production systems, and philosophical theory on the creation and evolution of complex adaptive systems for holonic manufacturing (Tharumarajah et al., 1998).

An overview of these different approaches is outlined in (Van Brussel, 1994) and in (Tharumarajah et al., 1996). Also, in (Tharumarajah et al., 1998) the holonic, bionic and fractal approaches are classified as agile approaches and some of them are compared. Reviews found in (Bongaerts, 1998) and (Wyns, 1999) summarize other agile-like approaches such as virtual manufacturing, random manufacturing systems, responsibility-based, and RapidCIM.

All these agile approaches focus on the efficiency of manufacturing systems and on how to adapt to a fast-changing environment, e.g. by means of the improvement of system integration, information availability, and control skills (Wyns, 1999). A common practice in these agile approaches is the use of *Artificial Intelligence (AI)* principles. Typical examples are Agents and *Multi-Agent Systems (MAS)* concepts<sup>11</sup> (Mařík, 2004).

The majority of these agile manufacturing paradigms are still within the early stages of their development, making it impossible to draw advanced conclusions (Wyns, 1999). In this context, the holonic-manufacturing paradigm was proposed as an alternative approach for constructing agile manufacturing systems, in the hope that it can better achieve implementation objectives (Tharumarajah et al., 1998).

The community working with issues related to *Intelligent Manufacturing System (IMS)*<sup>12</sup>, such as agile manufacturing, has given attention to the evolution of holonic paradigms presenting research related to *Holonic Manufacturing Systems (HMS)* and generating a critical mass of HMS research<sup>13</sup> (Deen, 2003) (Morel et Grabot, 2003) (Valckenaers et al., 1998).

An HMS is a highly decentralized manufacturing system, consisting of autonomous and cooperating agents, called *holons (HLs)* that respect some flexible control rules, forming a holarchy (Valckenaers et al., 1998). In fact, the HMS is also referred to as a type of control-oriented manufacturing (McFarlane, 2003).

Holonic manufacturing is neither an example of nor an alternative to multi-agent control. Instead, it is a complementary approach, a systems engineering approach to the development

---

<sup>11</sup> In fact, MAS has been used in the scope of manufacturing system research, such as when dealing with control issues. Some examples can be found in (Bongaerts, 1998), (Germain et al., 2003), (McFarlane et al. 2003), (Müller, 1998), (Patriiti, 1998), (Prado et al., 1998), (Simão, 2001), (Wyns, 1999), and (Valckenaers, 2001). However, MAS has too large an application domain, and it is not an engineering archetype for constructing systems, especially within the Manufacturing System domain (Bongaerts, 1998).

<sup>12</sup> See Intelligent Manufacturing System <http://www.ims.org>.

<sup>13</sup> See, for example, the Holonic International Consortium <http://hms.ifw.uni-hannover.de/>.

of control-oriented manufacturing system infrastructures, rather than a solution mechanism for solving individual manufacturing control problems (McFarlane et al. 2003).

McFarlane states that the defining feature of holonic manufacturing is that it represents the only methodology for control system design that manages short and long term changes in the manufacturing environment “as business as usual” (McFarlane et al., 2002).

### **2.3. Holonic Manufacturing**

In HMS, the entities directly or indirectly concerned with production must have certain coefficient of smartness, e.g. related to autonomy, integration, and co-operation, permitting the exploitation of the flexibilities of the manufacturing system, aiming at adaptability, and therefore, agility (McFarlane et al. 2003). A hypothetical transformation from an ordinary production scenario to one that is auto-organized is described here, as a means of presenting the holonic paradigm, thereby facilitating an understanding of the role of entity smartness.

A common manufacturing process is the production in fixed lots. In such a process, the manufacturing entities, e.g. controllers and resources, are prepared and fixed to produce one type, or a few types, of products during a defined period. After that, the orders to effectively start the production arrive from a superior hierarchical level of control (Graham, 1988).

This policy is not appropriate in manufacturing organizations aiming for adaptability, as in the case of mass customization. The response time would be too long for treating each customization in relation to a concurrent market environment (Mařík, 2004)(Ollero et al., 2003). One possible solution is be adding some “expertise” in the production entities, aiming at better exploitation of the manufacturing system potential (Deen, 2003)(Morel et Grabot, 2003).

In this sense, each order, concerned with a specific product-instance, and carrying customized information, could be a smart entity that knows the capacities and states of the resources. Such a smart-order would drive its own production, e.g. allocating and setting the appropriate resources for carrying out its customized operation. Of course, these resources should have a certain degree of flexibility, and even self re-configurability, in order to support the targeted operations (Wyns, 1999).

A smart-order could still plan its production with other similar smart-orders, respecting some imposed restrictions (e.g. production deadline), launching itself, perhaps together with other smart orders, into the production environment in suitable moments (Bongaerts, 1998)(Mařík, 2004). The launching of similar orders together would, for example, optimize the resource utilization, thereby avoiding unnecessary resource reconfiguration (Muhl, 2002)(Muhl et al., 2003)(Sautter, 1991). Generally, such optimization of activities is important in manufacturing systems.

Each resource (e.g. machines or work-station) could be also enhanced with some expertise and improved with some autonomy to assist the smart-order activities. The expertise could be implemented by a high-level interface for showing its enabled skills and present states and for receiving service requests. The autonomy could consist in the execution of the requested services in the most independent way possible, and in self-monitoring by the resource of its own states and skills (Van Brussel et al., 1998).

The resource smartness could be implemented by means of a computational entity, a “virtual resource”, connected to the real physical mechatronic-resource (Mařík, 2004). Such a virtual resource/physical resource pair is considered as a smart-resource (McFarlane et al. 2003). A

smart-resource encapsulates some control services, such as state-monitoring and command services, and enables them at virtual level (Wyns, 1999).

In this described environment, smart-resource and smart-order are homogenized agents at the virtual level, facilitating negotiations for reaching an adaptable production (Minski, 1985)(Wyns, 1999). The smart-order could enter the system and directly negotiate with smart-resources (e.g. about present production possibilities), as well as with other smart-orders (e.g. about optimization or to solve competition problems), launching itself, possibly together with other smart orders, into the smart-production environment.

These smart-entities are like the holons proposed in research into holonic manufacturing<sup>14</sup>. A holon (HL) is currently defined as *an autonomous and cooperative building block of a manufacturing system for transporting, storing and/or validating information and physical objects. A holon consists of an information processing part and often a dedicated physical processing part. In addition, a holon can be part of another holon* (Van Brussel et al., 1998).

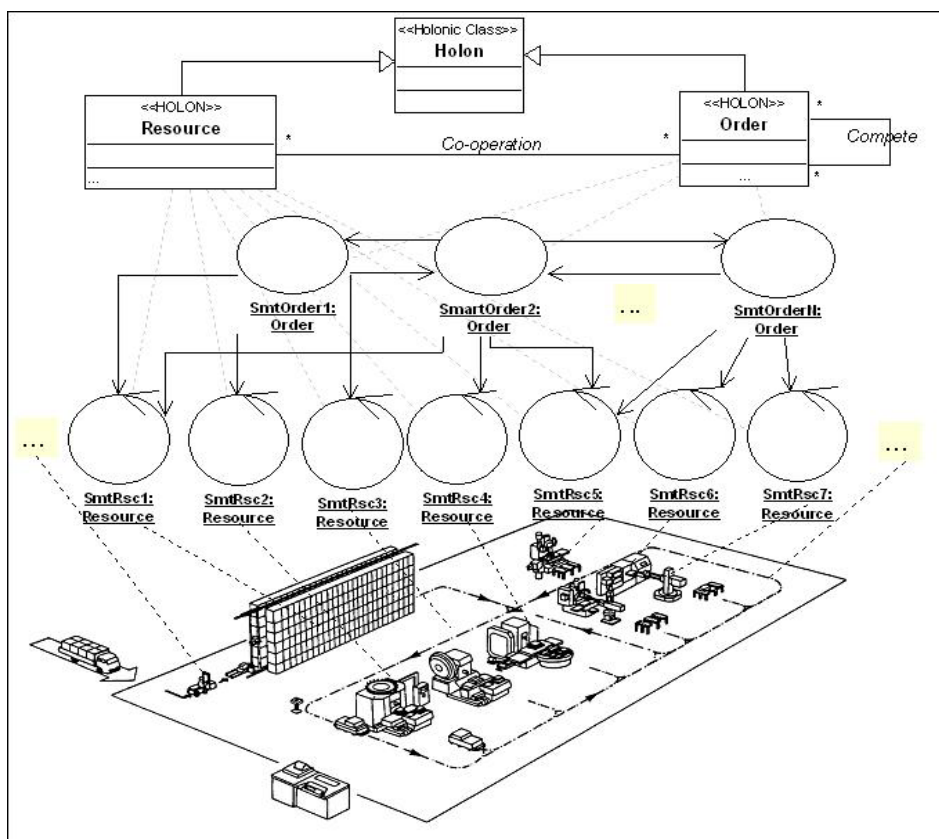


Figure 8: Object and Classes in UML for model holons.

*Resource-HLs* (i.e. smart-resources) are examples that have a physical (hard) part, while *Order-HLs* (i.e. smart-orders) are examples that do not have such a physical part<sup>15</sup>. Therefore,

<sup>14</sup> The word holon is a combination of “holos” (a Greek word meaning “whole”) with the suffix “on”, which suggests “particle” or “part” (such as in the words “proton” and “neutron”). It refers to the hybrid “whole/parts” nature of elements of real-life systems. That is, holons are composed of other holons, so that whether they are viewed as wholes or parts is a relative matter that depends on one’s point of view. They can be considered as wholes, since they include other holons as parts. Yet, they can also be considered as parts, since each holon can be part of another holon that includes it. This observation comes from analyzing hierarchies and stable intermediate forms in living organisms and social organization, and simply reinforces that although it is easy to identify sub-wholes or parts, wholes and parts in an absolute sense do not exist anywhere (Bongaerts, 1998).

<sup>15</sup> Some interpretations of the holonic approach define a holon as necessarily containing a physical part, because even a holon that is more informational (soft) needs an environment (e.g. a microprocessor) in order to survive. Gouyon et al. take this approach when they attempt to formally define a type of holon that they refer to as a *Resource-HL* (Gouyon et al., 2004).



holons may be classified as *Hard-Holons* or *Soft-Holons*<sup>16</sup>. However, holons should be not partitioned into physical (hard) and computational (soft) parts. Conceptually, each holon is an atomic entity, capable of displaying its states and goals, as well as of negotiating with other holons and cooperating to carry out services in a homogenized way (McFarlane et Busmann, 2003).

Since a holon is an auto-contained and cohesive entity, it can be understood as a kind of object or instance in the *Object Oriented (OO)* approach within the field of System Engineering<sup>17</sup>. Figure 8 presents a UML diagram over a sketch of a flexible manufacturing system plant. In this diagram, Resource-HLs and Order-HLs are presented as objects, with their respective classes, in an attempt to use a formalism to represent the entities involved within the plant<sup>18</sup>.

The above-described scenario makes plausible the feasibility of an order-driven production approach to mass customization. However, a set of questions and problems remains, such as the possible inconsistency between information flow (Order-HL) and respective physical flow (products). For example, an Order-HL could contain information indicating that its respective product is in a given place while the product is, in fact, in a different place.

One solution to the above-mentioned problem of inconsistency between physical and informational flows is the integration of informational (soft) and physical (hard) parts, whereby each Order-HL is connected to its respective product, its “hard-part.” The composite resulting from integration of a product and its respective smart-order has been referred to as a *smart-product* (Kärkkäinen et al., 2003).

In fact, research complementary to the research in the area of HMS has also taken place, such as the research into *smart-products*, also known as *intelligent products*. Studies of smart-products attempt to find solutions providing for product traceability, as well as to for integration and agility, within varying production contexts<sup>19</sup> (McFarlane et al. 2003). Moreover, the smart-product and has been considered as a type of holon referred to as a *Smart-Product-HL*, whose physical part is a specific product (Bajic et Chaxel, 1997)(Morel et al., 2003).

Smart-Product-HLs cooperate with other holons to reach their production goals (Gouyon et al., 2004). Resource-HLs and Order-HLs are examples of the types of holons with which Smart-Product-HLs may cooperate. However, within this context of Smart-Product HLs, Order-HLs no longer drive the production. Instead, they have other functions, such as initializing the Smart-Product-HLs or organizing the launching of the production process.

The real assembly cell presented in Figure 9 was holonified as described in (McFarlane et al. 2003). These researchers generated holons similar to those described in this section (i.e.

---

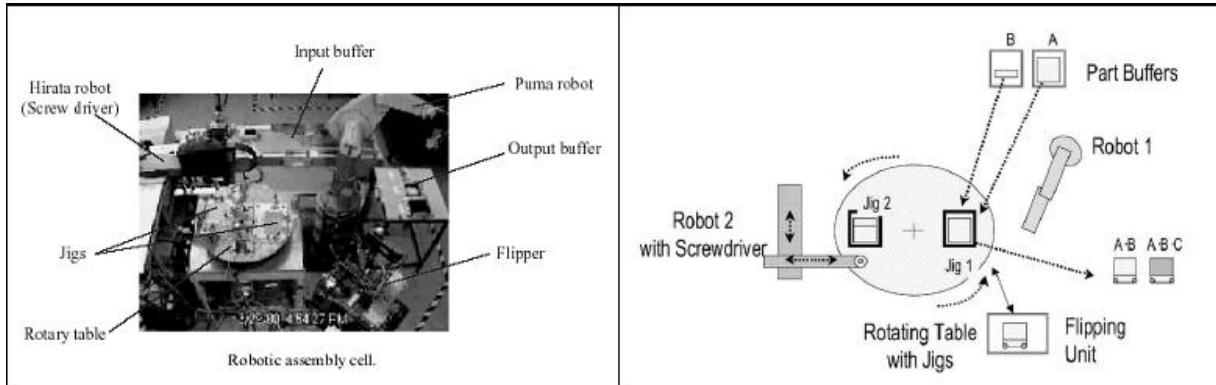
<sup>16</sup> In fact, one of the first practical problems one encounters when attempting to create HMS is the technical difficulty in constructing Hard-Holons in such a way as to preserve the atomicity between soft and hard parts (McFarlane et al. 2003). However, some examples of efforts and viability to construct Resource-HLs can be found in (Van Brussel et al., 1998).

<sup>17</sup> Within the OO paradigm, highly cohesive software or system modules, representing abstract or real entities, are defined as *objects* including methods (services) and data. The classes of *objects* can be defined by entities called *classes*. The OO approach presents a unified formalism called Unified Modeling Language (UML) that allows for the expression of both *objects* and *classes* (Rumbaugh et al., 1999).

<sup>18</sup> Object Orientation is a paradigm and technique that was originally a tool for software engineering. Nevertheless, nowadays, it is considered as a general-purpose technique, and has been applied within other areas of System Engineering as well. This expansion in the application of the OO approach has occurred mainly since adoption of Unified Modelling Language (UML) within the field (see [www.omg.org](http://www.omg.org))

<sup>19</sup> A smart-product is an entity linked to a specific product that “smartly” orients its production (e.g. aiming for agility) communicating/co-operating with other entities in the factory (e.g. aiming for integration) and records information about operations performed on the product (e.g. aiming for traceability) (McFarlane et al. 2003).

Resource-HLs and Smart-Product-HLs). The flexibility to produce two different types of products without a predetermined sequence has been observed in the use of this holons as well as in the flow cohesion that was made possible by the implementation of Smart-Product-HLs using an auto-id technology called RFID<sup>20</sup> (Radio Frequency Identification).



**Figure 9: Holonic Robotic Assembling Cell (McFarlane et al. 2003).**

This research exemplifies the technical viability of the Base-HLs, e.g. Smart-Product-HLs and Resource-HLs. However, it was limited to the scenario where Base-HLs negotiate only among themselves, whereas Base-HLs should be integrated within a smart-environment that also includes other holons whose function is to advise or regulate these Base-HLs. For example, control holons should exist for avoiding heterarchical behavior by Base-HLs and the drawbacks that heterarchical systems entail (Wyns, 1999).

The smart-environment appropriately constituted by a diversity of holons is called a *Holonic Manufacturing System (HMS)*. HMS is also defined as a highly decentralized manufacturing system, consisting of autonomous and cooperating agents, called *holons*, and a *dynamic hierarchical structure*, called ‘holarchy’, which imposes rules upon the holons<sup>21</sup> (Bongaerts, 1998).

## 2.4. Large System Control

Smart-Product-HLs and Resource-HLs interacting in a heterarchical way is insufficient for HMS realization, because this can result in problems such as state unpredictability, system deadlock or state explosion (McFarlane et al. 2003)(Wyns, 1999).

A set of Soft-HLs is necessary to intermediate and organize the Smart-Product-HL and Resource-HL collaboration as well as to control Resource-HL cooperation. This set of cooperating Soft-HLs performs a function known as *Holonic Control (HC)*, and can be abstractly referenced as a unique entity called a *Holonic Control System (HCS)*.

The HCS must also provide some guarantees of agile operability of the system, implying that strong hierarchism is undesirable. Briefly, the adaptability skill should be preserved, requiring a holarchic organization. For that reason, HCS can be considered as the dynamic hierarchical structure, within the HMS introduced in the last section.

<sup>20</sup> An RFID tag containing a microchip, and attached to a product, is capable of storing (at least) a unique identification number and communicating this number via an RFID communication system (McFarlane et al. 2003).

<sup>21</sup> According to the HMS Consortium (<http://hms.ifw.uni-hannover.de>), an HMS is also a holarchy that integrates the entire range of manufacturing activities from order-booking through design, production, and marketing to be carried out within the agile manufacturing enterprise (Wyns, 1999).

The HCS within an HMS may be considered as an evolution of a Shop Floor Control (SFC) within current Manufacturing Systems<sup>22</sup>. The Holonic Control would improve the functionality of an SFC, e.g. resulting in agility in the production process, and justifying the application of the name *Holonic SFC (HSFC)* (Wyns, 1999).

The HCS within an HMS, foreseen as necessary for agile manufacturing, is also compatible with industrial interests and research into *Manufacturing Execution Systems (MES)*, a name currently applied by the industrial sector to SFC (ISA 95 - Part 1) (Qiu et al., 2003)(Wyns, 1999). Therefore, HCS can be understood as a *Holonic MES (HMES)* (Cheng et al., 2004). In fact, HMES is also a research objective in the IMS community (Ollero et al., 2003)(Morel et al., 2003).

In spite of the importance of HCS, it is only one subsystem among a set of other decision subsystems within the HMS, concerned with keeping the manufacturing enterprise agile. The HCS, working independently, is insufficient for regulating the Base-HL collaborations; they require the help of other collaborating subsystems that contribute in some way to the achievement and maintenance of agility.

The other decision subsystems might include, for example: *Holonic Planning (HPla)*, *Holonic Scheduling (HSch)*, *Holonic Flow Management (HFM)*, and *Holonic Fault Supervision (HFS)*<sup>23</sup>. However, a decision subsystem could be part of another decision subsystem, since in HMS the notions of whole and parts are relative. In order to better understand complex holarchies, we may group together a set of these decision subsystems that cooperate to perform a determinate function, and refer to this set as a *Large Holonic Control System (LHCS)*.

The role of the LHCS is to help organize and perform holarchy dynamics, where the dynamics are constituted by the interaction of all “heterogeneous” holons. We refer to these holons as “heterogeneous” because they are related to different parts of the HMS, e.g. Soft-HLs are related to the enterprise corporate level and Hard-HLs are related to the manufacturing shop floor level (Mařík, 2004). LHCS (a type of manufacturing control) should allow agile production respecting B2M issues, such as the online mass customization<sup>24</sup>.

---

<sup>22</sup> *Shop floor control (SFC) is the group of activities directly responsible for managing the transformation of a planned order into a set of outputs. It governs the short-term (on-line) detailed planning, execution, and monitoring of process preparation and resource allocation activities needed to control the flow of an order from the moment that it is released by the planning system for execution until the order is filled and completed. In order to fulfil its responsibilities effectively and efficiently, shop floor control systems may contain activities such as: reactive scheduling, order-release, assignment of resources (labour, machines, tooling, materials, ...) to orders, on-line process planning, downloading of process parameters, data collection and monitoring, etc. Shop floor control (SFC) is often also referred to as Manufacturing Activity Planning (MAP), Production Activity Control (PAC), and Manufacturing Execution Systems (MES) (Wyns, 1999).*

<sup>23</sup> As a first approximation, each of the decision subsystems within the HMS’s control system (LHCS) may be defined as fulfilling one of the decision subsystem functions observed in a “similar” but real modern MS, e.g. HPla is concerned with process plan elaboration, HSch is concerned with master scheduling, HFM is concerned with online scheduling, HFS is concerned with fault- detection/recovery, and HC with piloting issues. But, when necessary for holonic purposes, such functions and/or integration levels may need to be modified (or redefined).

<sup>24</sup> In some way, LHCS is equivalent to the (large) manufacturing control operating within current manufacturing organizations. Manufacturing control is used as a generalization of shop floor control. It covers similar responsibilities but it may take into account a broader space and/or time domain. While shop floor control is limited to one part of the factory, manufacturing control may consider relations across different shops, or even across different plants. Manufacturing control may also incorporate mid-term or long-term objectives of the manufacturing system. Therefore, when the term manufacturing control is used, it may refer to a wide scope of diverse systems, ranging from the control of an individual cluster of machines up to the management of several plants, including purchasing, material requirements planning, design, process planning, etc. Alternative terminology used for manufacturing control system is manufacturing planning and control system (MPCS) (Wyns, 1999).

## 2.5. Control Formalization

In order to facilitate a more formal comprehension of holonic control, it is interesting to consider the formalism addressed in (Fusaoka et al., 1983), where the process for automating a system consists in satisfying the conditions expressed in Equation 1. Basically, the dynamics of a system and its control must allow reaching the goals established for the system.

$$\text{Dynamics} \wedge \text{Control Rules} \supset \text{Goal}$$

**Equation 1: Fusaoka's Equation.**

Within the manufacturing context, the automation process involves the definition of the *unknown control rules* of the *known dynamics of physical processes*, for the purpose of reaching *goals globally assigned to the system*<sup>25</sup>. The dynamics is in the form of discrete-time behavior, the control rules are in the form of discrete-event behavior, and the goals are in the form of information-based behavior (Morel et al., 2003).

In this sense, LHCS composition could be formalized by the follow equation interpretation: (a) the dynamics as known hard-holons; (b) the control rules as unknown LHCS entities; and (c) the goal as agility. This approach formally opens the control system field to the development of agile manufacturing systems.

Similarly, the equation can be interpreted within the context of the construction of hard-holons. For example, in the Resource-HL case, the equation could be: (a) the dynamics as the resource; (b) the control rules as the unknown virtual resource; and (c) the goals as software integration and smartness.

In fact, the hybrid nature of real manufacturing systems compels this discipline to address theoretically the challenging issues relating to one type of integration, namely, the integration of event-driven components and complex time-driven components, in order to better take into account system scalability (Morel et al., 2003).

## 2.6. Holonic Thinking

One goal sought within research into manufacturing systems is systemic integration by means of the application and development of control systems such as MES (ISA 95 - Part 1). In a general way, a factor that complicates the integration of systems is their heterogeneity (Minski, 1985). A Manufacturing System (MS) is a composite of heterogeneous subsystems and this makes difficult the construction of an integrated system (Wyns, 1999).

In HMS, all holons are homogenized and integrated at the interface level. This means that all holons negotiate by means of exchanging information, e.g. by presenting their services or requesting services, without making public either how they perform these services or how they are internally structured (e.g. the agents and specific controllers contained within them).

As specific examples, a holon encapsulating a robot and another encapsulating a lathe are considered as similar entities within the holarchy, since all holons are service providers. The holon in the first example would be responsible for transportation services, whereas the holon in the second example would be responsible for lathing services. An HCS could request these services at a high informational level without needing to be concerned with subtleties in their execution, because details of execution are encapsulated internally within each holon.

<sup>25</sup> This formulation is compliant with System Theory (Cassandras et Lafortune, 1999), which remains the scientific foundation for Automatic Control of the dynamic behavior of manufacturing systems (Morel et al., 2003).

Conceptually, holons are implicitly integrated within the holarchy. They are thought of as encapsulating specific processes related to services, hiding the details of the services, while providing access to these services from within the holarchy by means of high-level computational interfaces.

Holons may be understood as either smart-objects, from within the Object Orientation (OO) paradigm, or as agents, from within the Distributed Artificial Intelligence (DAI) paradigm. Similarly, the negotiations and service *instigations* (service requests) are performed via informational exchanges, while the methods for carrying out these services are encapsulated inside of each entity (Mařík, 2004). In the case of holons, these services may be either informatics (computational) or automatics (informatics and automation) services.

Therefore, at the interface-negotiation level, all holons (e.g.  $h_0, h_1, h_2, \dots, h_n$ ) pertain to the same computational universe  $H$ . This homogeneity allows for modeling holon interactions independently of their internal implementations. This modular treatment facilitates the system engineering process, since functionally independent steps within the HMS design (e.g. the design of the holons and the definition of their interactions) are effectively treated in a separated ways.

Despite the benefits of the holonic thinking, engineering tools are still needed for modeling holon interactions. For example, some formalism is necessary for modeling interactions of the control entities with Resource-HLs and Smart-Product-HLs when designing Holonic Control Systems (HCSs) for HMS.

The system engineering school has used the *Unified Modeling Language (UML)* to model interaction of entities such as smart-objects, agents, and holons, by providing a simple interpretation of these entities as objects within the OO paradigm. However, UML could also be used to help in certain formalization of the holon concept, as well as to provide a more fundamental modeling terminology for the composition of HMS and its subsystems, such as the Holonic Control System.

The UML includes a meta-model for defining, at a high-level, the generic entities that may be included within lower-level UML diagrams of applications<sup>26</sup>. A fundamental *meta-entity* is the *meta-class Class* that defines the *Class entities* that may be included within UML models. In Figure 10, the *meta-entity Holonic-Class* is derived from the *meta-entity Class* at the UML meta-model level (Morel et al., 2003), thereby making the *Holonic-Class* available for use in lower-level UML models<sup>27</sup>.

Once the *Holon-Class* has been defined at the meta-model level, as in Figure 10, the possibility is opened of including instances of the *Holon-Class* at the UML model level. This allows a kind of formalization of the holon concept, because the holon existence within UML diagram at the model level is effectively defined within the UML meta-model.

This derivation of the *Holonic-Class* from the *meta-entity Class*, at meta-model level, also allows us to define a terminology for use within UML models of holonic systems:

- *Holonic-Class* – a *Meta-Class* derived from the UML *Meta-Entity Class*.
- *Holon* – the *Meta-Instance* of the *Meta-Class Holonic-Class*. It represents a class of *Holonic-Instances* at the model level. Instances of the *Holon Holonic-Class* are *Holon-*

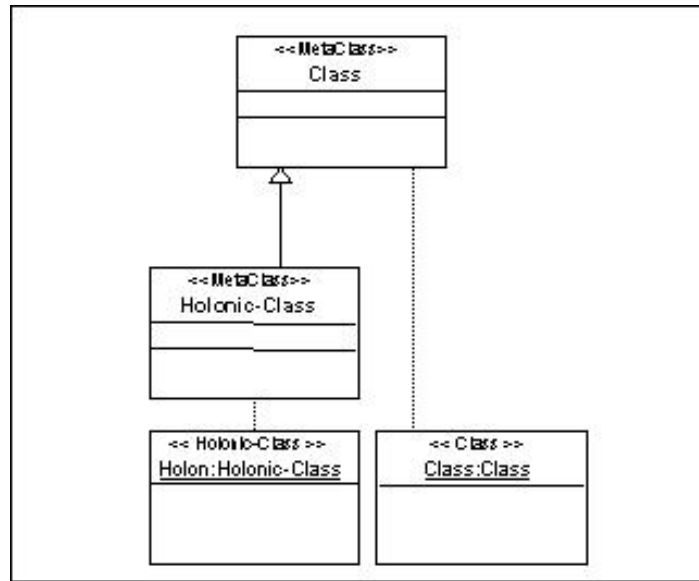
---

<sup>26</sup> UML 2.0 superstructure draft specification, OMG, 2003, [www.uml.org](http://www.uml.org).

<sup>27</sup> The instances of *meta-classes* in *meta-class diagram* of Figure 10 means *classes* in a *class diagram*. The UML was interpreted in this way to well split the meta-model and model.

*Instances.* *Holonic-Classes* derived from the *Holon Holonic-Class* have *HOLON* as their stereotype.

- *Holon-Instance* – an instance of the *Holon Holonic-Class* or of a class derived from the *Holon Holonic-Class*. *Holon-Instance* has as its stereotype the name of its *Holonic-Class* or the name of the *Holonic-Base-Class* from which its *Holonic-Class* is derived. A *Holon-Instance* may be more simply referred to as a *holon*<sup>28</sup>.
- *Holonic-Relation* – an association or aggregation of *Holonic-Classes* or *Holon-Instances* (i.e. *holons*).
- *Holonic-Diagram* – any UML diagram including one or more *Holonic-Classes* or *holons*<sup>29</sup>.



**Figure 10: UML meta-class diagram.**

The terminology presented above can be useful for modeling the entities and subsystems that compose the HMS. For example, using this terminology, it is possible to construct models for specific HCSs, or even meta-models for representing an HCS or an LHCS in a more generic way.

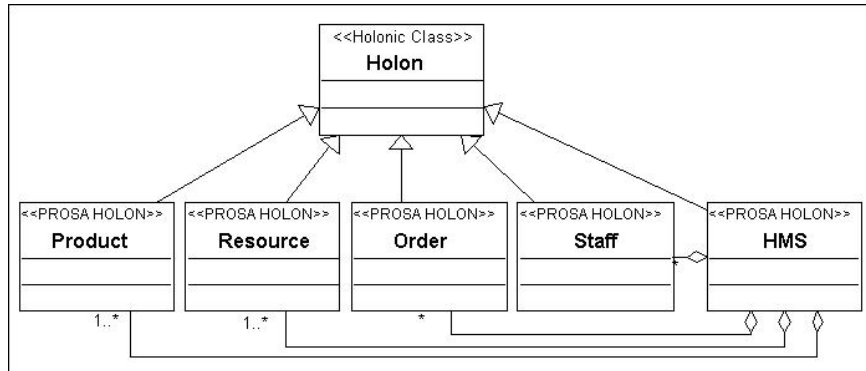
However, the modeling and development of an HMS requires other system-engineering techniques in addition to definitions and terminologies. For example, models and meta-models are needed in order to provide some formalization and to help in the design of HMS subsystems (such as those related to the LHCS) and their inter-relations (Gullander et al., 1998)(Morel et al., 2003).

In order to meet the requirement described above, a reference architecture for holonic control within a manufacturing system, called PROSA (Product-Resource-Order-Staff Architecture)

<sup>28</sup> Additionally, each *holon* mentioned without direct identification is written in the follow way: by the name of its *Holonic-Class* followed by a dash and the name of its stereotype, i.e. *Holonic-Class Name - Stereotype Name*. For example, if a *holonic-class Smart-Product* is defined, each instance must be referred as a *Smart-Product-HOLON*. A *Smart-Product-HOLON* means a (undetermined) *holon* (an instance) of the *Holonic-Class Smart-Product*. Still, the *Stereotype Name* can be substituted by its acronym, e.g. an alternative to *Smart-Product-HOLON* would be *Smart-Product-HL* once *HL* is the acronym of *HOLON*.

<sup>29</sup> For example: (a) *Holonic-Class Diagram* - a class diagram from UML with *Holonic-Classes*; (b) *Holonic-Instance Diagram* - an instance diagram from UML with *holons*; (c) *Holonic-Sequence Diagram* - a sequence diagram from UML with *holons*; and (d) *Holonic-Activity-Diagram* - an activity diagram from UML whose activities are relative to *holons*.

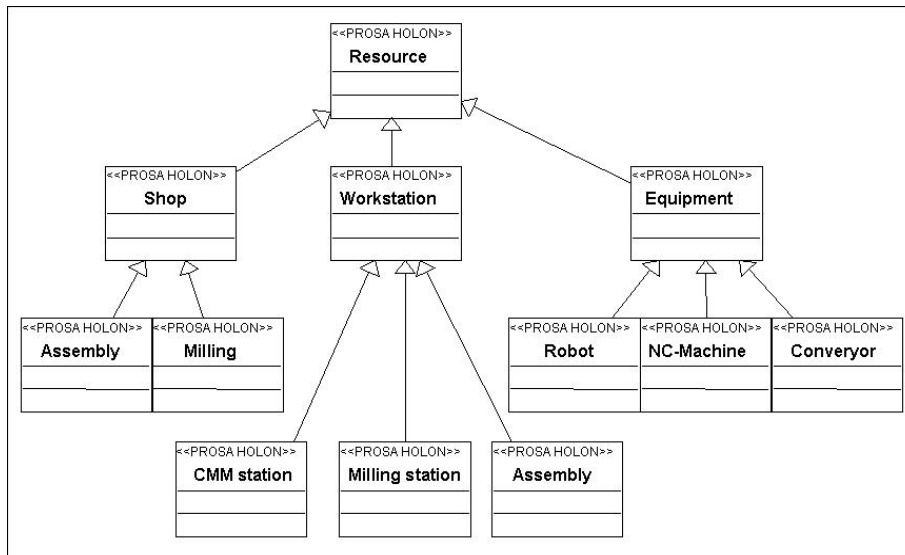
(Bongaerts, 1998)(Wyns, 1999), was developed. PROSA presents a set of useful concepts for the development of models and meta-models of subsystems within the scope of HMS, as well as for entire HMS.



**Figure 11: Main holonic-classes in PROSA.**

PROSA is an architecture that has been frequently referenced in the literature because it comprises a set of concepts that are well accepted within the IMS community. Basically, PROSA defines four classes of holons, namely *Product*, *Resource*, *Order* and *Staff*, as a generic Architecture. These classes of PROSA holons (PHL) are presented in Figure 11 using the terminology defined above (Wyns, 1999).

The *Order Holonic-Class*, for modeling *Order-PHLs*, brings together the generic models of the aforementioned *Order-HLs* and *Smart-Product-HLs* within one, unique, modeling entity. The *Resource Holonic-Class*, for modeling *Resource-PHLs*, is the same aforementioned generic model used for *Resource-HLs*. Additionally, PROSA proposes subclasses for the *Resource Holonic-Class*, as presented in Figure 12.



**Figure 12: Resource-HL specialization.**

The *Product Holonic-Class* is the generic model for *Product-PHLs*. Each *Product-PHL* is a kind of knowledge supplier to some *Order-PHLs*. For example, when an *Order-PHL* is created, its knowledge about appropriate *Resource-PHLs* to carry out each production step comes from a concerned *Product-PHL*.

The *Product-PHL* contains information concerning *Order-PHLs* instanced from the same (sub) type, in particular about the suitable *Resource-HLs* for these *Order-PHLs*. Since this

information is provided to the *Order-PHLs* at the time they are created, they do not need to query *Resource-HLs* to determine if they are appropriate<sup>30</sup>.

The *Staff Holonic-Class* is the generic model for *Staff-PHLs*. The *Staff-PHL* is responsible for advising the other holons, defined in PROSA, about their collaborations (e.g. providing information helpful for avoiding inappropriate interaction such as ones resulting in deadlocks). It is a kind of control entity, used to avoid heterarchical behavior.

The Staff is defined in a very abstract way. In order to be useful, it would be necessary to derive classes from *Staff*. For example, each LHCS subsystem could be modeled by deriving it from the *Staff class*, even if a *Staff-PHL* is seen, sometimes, as a monolithic entity in PROSA.

As a kind of drawback, PROSA lacks mechanisms for constructing *Staff-HLs* and mechanisms for *predefining* the activities (“dynamics”) of *Staff-HLs* (e.g. mechanisms for suitable advice concerning Base-HL interactions). In short, PROSA presents a very generic *Staff-HLs* definition without concern about relatively easy applicability.

The set of concepts and formalism presented here will be useful in the next section, as well as in development of the research project described in the current doctoral thesis. The next section presents the concept of Holonic Control as a distributed and complex system, accentuating its important role as an intermediary entity between Base-HLs and other LHCS subsystems.

## 2.7. Holonic Control

Among possible subsystems within a *Large Holonic Control System (LHCS)*, there exists the *Holonic Control System (HCS)*, also known as *Holonic Discrete Control System (HDCS)*. A HDCS is abstractly viewed as a unique controller entity, however it is interpreted here as a set of soft-holons exercising control functions within the HMS<sup>31</sup>. The HDCS is especially interesting because of its remarkable role in the foreseen HMS.

The HDCS is responsible for organizing the cooperation of Resource-HLs by means of service requests based on decisions it makes. These decisions could be based on Resource-HL capabilities and states, i.e. discrete-information about the process, and internal knowledge of control entities, e.g. specialized information so as to avoid deadlock or consider production priorities.

Additionally, in the product-driven case, the decision should be based on Smart-Product-HL “desires”, i.e. specific goals in the form of discrete production steps determined via the information from its process plan<sup>32</sup>. Figure 13 shows a set of HDCS entities as intermediate entities, operating between Resource-HLs on the one hand and Smart-Product HLs on the other. As indicated on the left side of the diagram by means of classes, these HDCS entities

---

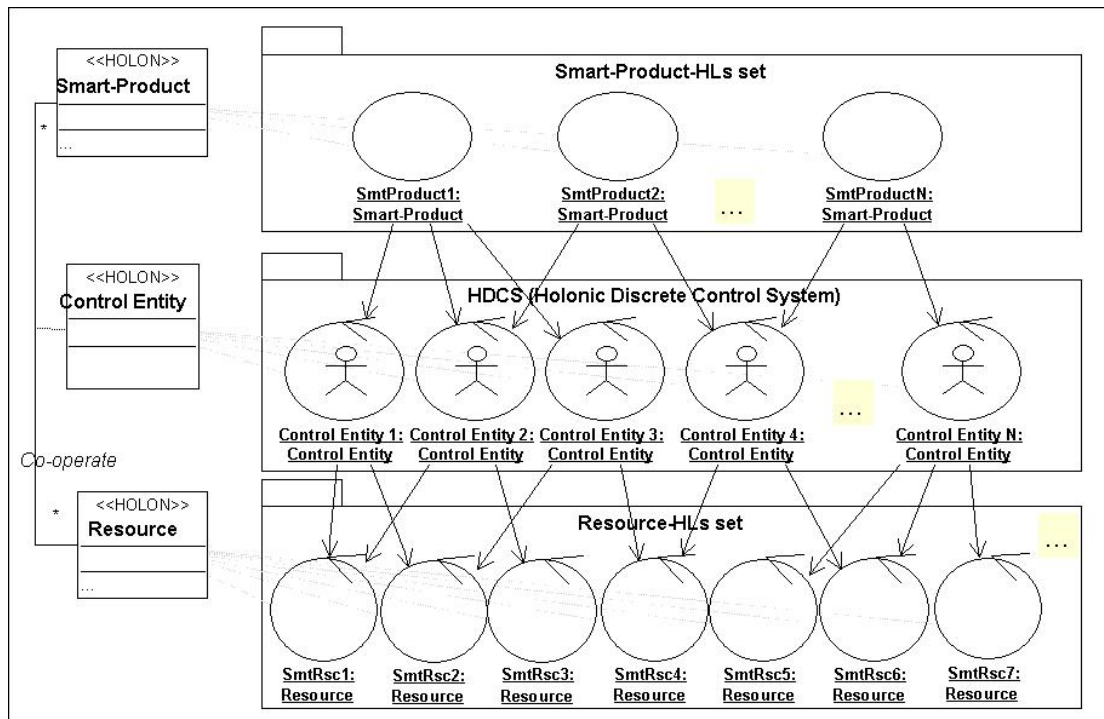
<sup>30</sup> The *Product-PHL* concept is useful for the project described within the current doctoral thesis, but it will be redefined as a *Product-Type-HL* related to *Smart-Product-HLs*.

<sup>31</sup> The common thread in recent advances in this area is the willingness to broaden the control mindset beyond the boundaries of classical control-theoretic approaches by bringing it closer to Computer Science techniques in order to achieve distributed networked automation, by including more and more software and Information Communication Technologies (ICT) in the control. *The main concern in achieving agile automation is that the notion of System has been so closely associated with hierarchical models that to regard information ordering as a dynamic and emergent characteristic of self-organizing complex systems is to challenge the current way of thinking that order can only occur through centralized control imposed through a prescriptive system control design, whether or not based on intelligent techniques* (Morel et al., 2003).

<sup>32</sup> Process Planning: An element that defines how products can be made, including the overall process plan defining the possible sequences of operations for the manufacturing of a product and the detailed operation process planning, prescribing the parameters of the individual manufacturing processes (Bongaerts, 1998).



serve to control cooperation among Resource-HLs, as well as between Smart-Product-HLs and Resource-HLs.



**Figure 13: Holonic Discrete Control System.**

The HDCS's decisions may be also based on advice from Soft-HLs in other decision-making subsystems<sup>33</sup>, preserving a systemic approach. The HDCS is responsible for holonic control in the HMS, organizing holon collaborations, directly influencing the agility of production. This agility is concerned with adaptability based on exploitation of flexibility. Such adaptability could involve, for example, re-configurability skills to deal with changing production priorities and even system faults.

A HDCS should exploit Base-HL knowledge and skills (i.e. their *potential*) in order to effectively achieve HMS agility, supporting a changing production context. HDCS could also emerge from some negotiation mechanisms between holons and, justly by this emergence, to know the *potential* of the holons over its holonic control. The HDCS emergence could even be viewed as part of its own scope (Mařík, 2004).

In some way, as previously discussed, an HDCS is equivalent to the Shop Floor Control (SFC) system at current manufacturing organizations. There does not exist a universal consensus in the academy regarding the nomenclature for describing an SFC, or the responsibilities entailed by an SFC (Wyns, 1999). Within industry, SFC is referred to as a *Manufacturing Execution System (MES)*, where it is aimed at arriving at a consensus about MES by means of normalization (as is done in ISA-95<sup>34</sup>), such as has been proposed by specific organizations (such as MESA).

<sup>33</sup> For example, soft-holons from HFM (Holonic Flow Management) advising the HDCS about how to optimize the fabrication flow.

<sup>34</sup> ISO and ISA are establishing a joint working group to develop ISA-95 on Enterprise Control System Integration. The resulting standard IEC/ISO 62264 Enterprise Control Systems Integration is a multi-part set of standards that defines models and terminology for defining the interfaces between an enterprise's business systems and manufacturing control system. It describes in a rather detailed way the relevant functions in the enterprise and the control domain and the objects normally exchanged between these domains. It is becoming the accepted model for B2M integration (Morel et al., 2003).

Generically, an SFC is understood as the entity responsible for deliberating about possible options among production processes and, in appropriate moments, for specifying or requesting services<sup>35</sup> from the factory elements (e.g. *units*, *process cells*, and *even areas*<sup>36</sup>) to carry out the production process at the plant level.

The deliberation, and consequent service requirement carried out by the SFC, takes into account predetermined plans from manufacturing management, for example, planning and scheduling. The implementation of these plans has been related to the Enterprise Requirements Planning (ERP)<sup>37</sup>.

The SFC is the element between the manufacturing management level and production level in modern *Computer Integrated Manufacturing (CIM)* (Bongaerts, 1998). SFC is similar to what is referred to as *Supervisory Control (SC)*, both being types of controllers. The difference between the two lies in the scope within which they apply their control functions.

A SC exercises control within the scope of a *process cell*, or even of *units*. Normally, it is responsible for supervising and coordinating the production entities, such as equipment modules (Chaar et al., 1993)(Silva et al., 1998)(Simão et al. I, 2001). SC is essentially a dynamic event-driven system whose controlled variables are manipulated as discrete states (Künzle, 1990)(Miyagi, 1996).

In terms of HMS, *Holonic SFC (HSFC)* is the HDCS at the Area-HL level, while *Holonic SC (HSC)* is the HDCS at the Process-Cell-HL level, or even at the Unit-HL level, as a lower similar level of the HSFC. Therefore, the Process-Cell-HL or even Unit-HL encapsulates HSC responsibilities, being potentially a Resource-HL at HSFC level.

In fact, the Process-Cell-HL, for example, encapsulates a Sub-HMS in a fractal point of view<sup>38</sup>. However, the organizational structures in HMS are flexible, for example a Resource-HL may be considered in both HSC and HSFC scopes if necessary.

## **2.8. Control Complexity**

An SFC or SC is, in general, a complex system due to the required automation level. Both are related to concerns about robustness (for security and costs), scale (large number of variables and states), and the relationships among its components (dependency between its variables)<sup>39</sup> (Chaar et al., 1993)(Mendes 1995)(Miyagi, 1996).

Other related requirement for controllers (e.g. SFC and SC) may also augment the complexity level surrounding this subject. Examples of these requirements are presented in Table 2. The

---

<sup>35</sup> Interfacing with support system: such as SCADA (Supervisory Control and Data Acquisition).

<sup>36</sup> This nomenclature is consistent with ISA-95 being equivalent to: workstations, manufacturing-cells, and plants.

<sup>37</sup> Enterprise Requirements Planning (ERP) system is a concept developed by Gartner Group describing the next generation of manufacturing business systems and manufacturing resource planning (MRP II) software (see <http://www.intentia.com>). It is an integrated suite of software applications to address their financial, order management, materials management and manufacturing information requirements. It incorporates the client/server architecture and uses graphical user interfaces (GUIs) (Bongaerts, 1998).

<sup>38</sup> The main characteristic of fractals is self-similarity, implying recursion, pattern-inside-pattern. An element set displays self-similarity not only because its elements occur in greater detail, but also because these elements show details with certain constant proportions or ratios among these elements, though these elements are not identical (Tharumarajah et al., 1998).

<sup>39</sup> The control of discrete event systems, such as SFC/SC, has been studied (Cassandras et Lafortune, 1999). The specialized literature presents approaches and methods for formalism and implementation (Chaar et al., 1993). There exist approaches proposing formalisms from synthesis (Ramadge et Wonham, 1987)(Cury, 2000) to generic architecture regarding modeling, implementation, integration or simulation (Brugali et al., 1998) (Langer et al., 2000) (Schmid, 1995) (Simão et al. II, 2003)(Wyns, 1999).

control architecture that solves these items from Table 2, or presents clear interfaces to solve them, is considered as a correct solution in this thesis.

The complexity related to control issues is increased in the holonic approach, which requires system distribution (e.g. for efficiency), holarchy integration (e.g. for co-operability and systemic coherence), agility (e.g. for competitiveness and just-in-time issues), and even smart-product orientation (e.g. for customization and traceability) (Bongaerts, 1998)(McFarlane et al. 2003)(Wyns, 1999).

**Table 2: Possible requirements related to controller correctness.**

Possible Controller Requirements	Some Concerned Bibliographies
Trade-off between determinism and reactivity.	(Char et al., 1993) (Pan et al., 1998) (Lhoste et Pétin, 2001)
Deadlock Avoidance	(Bongaerts, 1998) (Char et al., 1993) (Wyns, 1999)
Conflict Identification and Resolution	(Cardoso et Valette, 1997) (Miyagi, 1996) (Simão et al. II, 2003)
Computational Performance	(Bako et Valette, 1990) (Pan et al., 1998) (Simão et Stadzisz, 2002)
Formal Models	(Cury et al., 2004)(Lüders, 2001) (Lima et Dórea, 2002)
Automatic Synthesis	(Eyzell et Cury, 2001) (Gouyon et al., 2004) (Lima II et Dórea, 2004)

This inherent complexity in HDCS and other implicit issues pertaining to complex systems (such as risks, costs, and design time) creates the need for the development of methods and the use of appropriate system engineering support tools for modeling, design, implementation, and tests.

A current issue within the holonic manufacturing, as foreseen by the research community, is the need for the development of proper models and implementation for both process-driven and product-driven HDCS in HMS (Mařík, 2004)(McFarlane et Bussmann, 2003). Within the scope of system engineering, the specific need is for the development or improvement of engineering tools, such as meta-models or architectural patterns, to aid in the engineering process of holonic controllers<sup>40</sup> (McFarlane et Bussmann, 2003)(Wyns, 1999).

These modeling tools can be related to a diverse set of engineering objectives, such as: (a) reduction in the time required to construct the system; (b) ease in the creation of instances, e.g. semi-automation or automation of instance-creation; (c) patterns, e.g. for system scalability, pre-tested solutions, and reusability of entities; (d) trade-off between generality and applicability; and (e) formalisms.

In addition to the need for developing tools oriented towards modeling objectives, there are also other engineering issues pertaining to HDCS-HMS. These include: the need for related tools pertaining to experimentation or testing (Macchi et al., 2003), comparisons such as in the area of evaluations of control policies (Brennan et Norrie, 2001), benchmarks (Frey et al., 2003)(Terzi et al., 2003), and human integration (Kotak et al., 2003).

## 2.9. Simulation

Experimentation and comparison of engineering solutions, such as different models for holonic controls, are important activities when seeking to determine the feasibility and advantages of each model (Brennan et Norrie, 2001). However, within the scope of HMS, direct experimental testing of global behaviors within the relevant physical

<sup>40</sup> Actually, in a general way, modeling engineering tools are needed to aid in the engineering process for all HMSs (Ollero et al., 2003)(Wyns, 1999).

manufacturing/control environment is extremely expensive and unrealistic. Thus, simulation is the only way out (Mařík, 2004).

Actually, the use of simulation is a recurrent theme when dealing with issues relating to manufacturing organizations, such as those regarding FMS (Flexible Manufacturing System)<sup>41</sup> (Bako et al., 1990)(Law, 1986). Moreover, in a concordant manner, an HMS solution can also be interpreted as a way to improve FMS technology.

A variety of research has been performed using simulations within the area of modern Manufacturing System (MS) such as FMS. More specific examples of this kind of research have been developed by LSIP<sup>42</sup> at CPGEI/CEFET-PR (Koscianski, 2000)(Künzle, 1990)(Rosinha, 2000)(Stadzisz, 1990)(Tacla et al., 1997). In fact, at LSIP, there exists a research branch related to design and simulation tools for MS<sup>43</sup> (Tacla et Tazza, 1995).

The use of appropriate tools is essential for developing satisfactory simulation and adequate design for MS. LSIP has considered a set of relevant requirements for design and simulation tools for modern MSs (Koscianski et al., 1999)(Rosinha et al., 2000)(Tacla et al., 1994)(Tacla et al., 1997). These requirements have been identified based on the specialized literature and on more over a decade of research pertaining to this subject (Carvalho, 2003)(Koscianski et al., 1999)(Tacla et Tazza, 1996)(Tazza et al., 1990).

It is appropriate to mention at this point that V. Mařík lists requirements for the development of agile manufacturing, including simulation as one of the items on the list. These requirements are in agreement with those requirements considered by LSIP for design and simulation tools for modern Manufacturing Systems (Mařík, 2004). These requirements from (Mařík, 2004) are summarized in an adapted way:

- An appropriate model of the manufacturing process to be controlled; this model must depict the relevant entities within the factory and their interfaces to the external world.
- A robust tool for simulating the manufacturing process, with a separate control process (as in real MSs); standard discrete event simulation tools like Arena, Grasp or Silk are not adequate and their usage for this purpose is limited.
- A suitable agent runtime environment for modeling the interaction of control agents.
- System integration strategies developed and implemented in the form of subsystem interfaces.
- Human-friendly interfaces for all system design and simulation phases.

The development of design and simulation tools that satisfy all of the above requirements has been considered as an appropriate research goal (Mařík, 2004). The relevance of this research goal is reinforced when it is considered the need of this tool type, with real low costs or even without costs, for research and educational centers. The availability of this tool type could allow, for example, customizable application for developing know-how in modern MS issues such as design, simulation, simulator, and related applications.

---

<sup>41</sup> A manufacturing system with a high degree of flexibility, usually consisting of automatically controlled NC machines and/or robots. Often very automated, including tool exchange, transport, and control (Bongaerts, 1998)

<sup>42</sup> *Laboratório de Sistemas Inteligentes de Produção* - Laboratory of Intelligent Production System.

<sup>43</sup> LSIP's members and associated researchers are interested in advanced techniques related to modern and complex production system (Chézaviel et al, 1999)(Lira et al., 2000)(Künzle et al., 1999)(Silva et al., 2000)(Schastai et al., 2004)(Stadzisz et Doll, 2002)(Stadzisz et Henrioud, 1998)(Tacla et Barthes, 2003)(Simão et al. 2005).

## 2.10. ANALYTICE II

The design of modern Manufacturing Systems (MSs), such as an FMS, is a subject encompassing several problems like group technology, equipment selection, scheduling, intelligent control, cost, and performance analysis. During the project phase, evaluation of different design solutions is required. This analysis can be accomplished by mathematical tools such as different as Markov chains, queuing networks, or stochastic Petri nets (Koscianski et al., 1999).

Alternatively, computational means, like discrete event simulation and other procedural tools, can accomplish the analysis. Using this approach, it is possible to code and test scheduling and control algorithms for which analytical techniques are not quite appropriate (Koscianski et al., 1999).

Although a number of simulators are available at the present time, it is difficult to find a tool that completely fulfills the requirements in a field of modern MS such as FMS (Koscianski et al., 1999). At LSIP, a simulation architecture was proposed featuring scalability and modularity, integrated as part of a computational tool for modern MS design and analysis. The resulting design and simulation tool for manufacturing systems is a prototype called ANALYTICE II.

ANALYTICE II is a research effort based on a previous one that generated the tool called ANALYTICE (Tacla et al., 1994)(Tacla et al., 1997)(Tacla et Tazza, 1995). ANALYTICE was a *time-oriented* simulation tool for FMS<sup>44</sup>. Likewise, ANALYTICE II was also developed focusing on FMS. However, ANALYTICE II primitives were conceived in terms of event-driven as well as in terms of object-orientation and functional independence of modules, implying high modularity (Rosinha et al., 2000).

Some ANALYTICE II properties are highlighted in Table 3. Among these properties is the graphical 3D representation of experiments, as illustrated in Figure 14, on the right side. The 3D representation is firstly to cognitively sensitize trainers/end-users (perhaps even users within industries) about the solution found in simulated designs, as well as to take into account physical interactions within the simulation, being a type of “realism feature” (Koscianski, 2000)(Koscianski et al., 1999).

**Table 3: Properties and particular features of ANALYTICE II<sup>45</sup>**

ANALYTICE II properties	
Event-driven	Resource Multi-model
Object Orientation	Splitting between resource emulation and control
Available Source Code	3D Graphical Animation
In-house solution	Scalability and High Modularity

Additionally, as another special feature of “realism” and also as a feature of modularity, ANALYTICE II has a clear separation between resource (e.g. automatic equipment) emulation and manufacturing control execution. This separation is imposed by a virtual network, working asynchronously, in the simulator kernel.

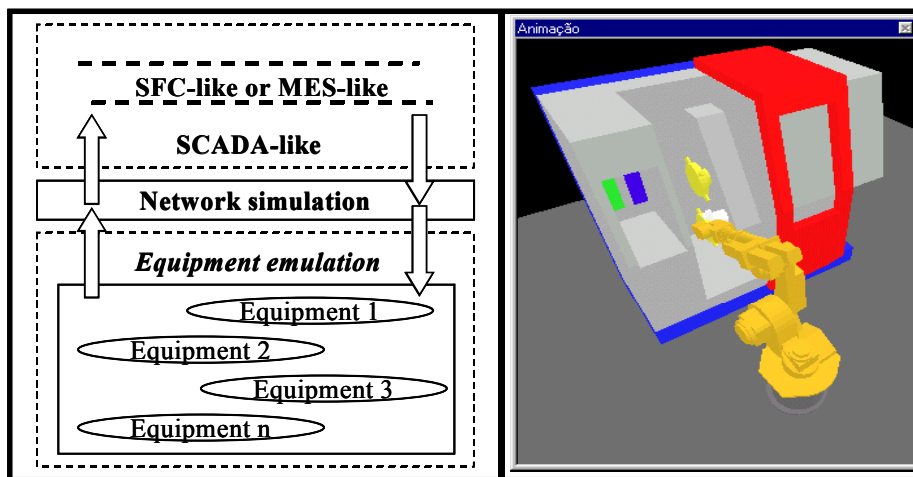
<sup>44</sup> These first research efforts have generated a set of publications; see for example (Künzle, 1990)(Sautter, 1991)(Setti, 1998)(Souza, 1991)(Stadzisz, 1990)(Stadzisz et al, 1991)(Stadzisz et Tazza, 1992)(Tacla, 1993) (Tacla et Tazza, 1996)(Tazza et al., 1990)(Tazza et al. 1992).

<sup>45</sup> These properties and features are subject of explanation in the next chapter.

Signals to resources, from control entities, are sent by means of this virtual network and signals about resource states are recovered from this virtual network to potentially update monitoring entities, as illustrated in Figure 14, on the left-side (Koscianski et al., 1999).

Due to the virtual network, it is possible to simulate entities like SCADA and SFC/MES in the manufacturing control side of ANALYTICE II without direct interaction with the resources (Simão et al. I, 2001)(Simão et al. II, 2001)(Simão et al. III, 2001). In some manner, these simulations can be considered realistic because networks also separate decision-making entities and resources in real applications.

The separation of emulated and control parts enables creation and/or testing of software solutions for industrial applications (such as scheduling, control, fault supervisor, etc), potentially applicable to real cases (Carvalho, 2003)(Rosinha, 2000)(Rosinha et al., 2000)(Silva, 2001)(Simão, 2001)(Simão et al. I, 2001). This means that ANALYTICE II allows the study of software engineering applications related to manufacturing systems (Simão et al. II, 2001)(Simão et al. III, 2001)(Simão et al., 2002)(Stadzisz et al., 2003).



**Figure 14: ANALYTICE II structure and its graphics animator module.**

ANALYTICE II makes possible the study of not only software engineering, but of system engineering as well. In fact, even system engineering related to holonic applications can be studied using ANALYTICE II once a solution to compose hard-holons, within it, has been developed. In this sense, the hard-holons (Base-HLs) could be created in a realistic way meaning that a network layer would separate their hard and soft parts.

The realistically simulated Base-HLs in ANALYTICE II would allow experiments with respect to holonic control solutions. In fact, it would effectively stimulate the development of holonic control solution once a fitting holonification of ANALYTICE II would also imply this type of control solution<sup>46</sup>.

Actually, the ANALYTICE II features have allowed considerations about its applications with respect to HMS issues and, moreover, they have even allowed considerations about its holonification to be effectively a design and simulation tool for HMS. Furthermore, the ANALYTICE II holonification would be in agreement with the need for appropriate holonic design and simulation tools as presented in the previous section.

<sup>46</sup> This solution should not be a specific solution, but rather, a generic one, applicable to a variety of simulation scenarios, and open to the interaction of other decision-making systems. Still, the solution should be easily applicable and open to correctness features. Moreover, it would be profitable if this same control solution also allowed heterarchical and hierarchical control implementation, enabling paradigm comparison. In this sense, it is important for the control solution to allow both process-driven and product-driven controls.

ANALYTICE II is truthfully an alternative tool and particular solution of LSIP, normally open to associated institutions. The holonification of ANALYTICE II could enable a platform to design, test, validate, and compare solutions for HMSs, aiming at better understanding HMSs, generating know-how about HMS and even achieving alternative and better solutions for HMS.

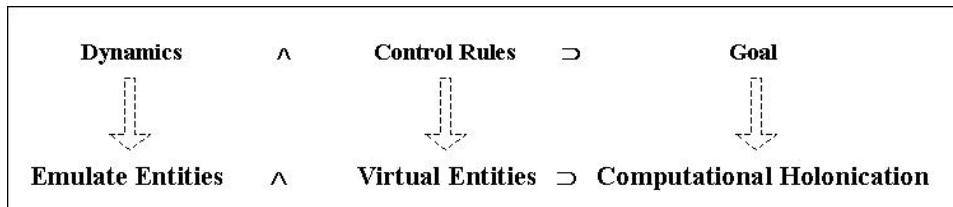
ANALYTICE II is also and mainly an educational resource and project developed with the intention of putting students within (similar) industrial situations through the development and use of a realistic simulation tool (Koscianski et al., 1999)(Tacla et Tazza, 1995). Consequently, ANALYTICE II must be synchronized with current research and future industrial tendencies.

There are several reasons to improve ANALYTICE II with respect to agile manufacturing, or specifically to holonic manufacturing. In this section, some important motivations for ANALYTICE II holonification have been presented.

### 2.11. Holonic Simulator

The **first objective** in this doctoral thesis is the holonification of ANALYTICE II. A first step in the ANALYTICE II holonification is the integration of emulated physical entities, from simulated manufacturing systems, at a high-level of control for achieving Base-HLs. This integration can be achieved by the *computational holonification* of these entities, such as achieve *virtual entities* equivalents to them.

The *virtual entities* would be a *software layer* to, at least, express states and receive service requirements related to emulated entities, encapsulating the concerned monitoring and the service requesting. In truth, the computational holonification of an emulated entity means generating a type of automatization of the entity. Therefore, the computational holonification of emulated entities can be expressed as an interpretation of the Fusaoka's equation, such as presented in the Equation 2.



**Equation 2: Holonification of Emulated Entities**

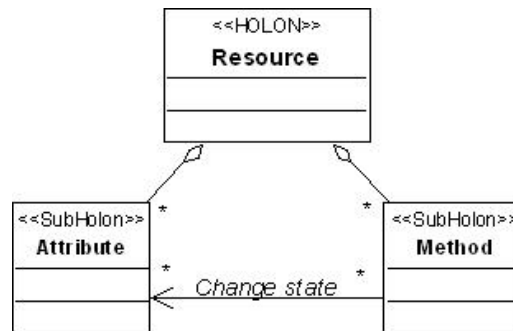
As a first step, the computational holonification can be focused on the emulated resources, aiming at Resource-HL design. In ANALYTICE II, the Resource-HL would be composed of the *emulated-resource* and a respective *computational agent*, i.e. a *virtual resource*. For the Resource-HL composition, the virtual resource would be located in the control layer of ANALYTICE II.

In short, the virtual resource should be primarily responsible for: (a) getting information from the virtual network relative to its respective resource; and (b) requesting services by sending of information to the resource, using the virtual network as well (Simão et al. I, 2001). In fact, the Resource-HL set would comprise parts of the control responsibility in HMS, specifically high-level monitoring and command (i.e. a kind of service requesting)<sup>47</sup>.

<sup>47</sup>The Resource-HLs, in some manner, substitute or improve the services concerned with SCADA layer in a set of decoupled entities.

Each Resource-HL could express the monitored states by means of subholons called *Attributes*, and it could receive service requests (from other entities, such as control entities) by means of subholons called *Methods* (Simão et al. III, 2001). In this way, ANALYTICE II would present Resource-HLs working in an external and (computationally) homogenized manner (Simão et Stadzisz, 2002).

The aggregation of Attributes and Methods by Resource-HLs is modeled in a holonic-class diagram in Figure 15. An Attribute indicating the “status” and a Method for requesting “the process of a part” in a Resource-HL related to a Machine would be specific examples. Attributes and Methods could have still other functions, such as keeping track of states monitored or requests carried out.



**Figure 15: Attributes and Methods for Resource-HLs.**

The products should also be taken into account in the Base-HL composition. In this sense, just as in the case of resource holonification, an appropriate computational agent could be proposed for each product, i.e. a virtual product, aiming at Smart-Product-HLs in ANALYTICE II (Gouyon et al., 2004)(Simão et al. 2005).

Once the holons are *homogenized* or *inserted* into a same universe (i.e. computational universe), the next step should be to create a holonic control solution, open to correctness features, for reaching a proper ANALYTICE II holonification<sup>48</sup>.

The solution to HDCS should be applicable to control the collaborations of Base-HLs like those simulated in the homogenized-ANALYTICE II. Effectively, this solution would have certain independence of the simulator once an instanced HDCS would not be *aware* about the *simulation-nature* of the Base-HLs. Nevertheless, this control *independent* solution applied in the homogenized-ANALYTICE II would allow finding a holonified-ANALYTICE II.

## 2.12. Control Solution

The control solution investigated within the current project is, in reality, achieved by finding a software architecture for control because all holons are *reachable* at computational or virtual *world*. The software control solution should consider a set of features, such as correctness and holonic features, aiming at an appropriate holonification of ANALYTICE II as well as a proper control architecture tested in ANALYTICE II.

During the process of developing this the doctoral thesis, a particular interpretation of correctness and holonic features was developed, which could be considered in control architectures (Simão et al. III, 2001)(Simão et Stadzisz, 2002)(Simão et al. II, 2003)(Simão et

<sup>48</sup> The proposed holonification of ANALYTICE II means generating a solution to emulated-entity holonification, as well as an integrated HDCS solution to deal with the (process-driven or product-driven) control question. The current research project models and designs/simulates HMSs with Base-HLs and HDCSs. These experiments are relative to *processes cells* (machining cell and flexible assembling cell) and relative to an *area-like* (production environment constituted by *process cells* inclusive).



al. 2005). These features are summarized in Table 4, and they should be taken into account in the elaboration of the foreseen HDCS architecture.

**Table 4: An interpretation of correctness and holonic features**

Correctness Features	Holonic Features
Trade-off between reactivity and determinism	Systemic integrated or effectively open for integration
Mechanism to avoid deadlock	Distributed or open to distribution, i.e. decoupled entities
Mechanism to conflict identification and resolution	Agile supporting, i.e. trade-off between hierarchism and heterarchism.
Some compatibility with synthesis formalism, e.g. Petri nets	Process-driven supporting
Low computational complexity	Product-driven supporting.

The solution should also consider system-engineering issues, such as the trade-off between generality and applicability. Actually, the control architecture is not intended for just a single control, but rather for use with the large set of production scenarios made possible by means of a diversity of Base-HLs that are testable in ANALYTICE II.

In fact, the control architecture should be concomitantly generic and applicable, i.e. as a meta-model, to the agile control over the collaborations of Base-HLs, like those simulated in ANALYTICE II. In some way, the meta-model being examined can be also viewed as an architectural pattern for a particular kind of control software, i.e. the HDCS (Stadzisz et al., 2003).

Additionally, the solution should present mechanisms to facilitate the control construction/instantiation and human friendly interfaces. In brief, the meta-model should be an engineering tool to help with the construction of the HDCS. The meta-model should allow constructing HDCS as specified in the Equation 3.

$$\text{Hard-Holons} \wedge \text{Control System} \supset \text{Correct and Holonic Features}$$

**Equation 3: Correct and Holonic Control Systems.**

Actually, the **main objective** of this thesis is to search for a special control solution to HDCS that must serve as an engineering tool. The term “special” means that it takes into account together those objectives described above and that it has as a principle the *homogenization* or *standardization* of concerned entities into a same universe H (i.e. holons).

The control solution tested in ANALYTICE II would implicitly achieve a proposition to conceptual solution to holonic control due to the simulator realistic feature. Effectively, this thesis aims to consider the proposed meta-model as a conceptual HDCS solution to real HMS, because the main difference between simulated and real holarchies foreseen would be the Base-HL implementation.

Engineering tools, such as meta-models, help to save time, financial resources, technical resources, and so forth. These benefits could be, for example, due to pre-designed and pre-tested solution. The development of engineering tools related to HMS is a subject being researched within the community related to IMS (Intelligent Manufacturing System).

*Morel et al.* begin the process of establishing a broader HMESE (Holonic MES Engineering) framework, as summarized in Figure 16 (Morel et al., 2003). The control solution that is the research goal of this doctoral thesis could be integrated as contributions at the Conceptual level, or even at the Organizational level, of this framework.

System models Abstraction Levels	Communication	Information	Processing	
			Data-driven	Event-driven
Conceptual	HMS - like			
Organisational	Unified (Patterned) Modelling LANGUAGES, e.g. UML-like			
Operational	IEC/ISO 62264 based on UML notation			
	IEC 61499	Relational Model	IEC 61499 based on UML notation	
Technical	MES Automation technology			

**Figure 16: HMESE framework (Morel et al., 2003).**

The intention of this framework is to integrate standardized MES technologies by unifying modeling languages in order to facilitate the move from the Integration in Manufacturing<sup>49</sup> (IiM) paradigm to the IMS paradigm for manufacturing enterprise control and management integration issues. This framework is a research initiative related to the CRAN (*Centre de Recherche en Automatique de Nancy*) affiliated with the UHP (*Université Henry Poincaré*).

The carrying out of the objectives presented in these two last sections would be contributions concerned with correlated research subjects from CPGEI/CEFET-PR and CRAN-UHP (both schools with doctoral programs), since these are the respective institutions to which the author of the current doctoral thesis is affiliated.

<sup>49</sup> Integration in Manufacturing (IiM) is the first 'systemic' paradigm to organize Humans and Machines as a whole system, not only at the field level, but also at the management and corporate level to produce an Integrated Enterprise System (Morel et al., 2003).

## **Chapter 3**

---

Holons in ANALYTICE II



## Chapter 3 : Holons in ANALYTICE II

### 3.1. Introduction

In this chapter, firstly, concerns about simulation are briefly explained. After that, ANALYTICE II structure is presented, including the interaction between control and emulated entities. A designed and simulated process cell in ANALYTICE II is also considered as an example.

This concise presentation of ANALYTICE II is based on the works published by LSIP researchers (Koscianski et al., 1999)(Rosinha, 2000) and on the use and development of this tool<sup>50</sup> (Simão et al. I, 2001)(Simão et at. II, 2001).

The computational holonification of resources in ANALYTICE II is also taken into account in this chapter. This holonification is highlighted as well as the homogenization of Resource-HLs. The homogenization is performed in order to facilitate the elaboration of holonic solution to control<sup>51</sup>.

Specifically, the holonification of resources related to the presented process-cell is proposed. This holonification takes into account homogenization based on self-similarity principles. Furthermore, the resulting self-similar holons are modeled in UML. This holonification serves as basis to propose and generalize a holonic model of Resource-HLs<sup>52</sup>.

### 3.2. Simulation Issues

The set of constraints and requirements that must be accomplished in the design of a modern Manufacturing System (MS), e.g. the need for an MS considering systemic integration and flexibility exploitation, is considerably large and complex. Quantitative and qualitative data, as well as subjective factors and some uncertainty embedded in human decisions, must all be taken into consideration (Koscianski et al., 1999).

Minor design changes can lead to significant variation in the results. Modifying scheduling and control decisions, for example, may change at the same time flexibility of routing, fault-recovery features, production volumes, and production due-dates.

There is no deterministic algorithm or method for designing a modern Manufacturing System (MS) automatically. This is generally an activity of increasing refinement, relying upon expertise supported by specialized tools. Moreover, changes within an existing MS often present problems similar to those faced when designing a new MS. In the both cases, time and money are serious concerns (Koscianski et al., 1999).

It should be clear that modern MS design is a critical area due to the risks and the large investments involved (Chao et Wang, 1995). In this sense, there are a number of tools that can help engineers, such as analytical and computational methods.

---

<sup>50</sup> The particular details of ANALYTICE II can be found in the works of Koscianski (Koscianski, 2000), Rosinha (Rosinha, 2000), Simão (Simão, 2001), and Carvalho (Carvalho, 2003).

<sup>51</sup> The Smart-Product-HLs in ANALYTICE II are highlighted in the chapter 6.

<sup>52</sup> If the reader is already familiar with ANALYTICE II structure or would like to read the technical (and important) details at another moment, he is advised to continue this reading in the section 3.6.

Generally, a good analytical model is difficult to build, complex to deal with (Viswanadham et al., 1995) and does not always capture all modern MS logic and functionality<sup>53</sup> (Zeigler et al, 1996). Computational simulation tools can, to some extent, bypass these problems and, additionally, prioritize a user-friendly interface (Breugnot et al., 1991), hide specialized knowledge within the software (Breugnot et al., 1990), and be used within training programs (Law et Kelton, 1991).

Specifically, in the case of MS analysis, simulators allow control and scheduled testing, and can provide a rich set of quantitative data, since virtually any aspect of the plant may be observed (Koscianski et al., 1999). According to the literature, there is a diversity of simulation tools in the field of MS analysis (Banks, 1996)(Jeng, 1995)(Tazza et al. 1992)(Zeigler et al, 1996), and a variety of approaches like numerical simulation or black-box models, distributed architectures, process-oriented, and event-oriented (Schriber et Brunner, 1996).

Regardless of its importance, simulation is only one of the tools used in the design and analysis of modern MSs. Other engineering tools are also required, and these should, ideally, be integrated, allowing information sharing during the different project phases, following a systemic approach (Koscianski et al., 1999).

The simulation environment called ANALYTICE II presented in this chapter, besides being a unique and in-house tool (as described in last chapter), is a component of an MS engineering toolkit under development at the LSIP/CPGEI/CEFET-PR. This software will integrate a set of MS design tools, providing modules for MS projects and analysis through simulation techniques (Koscianski et al., 1999).

### 3.3. Simulation Environment

The simulation environment of ANALYTICE II encompasses a set of modules for carrying out the tasks that emulate the MS operation. Figure 17 shows a conceptual block diagram of this environment, including each module described in this section. ANALYTICE II was designed to apply a discrete-event simulation approach, and it is based on results of previously developed works (Tacla et al., 1997)(Tazza et al. 1992).

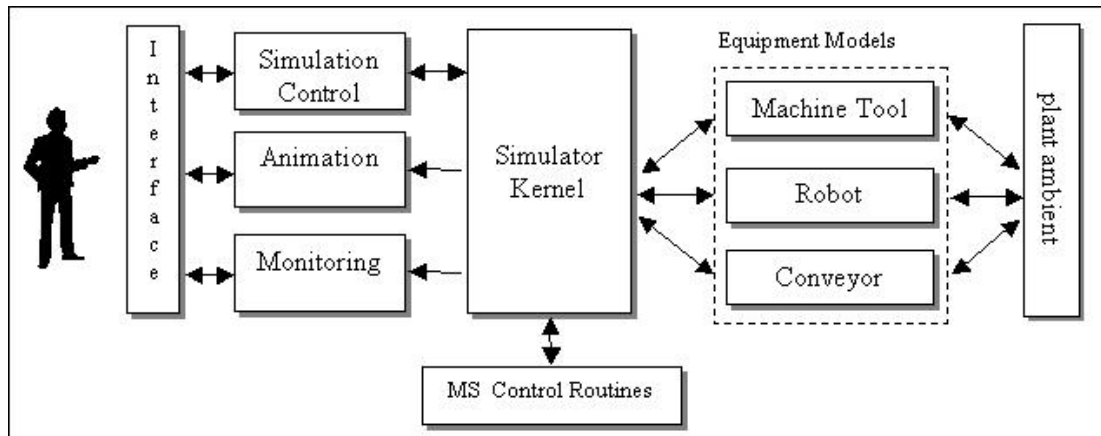


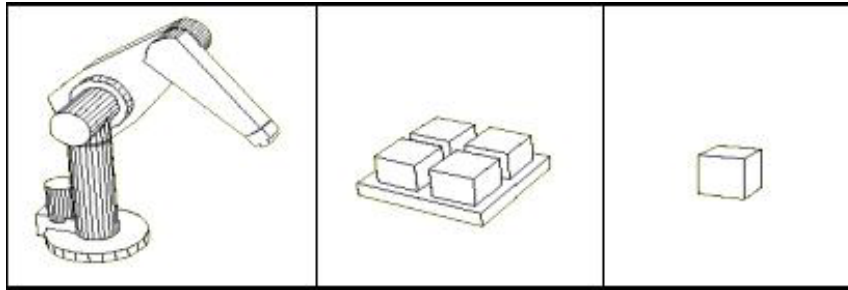
Figure 17: Conceptual Block Diagram of the ANALYTICE II Simulation Environment.

<sup>53</sup> According to Solot and Van Vliet, analytical methods can be divided into two categories, queuing networks and Petri nets (Solot et Van Vliet, 1994). They cover a broad range of optimization issues: processing capacity, buffer capacity, facility layout and workload allocation among others. Moreover, mathematical techniques, concerned to analytical methods, can be used ranging from simple linear programming to models as complex as multidimensional linear hyperbolic partial differential equations (Viswanadham et al., 1995). In spite of the drawbacks of analytical methods, there are advantages to using them, such as speed and direct calculation of minimum and maximum values, rather than obtaining averages.

The *simulation environment* of ANALYTICE II takes as input equipment, workstations, or plant models. The equipment model is highlighted because workstations and plant models are composed of a set of equipment. These models are constructed using other design environments integrated with the simulation environment.

An *equipment model* is basically defined by means of a Behaviour Model (BM) and a geometrical and kinematics model. The BM may contain the equipment processes (e.g., ‘moving arm’ and ‘milling part’), state variables, processes related to a dedicated controller, and effectively its interfaces with the simulator environment, which should correspond to the real interfaces of the real equipment.

Each piece of equipment is an independent entity that communicates with the simulator *kernel* using an event-based mechanism<sup>54</sup>. Examples of industrial equipment that may be simulated are machine tools, robots, AGVs, and storage systems. These types of equipment can virtually operate on simulated parts and/or pallets, depending on their specifications. Equipment, pallets, and parts are the simulator primitives (Figure 18).



**Figure 18: Simulator primitives: equipment, pallets, and parts.**

The basic responsibilities of the *kernel* are to manage a time-ordered list of events, update the simulation clock, and interact with and coordinate the work of the other modules. Emulated automatic equipment can communicate with control routines and, potentially, with other equipment through the exchange of messages and control signals, using the simulator kernel as a mailer. Therefore, the *kernel* is considered a *virtual network*.

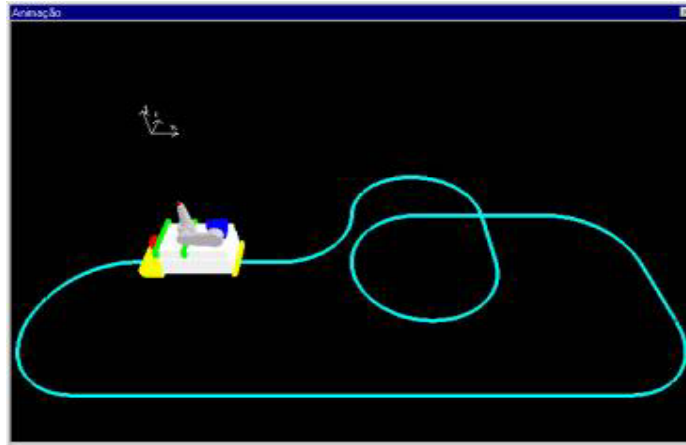
The *plant ambient* module handles some simulation occurrences that are external to the equipment. An example is the grasping and releasing of parts. Another example is the queries from AGVs about their paths. Thus, the *plant ambient* maintains information about the plant layout and the physical positioning of *parts* and *pallets*.

The *monitoring* module improves user interaction by granting visibility to internal variables of each piece of emulated equipment. It allows the creation of reports (if appropriate interfaces are developed), provides graphical output, and operates with a sampling rate. This module could gather data for later simulation analysis if a module for the storage of data were defined.

The *animation* module presents a 3D graphical animation of the MS execution and is intended to provide an informal validation of the model, allowing for the localization of possible modeling mistakes (Breugnot et al., 1991). Figure 19 presents, as an example, the 3D graphical animation of an emulated AGV in ANALYTICE II.

The fact that animation should be continuous while the simulation is discrete-event driven requires the slicing of the animation. This slicing was accomplished with the aid of animation pseudo-events, called *Time-Slices (TSs)*, which cause the animator to periodically redraw its output.

<sup>54</sup> In the ANALYTICE II implementation, each BM runs in a specific thread.



**Figure 19: 3D graphical animation of an AGV in ANALYTICE II.**

The *equipment* export an interface that is used by both monitoring and animation modules to consult the value of its internal variables. In fact, during the construction of each equipment model, the designer specifies which internal variables can be observed by the monitoring and which variables are related to kinematics actions of the equipment.

These variables constitute the equipment's interface with the monitoring and animation modules. Monitoring and animation could be configured, or even turned off. To synchronize simulation with physical time and improve animation realism, there is a time delay mechanism performed by the kernel.

The *MS control* routines communicate, in an asynchronous way, with the kernel using a specific interface. It must be a discrete-event dynamic system control, and it uses the same event-based mechanism as the equipment models.

The *user interface* has two purposes. First, it allows the configuration of simulation parameters. Second, it provides a means to manipulate the state of equipment and control during the simulation process<sup>55</sup>.

### 3.4. Simulation Architecture

The developed simulation architecture can simulate *discrete event dynamic systems*, focusing on MS features. Such systems have a finite set of states whose changes occur only at specific time instants, namely events (Law et Kelton, 1991).

Figure 20 outlines the dynamic model of the main elements in the simulation architecture by using a predicate-transition Petri net notation. The simulation starts with the firing of transition  $T_1$ , which sets simulation clock,  $t_{sim}$ , and adds a Time Slice (TS).

After the  $T_1$  firing, the net enters in a cyclical behavior (path) that corresponds to the execution of the time-synchronization mechanism ( $T_2$ ), update of  $t_{sim}$  ( $T_3$ ), and *event treatment*.

The *event treatment* is the control execution ( $T_4$ ,  $T_6$ ,  $T_7$ ) or equipment treatment ( $T_5$ ), implying an equipment activation ( $T_8$ ,  $T_{10}$ ) or equipment geometric update ( $T_9$ ,  $T_{11}$ ) followed by animation ( $T_{12}$ ). Finally, after the *event treatment*, the monitoring ( $T_{14}$ ) is carried out.

<sup>55</sup> It is possible, for example, to cause the breakdown of a machine (if the designed equipment allows this) and verify if scheduling and routing algorithms react as planned.



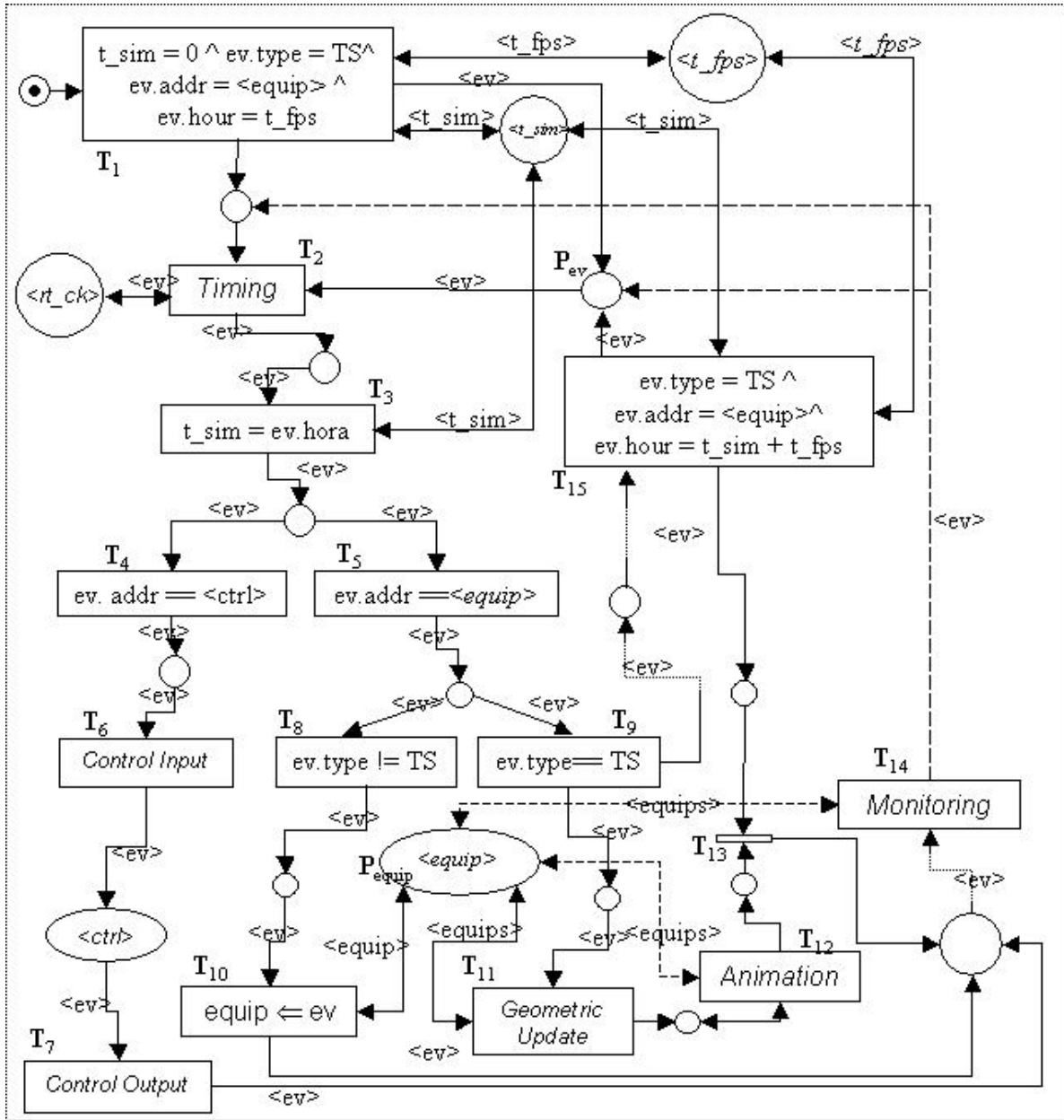


Figure 20: Petri Net of the Simulator Kernel.

Events can be created in different ways:

- A piece of *equipment* schedules an event when its state has to change at some future instant, such as at the end of a process, e.g., at the end of the process of milling of a part for a limited amount of time;
- The *MS control* issues a command to the piece of *equipment*, which is sent in the form of an event;
- A piece of *equipment* sends a message or control signal to the *MS control* or (potentially) to another piece of *equipment*;
- The *plant ambient* deals with an interaction and sends an event to inform a *piece of equipment* about its occurrence (e.g., a conveyor receives a part from a robot).

A typical event is processed as follows. A robot (a piece of equipment) receives an event  $E_1$  carrying a command instructing it to move its arm, with a given speed and final position. The

robot calculates the amount of time that the task will take and changes its state to ‘moving arm’. After that, the robot schedules an event  $E_2$  in the kernel for the time of completion of the process ‘moving arm’.

All this action corresponds to transition  $T_{10}$  in the Petri net (Figure 20). On the occurrence of event  $E_2$ , the equipment should: finish the ‘moving arm’ process, update internal variables (e.g., joints’ angles), and change its state to ‘stopped’.

The *simulator kernel* (presented as a module in Figure 17, with its main dynamics presented in Figure 20) arranges in a proper order the activities of the following modules: animation, monitoring, and MS control. It activates the equipment models as events occur ( $T_5$ ), and is also responsible for simulation timing control ( $T_2$  and  $T_3$ ).

The main kernel components are the simulation clock, a time-ordered event list corresponding to place  $P_{ev}$  in Figure 20, and a list with references (links) to the simulated equipment corresponding to place  $P_{equip}$ . The kernel interface provides functions to schedule new events and to cancel a specific event, or all events, scheduled for some piece of equipment<sup>56</sup>.

The MS control also receives stimulus from the kernel to execute when an event is addressed to this MS control (after the firing of transition  $T_6$ ). During its execution, new events may be generated (commands to equipment,  $T_7$ ) and sent to the kernel event list (traversing the monitoring,  $T_{14}$ )<sup>57</sup>.

The ‘next-event time advance’ (Law et Kelton, 1991) was the adopted strategy for clock handling. The algorithm consists in removing the first node (with an event) of the event list, updating the simulation clock to this event time, and sending the event to its target.

In order to obtain a fixed frame rate for animation an artificial event called *time-slice (TS)* was created. On the processing end of a TS, the kernel allows the animation module to draw a new frame. This new frame reflects the state of the system at that instant.

A new TS is scheduled by the simulator kernel immediately after a TS has been processed (transition  $T_{15}$ ). The interval between consecutive TS events could be configured by the user. The monitoring module is also allowed to run when a TS event occurs.

The time-slice events have a special treatment. When a TS event is the next-in-line for processing (at the head of the list), it is removed from the list and sent to every piece of equipment being simulated (transition  $T_9$  and  $T_{11}$  of the Petri net).

Each piece of equipment has specific handlers in which the variables related to geometric information (e.g. joint angle) are appropriately updated, if updating is required (transition  $T_{11}$ ). Every other event is sent to only one Behaviour Model (BM) that is the target of the event (transition  $T_{10}$ ).

In the case of the ‘moving arm’ process, used in the example above, the robot animation handler should update the joint angle each time a TS is received. The movement speed and the time difference between adjacent TSs are the two pieces of information required to perform this calculation.

When the animation module receives the TS notification from the kernel (transition  $T_{12}$ ) it reads from each piece of equipment the variables associated with geometric information, and

<sup>56</sup> This last feature could be used, for example, in the processing of a breakdown of a piece of equipment.

<sup>57</sup> Actually, the kernel has the role of a virtual network between the control and the equipment within which events (information packets) are exchanged between them

uses this data to update the screen output<sup>58</sup>. Monitoring operates in the same way and corresponds to the firing of transition  $T_{14}$ .

One of the objectives of this research project is to allow the simulation speed to vary from accelerated to decelerated (slow-motion). The user, through a ratio like 1:1 or 2:1, would specify the relation between simulation time and physical time.

To implement the speed control, there is a delaying mechanism that uses three kinds of information: i) the computer real time clock, RT\_CK; ii) the difference between RT\_CK and the next event time; and iii) the above-mentioned ratio.

By a simple computation, the amount of time the simulator must wait to synchronize can be determined (this corresponds to the Petri net transition  $T_2$ ). By turning off TS events, the simulation could run at the maximum speed supported by the computer.

In the Petri net of Figure 20 the following predicates appear:

- t\_fps: time difference between two adjacent time-slice events;
- rt\_ck: real time clock (the computer system clock);
- t\_sim: simulation clock (maintained by the kernel);
- ev: an event on the event list<sup>59</sup>.

### 3.5. Functional Prototype

The fundamentals behind the construction of ANALYTICE II, a simulator for modern Manufacturing Systems (MS), have been briefly discussed in the previous sections. The presented simulation environment and its kernel architecture were also outlined.

The main features of that architecture are:

- Usability – the Behaviour Model (BM) of equipment is coded in such a way as to hide the simulator internals from the designer.
- Flexibility – by using a specified interface, new machine-tools and software components, such as schedulers or knowledge-based controllers, can be incorporated.
- Modularity – the support for replaceable units, e.g. *animation module*.
- Systemic – the simulator is being developed as a part of a larger environment that ideally integrates design and analysis tools.

At present, ANALYTICE II has been developed as a functional prototype, which allows demonstration of the validity of the simulation architecture, a major concern during the conceptual project of this type of tool. The prototype provides the features described in the previous section.

The prototype provides, for example, the 3D graphical interface, as well as the user interface for interacting with the equipment (Figure 21). Also integrated environments to create new emulated equipment are developed in a partially user-friendly way. Also, a library of

---

<sup>58</sup> Additionally, there is the *equipment* access in the *plant ambient* that was not outlined in Figure 20. It is done in the following way: a piece of *equipment* informs the ambient about an action such as a part release. The ambient checks the facility layout for the presence of some piece of *equipment* or *pallet* within the suitable proximities of the release coordinates. If a match is found, an event is sent through the *kernel* to the target. Also, in the context of the *plant ambient*, path information is sent directly from the *ambient* to the requesting *vehicle*, without the aid of the *kernel*.

<sup>59</sup> The firing of transition  $T_2$  removes the first event from  $P_{ev}$ .

equipment is provided, making available some machining equipment, robots, stores, and AGVs.

This prototype was developed and described by Koscianski (Koscianski, 2000) and Rosinha (Rosinha, 2000). The research described there was directly coordinated and advised by Prof. P. C. Stadzisz and Prof. L. A. Künzle. A second effort concerning ANALYTICE II was an initial architecture control proposed by Simão (Simão, 2001). The results and conclusions from this research formed the basis for writing the current doctoral thesis.

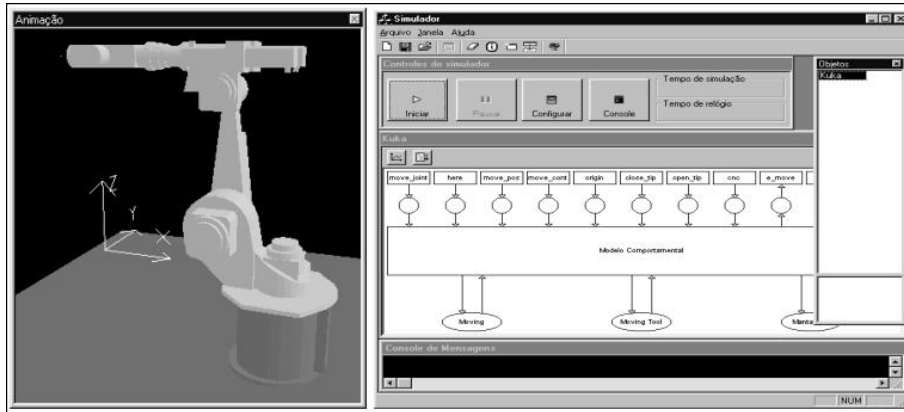


Figure 21: 3D graphical interface and user interface in ANALYTICE II.

### 3.6. A Process-Cell in ANALYTICE II

An example of a production system designed and simulated in ANALYTICE II is the cell presented in Figure 22. The process-cell creation was established upon realistic equipment projected for use within ANALYTICE II and put into its equipment library.

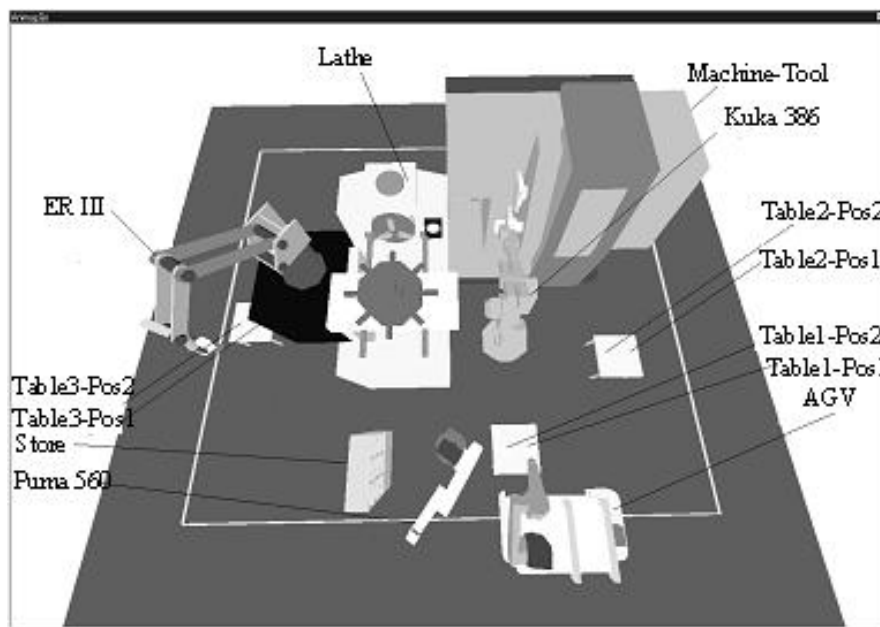


Figure 22: Virtual Machining Cell simulated using ANALYTICE II<sup>60</sup>.

In this simulated cell, each piece of emulated-equipment can be classified by its function. There is equipment for machining (a lathe and a machine-tool), transport of parts (a Puma

<sup>60</sup> Figure extracted from ANALYTICE II graphics animator module.

560, a Kuka 386, an ER III, and an AGV with end-effector), and storage (a 3x3 store and three tables with two positions), as presented in Figure 22<sup>61</sup>.

Initially, for this process-cell, the production of two products, i.e. products V1 and V2, could be considered. The virtual product V1 could have as its process plan the first one defined in Figure 23, while the virtual product V2 could have as its process plan the second one defined in Figure 23.

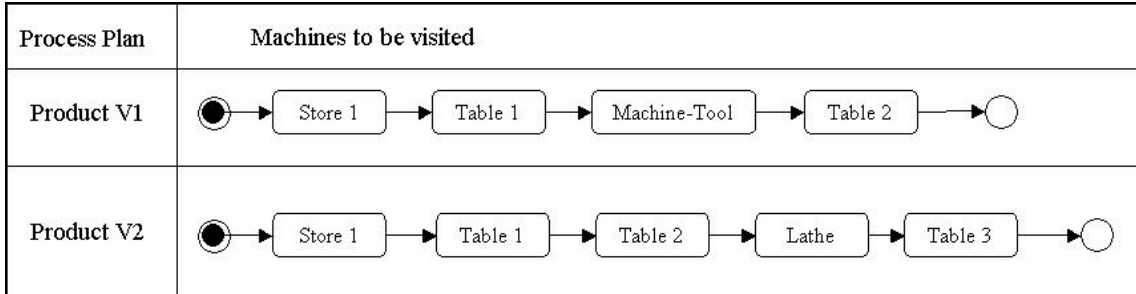


Figure 23: Process plans of virtual products.

### 3.7. Computational Holonification

ANALYTICE II architecture allows the composition of discrete event control (DEC) in order to coordinate emulated resources such as those presented in the previous section. This DEC can be thought of in different ways. For instance, it could be organized as a hierarchical control, a classical solution in manufacturing systems (Rosinha, 2000).

The intention of this research project is to create a control architecture, tested over ANALYTICE II that allows instantiating more than hierarchical or even heterarchical controls. The idea is that the solution permits constructing agile control, focusing on holonic control. In this sense, a first issue in holonic control can be related to the heterogeneity of subsystems, its negative implications for control, and the overcoming of the implied problems by means of holons.

In real manufacturing systems there exists a clear separation between automatics and informatics subsystems involved in production issues. The interaction of these heterogeneous systems happens by means of networks. A similar context is found in ANALYTICE II, where emulated elements of plant and high-level control entities are separated by means of a virtual network.

The constituents of manufacturing systems are not only heterogeneous in relation to their natures (e.g. automatics or informatics entities) but also heterogeneous in their own constitutions (e.g. they have their own interfaces, languages, and specificities). Heterogeneity is an inappropriate feature for constructing a holonic control because it implies one more complicating factor, i.e. the control must *understand* the specificities of each concerned entity and *promote* the communication of entities with distinct natures.

The holon concept, previously discussed, permits the conception of all entities within manufacturing scope as if they possessed the same nature, i.e. as holons. Therefore, it conceptually creates a certain degree of homogenization for MS entities. A possible way to

<sup>61</sup> The Lathe and Machine-Tool function is to process parts following the specification of NC programs. The Tables each have two buffer positions (pos1 and pos2), while the Store has nine (9) buffer positions in the form of a 3x3 matrix. The PUMA 560 robot transports parts from Store to Table1 (and vice-versa), KUKA 386 transport parts from Table 1 to Machine-Tool (and vice-versa) or from Machine-Tool to Table2 (and vice-versa). ER III transports parts from Table3 to Lathe (or vice-versa). And the AGV (with end-effector) transports parts from Table3 to Table1 (and vice-versa).

interpret the holon concept within a practical application is to consider the MS entity as integrated at the informatics level.

This technical interpretation of the holon concept can be achieved by means of a *computational holonification*, using integrator agents as a technical artifice. Computational holonification means that each physical entity (e.g. a product or a piece of equipment) has its own respective integrator computational agent.

By means of the computational holonification, holons would carry out their interactions (e.g. requesting services from, or enabling services for, other holons) within the computational scope, independently of the base nature of each one of the holons.

One specific example of computational holonification is resource holonification. Resource holonification allows holonic control over resource cooperation within the virtual/computational world. In the next section, a way to construct Resource-HLs within ANALYTICE II by means of computational holonification is discussed.

### 3.8. Resource Holonification

The computational holonification of a resource can be accomplished by means of an equivalent computational entity at the informatics level, i.e. a *virtual resource*. At minimum, this virtual resource would consist in a reactive agent enabling monitored data about the respective resource and receiving service requirements for the respective resource. Figure 24 depicts, for example, a resource set computationally holonified by means of virtual resources, as well as computational control entities interacting with these virtual resource at a high-level.

In ANALYTICE II, a virtual resource can be created on the control side for each resource on the emulation side, with the communication between them carried out via virtual network. The communication between the pair of physical and virtual resources is carried out within the scope of the Resource-HL that encapsulates this pair, thereby hiding the pair communication from other holons, such as control holons<sup>62</sup>.

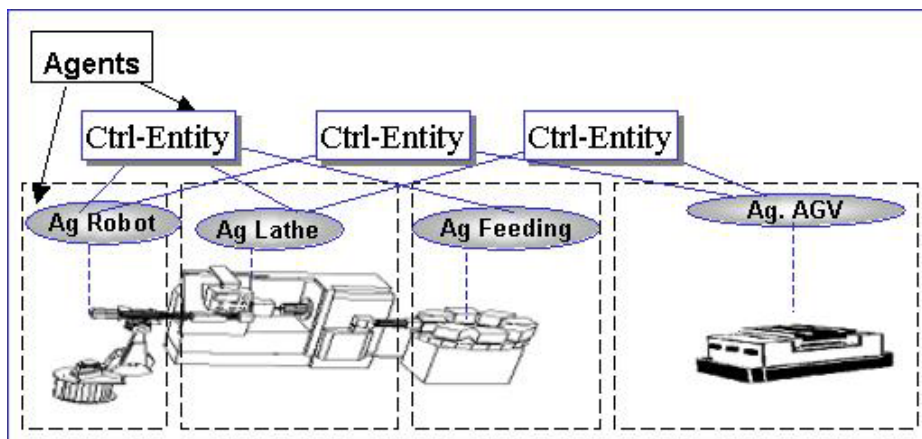


Figure 24: Virtual resource (agents) to represent and treat each piece of equipment.

The physical resource and virtual resource must be as cohesive as possible within the composition of the Resource-HL. This cohesion means that the resource-agent must be aware (in real time) of the physical resource states and that service requirements must be sent as

<sup>62</sup> In a real case, a Resource-HL for machining might contain a machine tool, sensing and actuation devices, a local controller, a network connection, and one or more coordinating software agents located either locally or on the network (McFarlane et al., 2002). Likewise, a Resource-HL for material handling might contain a robot, sensing and actuation devices, a local controller, a network connection, and one or more coordinating software agents located either locally or on the network (McFarlane et al. 2003).

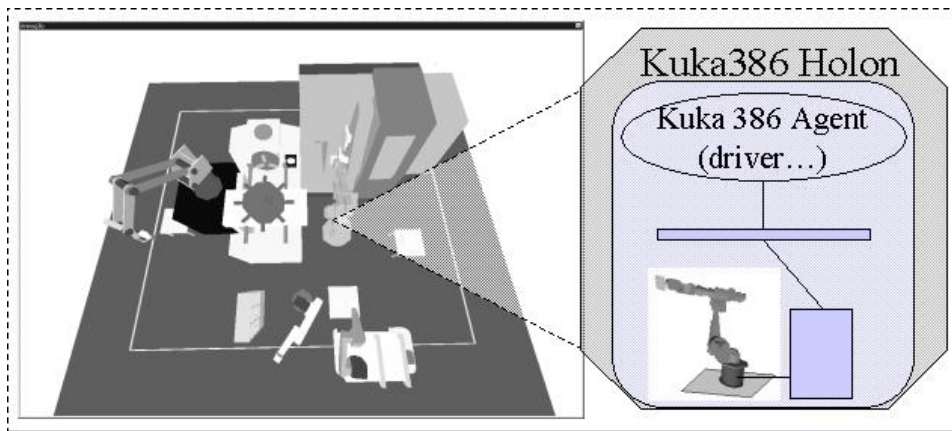
quickly as possible to the physical resource. Within a Resource-HL, metaphorically, the virtual-resource encapsulates the physical resource.

Actually, the computationally holonified resource can facilitate the construction of a computational holonic control once each resource is, by technical artifices, homogenized as a holon reachable within the *informatics scope*. The composition of Resource-HLs can be viewed as a decoupling mechanism between the computational control and concerned automatics activities (i.e. monitoring and command) encapsulated within Resource-HLs<sup>63</sup>.

### 3.9. Resource Holon in ANALYTICE II

The computational holonification of resources has been implemented in ANALYTICE II. Figure 25, for instance, presents the Resource-HL called *Kuka386* that is implemented in ANALYTICE II. The *emulated robot kuka386* and an *equivalent software agent* representing the robot at control side of ANALYTICE II compose this *Resource-HL Kuka386*. The agent, i.e. *the virtual kuka386*, handles monitoring and services requirements for the emulated robot by means of the virtual network.

In each Resource-HL implementation in ANALYTICE II, like the implementation of the Resource-HL *Kuka386*, the virtual resource receives signals (“events”) from the emulated resource via virtual network and appropriately interprets them in a manner that the holarchy can understand (i.e. it suitably expresses the information to the holarchy entities). Also, the virtual resource transfers “stimulus” (i.e. services requests from other holons in the holarchy) to the emulated resource via the virtual network. In fact, the virtual network enables appropriately routing packets of information between virtual and emulated entities.



**Figure 25: Example of Resource-HL in ANALYTICE II.**

The implemented Resource-HLs (in ANALYTICE II) comprise the monitoring and the high-level command. Typically, the monitoring and the high-level command are considered control functions, however they are assigned to the Resource-HLs in the holonic approach described in this thesis. Thus, in this approach, the Resource-HLs carry out these typical control functions while the holonic control is responsible for the *cooperation* of Resource-HLs for purposes of achieving agile objectives.

Effectively, the control in automated MS has as a fundamental role to observe the states of resources and their internal processes for making decision. These decisions, which are based on those observations and on a predetermined logic, are related to actions to be carried out in

<sup>63</sup> Decoupling mechanisms play an important role in increasing the size and complexity of a system that survives in demanding environments. Not only does it allow system components to function relatively independently, but more importantly it enables the different system elements to evolve independently over time (Wyns, 1999).

the *manufacturing processes* (e.g. *resource cooperation*) by means of the resources capabilities. In an *alternative* manner, the control activity in the holonic approach is not directly concerned with “simple” resources, it is concerned with Resource-HLs *encapsulating* part of the typical control activities.

### 3.10. A set of Resource-HLs

The computational holonification, as described in the previous section, was applied to all resources in the simulated cell presented in Figure 22. Also, a holonic modeling of the implemented Resource-HLs was done using the UML derivation presented in the last chapter.

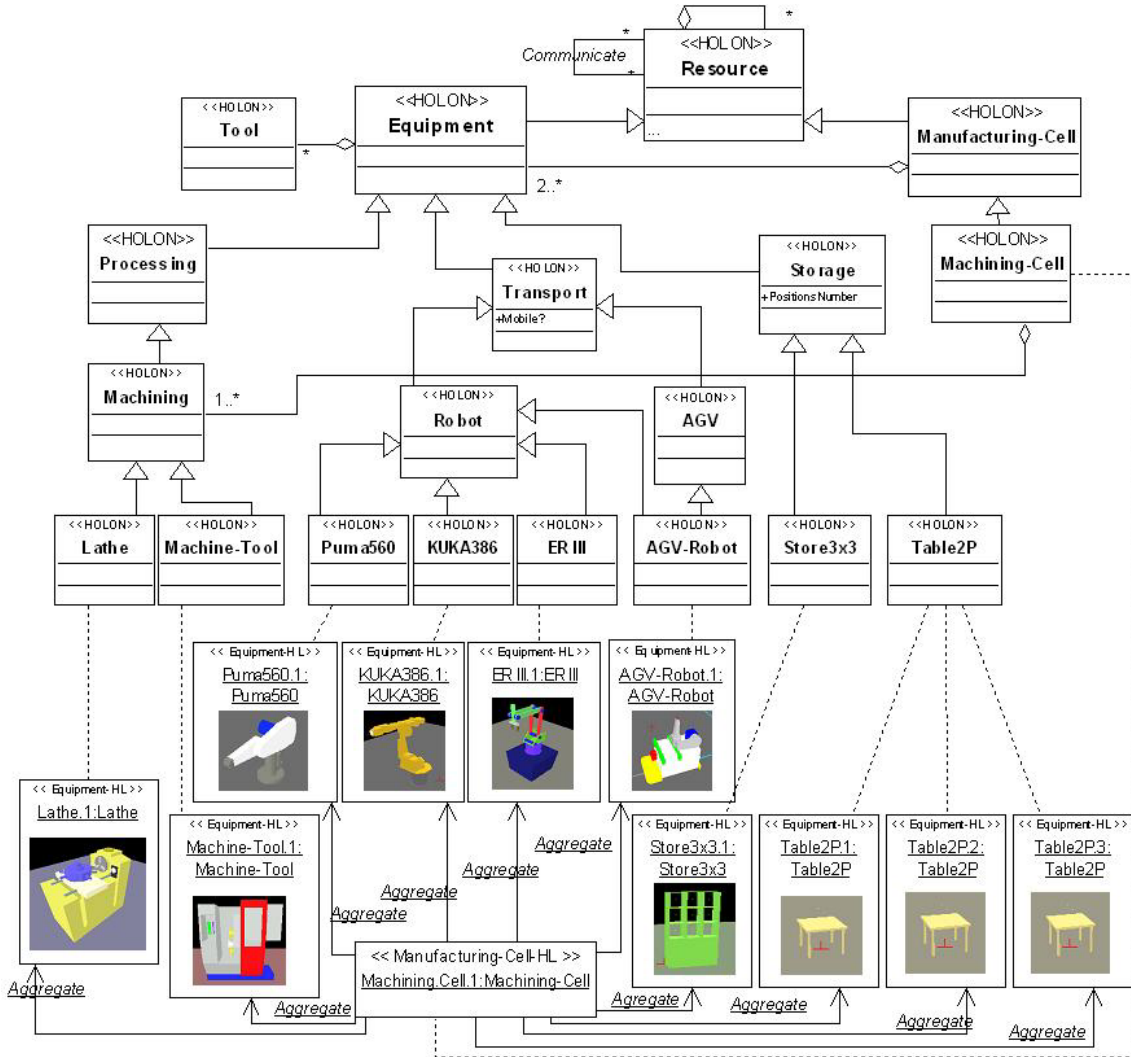


Figure 26: Specialization/generalization tree about Resource-HL<sup>64</sup>.

The model, a *Holonic-Class Diagram*, allows a better classification and understanding of different kinds of implemented Resource-HLs. This holonic modeling is presented in Figure 26, where *Resource* is a *Holonic Base Class*. The *Resource* has two subclasses, the *Equipment* and *Machining-Cell Holonic Classes*.

The *Equipment Holonic-Class* has three subclasses, namely the *Processing (Machining)*, *Transport*, and *Storage Holonic-Classes*, each one with its own subclasses. Also, in this

<sup>64</sup> This graph of Holonic-Classes is coherent with efforts to classify resources or their holons found in the literature (Langer et al., 2000)(Wyns, 1999).



model, the *Equipment Holonic-Class* is related to the *Tool* and *Manufacturing-Cell Holonic-Class*s, which can be understood as a holonic interpretation and modeling of tools and manufacturing-cells. The *Manufacturing-Cell Holonic-Class* is complementarily specialized in a *Machining-Cell Holonic-Class* for allowing instantiate Machining-Cell-HLs<sup>65</sup>.

Figure 26 also shows *Holonic-Class* instances, i.e. holons, within this *holonic model*. In particular, they are instances of the *Machining-Cell* or of subclasses derived from the *Equipment Holonic-Class*. In fact, these instantiations demonstrate a model property: the *Equipment-HL reoccurrence*<sup>66</sup>. The *reoccurrence* property is important because it permits considering some generality in the model, even in only one designed system.

### 3.11. Self-similar Resource-HLs

In computational holonification over the equipment of the simulated cell it was observed, by means of ANALYTICE II simulations, that the Resource-HLs behave as expected. However, in this holonification and also in the related holonic modeling, it was observed that each Resource-HL has its particularities. The Resource-HLs, even if entities of a same nature (i.e. holons), still present some heterogeneity. Therefore, a second question related to resource holonification concerns how to overcome the heterogeneity with respect to Resource-HLs, with the aim of facilitating the attainment of a holonic control solution.

The homogenization of Resource-HLs can be by means of similar interfaces, whereby each Resource-HL would have a similar manner to interact with other holons in the holarchy. The Resource-HLs would present only a patterned interface to the holarchy carrying out their functions (monitoring and service execution) decoupled from holonic control as much as possible. Since the interfaces would hide internal differences of Resource-HLs, they would appear to the external entities within the holarchy as similar entities and even as self-similar entities if considered Resource-HLs composing other high-level Resource-HLs.

Before defining a patterned interface to Resource-HLs, it is necessary to understand their interactions with the other holons within the holarchy. Each Resource-HL, at the interface level, can have different services and information enabled for use by other holons, such as control holons. For instance, the *Machine-Tool.1*, presented in Figure 26, could have as enabled services ‘*prepare the machining of a kind of part*’ and ‘*start the machining of a part*’ while also making available to the other holons such information as its ‘*status*’ or the ‘*status of an internal tool*’.

Note that Resource-HL information that could be made available is concerned with the states of a feature related to the holonified resource while an enabled Resource-HL service is related to a set of possible services the Resource-HL can provide. In other words, the Resource-HL must provide information about its attributes as well as methods for executing its services.

Based on the above analysis, a proposal shall be made to make available to other holons within the holarchy the Resource-HL methods and attributes states. A proposal could be made

<sup>65</sup> A Machining-Cell-HL “encapsulates” an HSC together with other holons such as Equipment-HLs, while the Equipment-HL, in turn, encapsulates Tool-HLs. The Tool-HL has importance inside the Equipment-HL that encapsulates it. For example, a dedicated and exclusive Tool-HL in the “inoperable” state can trigger its aggregator Equipment-HL to be in an “inoperable” state as well. Tool-HLs are also important in the HSC context when Equipment-HLs share them, but this is not the case in the considered machining system.

<sup>66</sup> More specifically - (a) *Machining Holonic-Subclass*: two reoccurrences, i.e. *Lathe.1* and *Machine-Tool.1*; (b) *Transport Holonic-Subclass*: four reoccurrences, i.e. *Puma560.1*, *KUKA364.1*, *ERIII.1*, and *AGV-Robot.1*; and (c) *Storage Holonic-Subclasses*: four reoccurrences, i.e. *Store3x3.1*, *Table2P.1*, *Table2P.2*, and *Table2P.3*.

using entities called subholons<sup>67</sup> (SHLs). The attribute states could be enabled by means of a type of subholon called an *Attribute-SHL*, while each method could be enabled by means of a type of subholon called a *Method-SHL*<sup>68</sup>. The aggregation of Method-SHLs and Attribute-SHLs by Resource-HLs is modeled in the diagram presented in Figure 27.

The interaction with the external world by means of Method-SHLs and Attribute-SHLs serves to homogenize interaction with the Resource-HLs within the holarchy at the interface level. For instance, a holonic control could interact with Resource-HLs by simply obtaining information from their Attribute-SHLs and by calling their Method-SHLs. At the interface level, the Resource-HLs are self-similar entities, which facilitates their coordination.

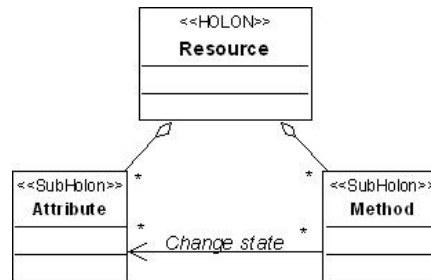


Figure 27: Attributes and Methods Holonic-Classes.

### 3.12. Self-similarity

The computational holonification of resources and interfaces for hiding differences between Resource-HLs have been proposed for avoiding heterogeneity, with the aim of simplifying interaction with Resource-HLs on the part of concerned entities in the holarchy such as control holons.

The reasons for avoiding heterogeneity in systems have been described in the general literature. For example, Minski (Minski, 1985), Waldrop (Waldrop, 1992), and Warnecke (Warnecke, 1992) have pointed out that unnecessary heterogeneity is avoided, and even suppressed, in complex systems in order to simplify the interaction environment<sup>69</sup> (Wyns, 1999).

The need for avoiding heterogeneity in complex systems is related to their typical roles within distributed information processing and decision-making. The distributed entities need to maintain, and even build, models of relevant parts of the system in order to optimize global goals by means of local decisions. However, this task can be complex, due for instance to very probable inconsistencies between distributed models. Any complexity reduction within this model-building task is likely to increase the overall performance of the system (Wyns, 1999).

<sup>67</sup> A sub-holon is a part of the holon. However, a sub-holon may itself be considered as a holon. In other words, a sub-holon is a holon that is part of another holon.

<sup>68</sup> This idea is based on features found in the knowledge representation approaches of Artificial Intelligence (AI), specifically the concept of a Frame (object-like) with slots (Rich et Knight, 1991) (Russell et Norvig, 1995), as well as on concepts of the Object Orientation (OO) paradigm, namely the Object and Class concepts where a Class, as well as the Objects that instantiate this class, contains attributes and methods (Rumbaugh et al., 1999). However, one difference between the approach described in this thesis and the AI/OO approaches is that Attribute-SHLs and Method-SHLs are conceived of and represented as independent entities, with their own behavior (as described below, in sections 3.13 and 3.14), being explicitly aggregated in their owner.

<sup>69</sup> Human societies react aggressively to individuals that behave abnormally, eccentrically or unfashionably. Minski (Minski, 1985) sees this as an effort of individuals to reduce the complexity of the environment that they have to take into account; this clarification is reinforced by the fact that this aggressiveness typically increases with decreasing mental processing ability of individuals (Wyns, 1999).

Concordantly, the concept of fractal or self-similar entities, as in Fractal Factory (Warnecke, 1992) or HMS, (Bongaerts, 1998) promises to reduce the complexities related to overall system development and efforts, as well as complexities involved in system execution, by means of facilitating heterogeneity avoidance. In this sense, the complexity in system development and execution can be reduced by using similar and self-similar entities due to the solution reusability (even in different application levels) and compliance with uniform models towards the remainder system whenever possible<sup>70</sup> (Wyns, 1999).

Wyns also states that model-building and model-compliance by the individuals helps the organization of groups and even the self-organization of groups. Individuals are able cooperate in achieving an overall system behavior that is mutually beneficial without requiring centralized planning or control. Similarity and self-similarity helps to create overall system behavior without a guiding overall control (Wyns, 1999).

The benefits expected when using compliance models have motivated, in this thesis' research, an orientation toward using entities that are modeled and implemented with the goal of achieving similarity and self-similarity (i.e. homogeneity) as much as possible. The proposal to design Resource-HLs so as to be self-similar holons aims to make easier the holonic control proposition and composition. The proposed (self)-similar Resource-HLs also facilitate their own composition with entities functionally independent and permit to considerate the generality of the Resource-HL external model due to patterned interactions in the instances.

In some way, the Resource-HLs with Attributes and Methods, as a patterned way of interaction, are (self) similar entities constituting a generic solution to enable high-level monitoring and services requesting to the holarchy. The solution is similarly applicable in the horizontal plan (means that it is applicable to different resource in a same *level* such as *equipment level*) as well as in a vertical plan (means that it is applicable to resources in *different levels* such as *equipment* and *manufacturing cell levels*).

### **3.13. Attribute Subholon**

The states of the attributes of Resource-HLs are kept and communicated to the holarchy by means of subholons (SHLs) generically referred to as *Attribute-SHLs*. An Attribute-SHL is an instance of the *Attribute Holonic-Class*, presented in Figure 28, whose stereotype is *SubHolon*. The Attribute-SHL is mainly referred to within this thesis as simply *Attribute*.

An example of an Attribute is one that indicates the Puma560.1 status, e.g. busy or not. Another example of an Attribute is one that indicates the status of the Puma560.1 end-effector, e.g. open or closed. Basically, an Attribute keeps discrete states values its aggregator Resource-HL communicates to it. The Resource-HL changes the Attribute state based on the information received from the encapsulated resource, or even based on inferred information.

In ANALYTICE II, for instance, the events related to the respective robot Puma 560 are recovered from the virtual net and addressed to the respective virtual resource that is responsible for updating the information concerting the Attributes. Alternatively, the virtual resource could infer the state of the physical resource based on the service requested and on an internal model establishing the service execution time (i.e. a watchdog).

An Attribute is able to communicate its present state (or even a past state, if such a state has been stored) to other holons. Moreover, each Attribute is able to automatically notify other

---

<sup>70</sup> Computer experiments have also shown how compliance models, as compared with simple models, enable individuals in a competitive environment to cooperate, and to obtain, jointly, a superior performance over non co-operative individuals (Wyns, 1999). Rosenschein gives examples of such cooperation strategies (Rosenschein and Zlotkin, 1994).

interested entities concerning its state at the point when the state has been changed or a specific state is reached<sup>71</sup>. In order to carry out notifications, each Attribute must know which holons need to be notified. This can be accomplished by having each interested holon communicate to the Attribute its need to be notified.

One example of a group of holons interested in Attribute notifications is a set of control holons that compose a holonic control. A more specific example would be the control holons, composing a Holonic Supervisory Control (HSC), these holons being interested in the Attributes of Equipment-HLs of the holonic process-cell presented in this chapter.

In fact, the designed Equipment-HLs and a related HSC could constitute the essence of a Machining-Cell-HL. The Machining-Cell-HL could also include Attributes as components. An example of such Attribute is one that communicates current capacity of the machining-cell for receiving parts. This Attribute might be useful for control holons in a Holonic Shop Floor Control (HSFC).

A Resource-HL, such as the Machining-Cell-HL, could also use the states of a group of Attributes to determine the state of another specific Attribute. For instance, the capacity Attribute in the Machining-Cell-HL could determine its state based on the status Attributes of relevant Storage-HLs.

Additionally, an Attribute could notify its “future states” to interested holons if models for predicting its probable state in a space-time are enabled. For example, the capacity Attribute in the Machining-Cell-HL could use predicted production to infer the probable capacity that the Machining-Cell-HL will have a  $x$  minutes from now and then notify interested holons of this predicted capacity. Another example could be a status Attribute in the Puma560.1, which could use kinematics models and joint-angles value known in real-time to predict a probable free (or ready) state.

The notification concerning predicted future states of Resource-HL Attributes can be useful for decision-making by other entities, while the notification current states of Resource-HL Attributes is essential to the proper decision-making on the part of the holonic control. To summarize, the Attributes provide a (self) similar way (i.e. a common interface) for Resource-HLs to use to communicate their states to other interested holons within the holarchy, such as holons of a Holonic Discrete Control System (HDCS).

### **3.14. The Method Subholon**

Resource-HLs can receive requests for services or, in the case of Equipment-HLs, “high-level commands” from other holons. The requests received by Resource-HLs are handled by a subholon generically referred to as a *Method-SHL*, or simply as a *Method*.

A Method is an instance of the *Method Holonic-Class*, presented in Figure 28, whose stereotype is a SubHolon (SHL). An example is a Method included in an Equipment-HL or in a Machining-Cell-HL that is responsible for recording the process information about processed parts.

In the case of an Equipment-HL that receives as a request a “high-level command”, this will be handled by a subholon called a *Command-SHL*, or simply a *Command*. This subholon is an instance of the *Command Holonic-Class* (presented in Figure 28) derived from *Method Holonic-Class*. An example is a Command to *articulate* (open or close) the end-effector used by PUMA560.1 or to move it to a specific position.

---

<sup>71</sup> For example the hypothetical Attribute Temperature within an Equipment holon could notify other holons of its state only when the temperature reaches a state such as High (e.g. if value > 30 degrees).

A Command, in an Equipment-HL, internally and correctly sets the command sent to the encapsulated resource. In the case of Equipment-HLs tested within ANALYTICE II, when a Command is required, a well-made (well-parameterized) information packet (i.e. event) is created to be sent to the emulated equipment by means of the virtual network.

Methods or Commands can also be composed of (sub) Methods and (sub) Commands to better organize their internal composition. Methods and Commands could, for example, have means to observe and communicate the services required as well as to record information about these services.

Requests for execution of Methods or Commands are made by means of *instigations* of other entities, such as control holons. A specific example could be the Holonic Supervisory Control (HSC), in the holonic machining cell presented in this chapter, *instigating* (i.e. triggering) Methods and Commands to control the cooperation of the Equipment-HLs.

Generically, the Methods and Commands provide a (self) similar way (i.e. common interfaces) within the Resource-HL for use when external entities (such as HDCS holons), need to request services.

### 3.15. Resource-HL Formalization

In this section, the formulation of the automatic synthesis of a control system, proposed by Fusaoka is used (Fusaoka et al., 1983). Fusaoka’s formulation has already been introduced in section 2.5.

In brief, Fusaoka’s formulation consists in defining the control rules of the dynamics of a physical system, based on the behavioral goals to be met, according to the following law: Dynamics  $\wedge$  Unknown Control Rules  $\supset$  Goal, here summarized as  $D \wedge U \supset G$ .

A holonic interpretation of that equation is presented in Equation 4 to formalize the composition of Resource-HLs (or more specifically Equipment-HLs), envisaging them as elements whose cooperation will be managed by a HDCS.

D =	capabilities of the resource	i.e. hard-part of the Resource-HL
	features of the resource	
U =	unknown virtual-resource	i.e. soft-part of the Resource-HL
G =	high-level service requirements	i.e. Methods availability
	high-level monitoring	i.e. Attributes availability
	interested entities aware	
	notification to interested entities	

**Equation 4: Composition of Resource-HL.**

In the Equipment-HL case, for instance, specialists in equipment could *determine the value of U*. Once this *equation is solved* for each Equipment-HL, the result is a set of pieces of equipment “encapsulated” in similar Equipment-HLs and then integrated into the holarity.

The *library* (of emulated piece of equipment) of ANALYTICE II provides the means for understanding the specification of each emulated piece of equipment and then generating the respective Equipment-HL by means of the construction of the respective virtual resource. The ANALYTICE II Equipment-HLs have been constructed in this way, *applying* Equation 4.

### 3.16. Holonic Control

In the scope of the HMS, the proposed simulated machining cell must be composed not only of Equipment-HLs, but also of other elements, such as the entities needed for carrying out the holonic supervisory control, i.e. control entities.

These control entities can be abstractly considered a Holonic Supervisory Control (HSC), as designed in the holonic-class diagram presented in Figure 28. An example of a HSC is one whose purpose is to control the cooperation of the Equipment-HLs needed for producing V1 and V2 parts, whose production processes were defined in Figure 23.

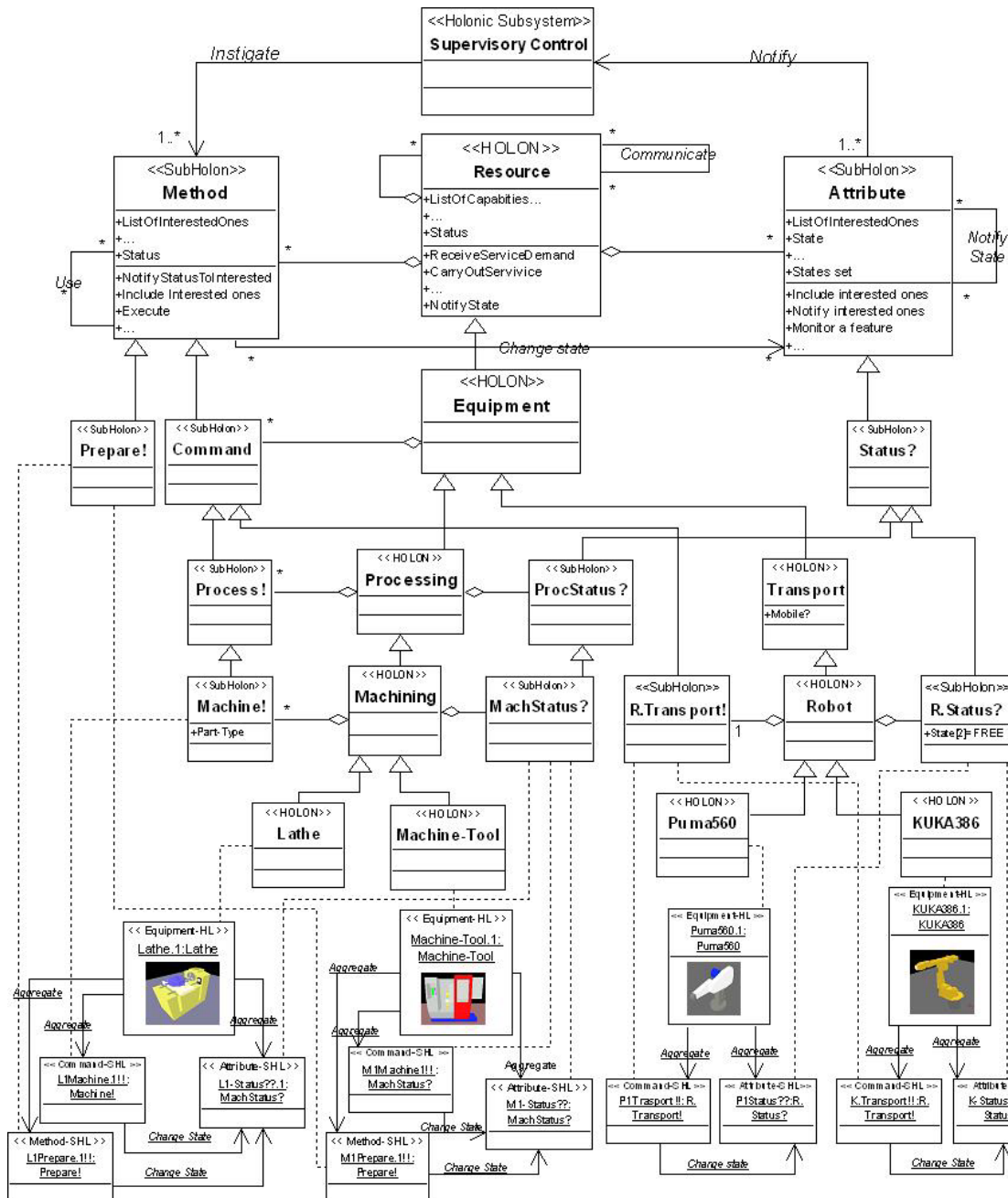


Figure 28: Attribute-SHL and Command-SHL.

The HSC could be process-driven or product-driven. Either way, Equipment-HLs are essential elements to be *managed* by the HSC. Thus, it is proposed that the HSC *instigates* the cooperation of Equipment-HLs, by means of Methods and Commands, after each decision-

making that is, inclusively, based on Attribute states (of the Equipment-HLs) provided in the form of notifications.

The diagram in Figure 28 shows possible interaction between the Resource-HLs and the HSC. Specifically, the *notification* interaction is modeled by the *notification relation* between the *Attribute Holonic-Class* and the *Supervisory-Control Holonic-Class* while the *instigation* interaction is modeled by the *instigate relation* between the *Supervisory-Control Holonic-Class* and the *Method Holonic-Class*.

Figure 28 also shows specific Equipment-HLs of the case study instanced from *Holonic-Subclasses* derived from the *Equipment Holonic-Class*, and how these *Holonic-Subclasses* are related to *Holonic-Subclasses* derived from the *Attribute* and *Method Holonic-Classes*. In fact, Figure 28 shows even some instances of these *Holonic-Subclasses* derived from *Attribute* and *Method*, which the HSC may later be assigned to manage.

The use of instances within the diagram of Figure 28 serves to highlight the similarities and reoccurrences of instances within the case study (as explained in section 3.10). Actually, they are all instances of subclasses derived from only three main base classes: *Equipment*, *Attribute*, and *Method Holonic-Classes*.

### 3.17. Remarks

This chapter has clarified the structure of ANALYTICE II, presenting a machining-cell designed within ANALYTICE II, and proposing a computational holonification of emulated-resources generating Resource-HLs.

The holonification aims to facilitate the design of HDCS architecture by using Resource-HLs that encapsulate specific resource complexities. In this sense, Resource-HLs have been modeled as *atomic entities* and their implementation within ANALYTICE II has effectively generated functionally independent Resource-HLs.

In the chapter development, before the resource holonification, it was presented ANALYTICE II as a robust simulator prototype, generated as result of more than a decade of research about manufacturing system with flexibility and integration features.

Although ANALYTICE II is a high-quality tool, solutions need to be found for dealing with some of its technical limitations. For example, it would be useful to distribute the simulation architecture (e.g. distribute the equipment emulation), to develop a better network simulation (e.g. to emulate communication protocols), and to develop a module for facilitating or automatically generating facility layouts<sup>72</sup>.

Other more important modifications that ought to be made include: the expansion of the equipment library, facilitation of experiment construction, and the development of a module to store, recover, and analyze simulation data.

The Behavior Model (BM) for each piece of emulated equipment, a kind of controller, is constructed using a Wizard Module that generates C++ code. However, currently, models must be finished with hard programming. Thus, it would be useful to develop a more “programmer-friendly” process for creating BMs. It would also be worthwhile to generically improve the BM by developing interfaces compatible with actual practices in the industry.

---

<sup>72</sup> Some efforts using a CORBA implementation have been made separating an implemented control and the other ANALYTICE II modules. The ORB (i.e. a CORBA implementation) used was the freeware Omni from AT & T Laboratories Cambridge (UK), downloaded from the internet site [www.uk.research.att.com/omniORB/index/html](http://www.uk.research.att.com/omniORB/index/html).

One industrial practice is the use of GRAFCET or Relay Ladder Diagrams for the *setting* of a specific controller such as PLC (Programmable Logic Controller). Also a current tendency is the functional-block orientation, highlighting the IEC 1499 approach<sup>73</sup>. In this sense, in the emulation of machining-equipment within ANALYTICE II, it is already possible to define pseudo-CNC (Computerized Numeric Control).

This last point could generate a discussion about the degree of detail appropriate for use within simulation/emulation, because this type of technique must consider essential features of the real system, at a certain level of abstraction, so as to test or observe some specific properties. Depending on the interest of each user, the discussion would be centered on the level of abstraction from the real system that is appropriate for the simulation/emulation.

Opportunely, the present chapter proposed a computational holonification of emulated resources. In particular, it proposed an agent (a virtual resource) for handling each emulated resource at a high-level, thereby hiding specific complexities. Therefore, this allows create some independence about the abstraction level in the operational part, within a systemic viewpoint.

This virtual resource/emulated resource pair constitutes the Resource-HL. Resource-HLs encapsulate some control responsibilities, namely high-level monitoring and command, with the aim of achieving a functional independence from other control functions, such as decision-making and coordination of Resource-HL cooperation.

A holonic model was also proposed for Resource-HLs and its interaction, based on resource encapsulation and (self) similar features. Resource-HLs were projected to be (self) similar ones: for all Resource-HLs, monitored states can be notified using patterned Attributes subholons and their services can be reached by means of instigations over patterned Methods/Commands subholons.

Consequently and implicitly, a first meta-model for Resource-HL was proposed based on Resource-HL reoccurrence and (self) similarities in the model. In this sense, this solution has been understood as system pattern to monitoring/command in some works (Simão et al., 2002)(Stadzisz et al., 2003), which consider the problem by design and architectural patterns viewpoint. It is expected that these patterns, based on the (self) similarity concept, may facilitate the decision-making and coordination issues in HDCS.

In this scenario of resource homogenization, it is feasible to create controls that use the states communicated by Attributes to determine how to execute coordination, based on inflexible strategy, and effectively coordinate instigating Methods/Commands. This structure can be applied to many levels of Resource-HLs, implementing hierarchic controls.

On the other hand, it would be equally feasible to improve each Resource-HL, by adding the capabilities for considering Attributes from its neighbors, deliberating about its and their Attribute states and instigating its Methods and/or their Methods, based on that deliberation, in a heterarchical way.

The next step of the current research project is to find an effective holonic solution to control, utilizing the best features of both hierarchism and heterarchism, and taking into account correctness features of the control in the terms defined in the last chapter.

---

<sup>73</sup> The IEC 1499 standard, actually IEC 61499 (IEC 61499), includes an architecture model for distributed applications in industrial environments. It is developed especially for PLC environments, and includes a graphical specification language called 'Function Blocks'. Within the IEC 1499 system model, a distributed application is running on several devices (nodes in network), and in one or more processes in each device. Because IEC 1499 is basically developed based on a functional design principle (Function Blocks), its support by object-oriented design is limited (Heikkila et al., 2001).



## **Chapter 4**

---

Rule and Agent-oriented Control



# Chapter 4 : Rule and Agent-oriented Control

## 4.1. Introduction

The computational holonification of the emulated resources in ANALYTICE II, for creating Resource-HLs, has been defined in the last chapter to reach integration. After that, in a context of process-driven production, the next step should be to define the *holarchy rules*, i.e. a set of decisional systems. These holarchy rules should allow creating a HMS established upon Resource-HLs such as those defined in the earlier chapter.

In the second chapter of this doctoral thesis, the holarchy rules were first related to the Holonic Discrete Control System (HDCS). For that reason, the next step in this doctoral thesis is to propose a solution to this type of control, applicable to Resource-HLs like those simulated in ANALYTICE II. As previously presented, this control solution is an objective of the thesis.

This chapter proposes an architecture aiming at a solution to process-driven holonic control. The proposed solution is extended to explicitly be a product-driven holonic control solution in a next chapter. In fact, the architecture presented here will be part of a product-driven control architecture.

A process-driven HDCS must carry out the control of Resource-HL *cooperation*, in order to complete established production steps. The HDCS could carry out Resource-HL coordination, instigating their services by means of Methods, for implicitly reaching the set of desired Resource-HL cooperation.

However, each *coordination* moment should be based on a previous HDCS *decision* about the proper moment for coordination, for example to avoid deadlock and/or to respect production priorities. Briefly, a *decision set* should determine the suitable *coordination* set, which is traditionally referred to as *causal relations*.

The *decisions* are made by evaluating of system facts, e.g. states of Attributes from Resource-HLs, using specialized knowledge about the controlled system. This specialized knowledge is a factor to determine the exploitation degree of production flexibility (i.e. the degree of certain adaptability for agility) once it determines how to coordinate the Resource-HLs.

In this scenario, a set of *causal relations* is the essence of the control system. A set of grouped and correlated *causal relations* within one unique entity would be considered a monolithic control, while a set of separate but correlated entities, each one *handling* a different causal relation, would be considered a decoupled control.

A holonic control should be a decoupled one, composed of decoupled entities such as soft-holons. Each Control Soft-holon could *handle* one *causal relation*, as illustrated in Figure 29. The Control Soft-holon would be constituted by one subholon to deal with the *decision part* of the causal relation, named Condition, and by another subholon to deal with the *conclusion part* (about coordination) of the causal relation, named Action.



Figure 29: Control Entity.

By means of an inference process, the *Conditions* of the *Control Soft-holons* would have the appropriate opportunity to evaluate Attributes of Resource-HLs and then to reach conclusions about Resource-HL cooperation. Therefore, the concerned *Actions* of the *Control Soft-holons* could potentially execute, determining cooperation by means of Method activations.

A common technique for expressing causal relation entities is a set of rules, commonly called *production rules* in computer science (Russell et Norvig, 1995). A system like that, constituted by *rules*, is usually referred to as *Rule Based System (RBS)* and/or *Expert System (ES)* within classical Artificial Intelligence (AI) (Jackson, 1990)(Rich et Knight, 1991)(Russell et Norvig, 2003).

An Expert System (ES) normally *works* within an application domain, e.g. discrete control, with substantial causal and factual knowledge about a system (Simão et Stadzisz, 2002). Usually the causal knowledge is obtained from human experts while the factual knowledge may come from different sources, such as human or automatic observation. In a control case, for example, sensors could provide the factual observation.

The ES needs some reasoning mechanism (i.e. an inference process), for processing its factual and causal knowledge, in order to reach deductions about the considered system and to apply the conclusions reached, potentially affecting that system. An ES also may need a way to present its conclusions to the user and a way to update its inference base, i.e. rules and facts (Russell et Norvig, 1995)(Simão et Stadzisz, 2002).

Structurally, an Expert System (ES), or commonly called Rule Based System (RBS), consists of facts about the observed system, causal relations (normally rules explaining the name RBS) that specify the different ways to change the facts, and an Inference Engine that effectively changes them (or instigates their changes) based on rules and facts, carrying out the inference process (Simão et al. II, 2003).

The Inference Engine (IE) has an inference cycle, i.e. a process to match rules and facts to obtain the proved rules (e.g. when a new fact happens), that allows carrying out appropriate changes (Russell et Norvig, 1995). The changes can be carried out directly in the base of facts or indirectly by calling external functions (that can change the base of fact), following the knowledge of the rules (Pan et al., 1998).

In the control case, the matching would be like the decision-making and changes would be like coordination (that actually result in change of facts). The control solution foreseen in this work could be seen as a type of RBS. Nevertheless, some drawbacks of this technique should be dealt with, such as the coupling of rules in the inference process, which is inappropriate for a holonic control. In fact, the inference process would be the essence of dynamics of this rule-based control, which should support decoupled rules.

It would be expected that in this type of control, i.e. like an Expert System, any *control instance* with appropriate knowledge about the controlled system and an efficient and appropriate inference process could contribute in the production agility. Summarily, it would be expected that this type of control be an agile control, i.e. the control that allows agile production.

The research project described in this thesis makes use of RBS concepts as a source of inspiration to define an approach to holonic control. In this chapter, a way to *create* RBS-like is proposed and explored as a means for achieving a holonic control solution. This solution, targeted for an HMS in which Resource-HLs are like those proposed for ANALYTICE II, is elaborated in a manner that is consistent with the goal of achieving *openness* to correctness features.

## 4.2. Rule-oriented Control

The control solution inspired by RBS assumes that experts could compose causal relations, such as rules, using specialized knowledge about the system to be controlled. The rules, created by artificial or human experts, would be concerned with arriving at conclusions about Resource-HL coordination by means of decisions based on Resource-HL states. Each control designed following this approach would be a type of Expert System (ES), which *executes* in the computational environment where Resource-HLs are reachable.

A rule-oriented control solution could allow human experts, e.g. trainers and students, to design the control and/or to understand it in a more intuitive way, because it is natural to think in terms of causal relations as rules (Jackson, 1990). Therefore, this foreseen approach can be appropriated for “human in the process” issues which are another important concern, in addition to that of achieving agility, when developing a holonic solution.

An example of control knowledge used within an expert rule is presented in Figure 30, which allows observing the easiness in the comprehension of its knowledge. The rule structure is based on commonly occurring rule *composition* within ESs. This control rule would make part of the HDCS or HSC (Holonic Supervisory Control) model for the holonic machining-cell considered in the last chapter.

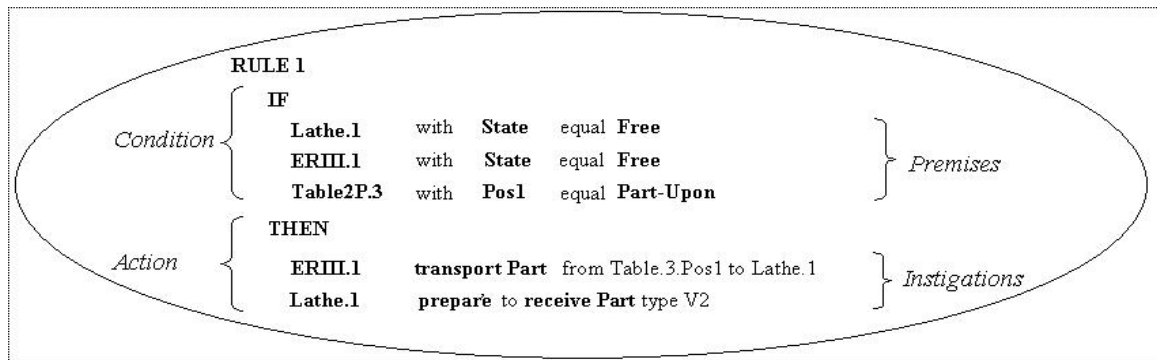


Figure 30: Control Rule expressing a causal relation<sup>74</sup>.

By the knowledge in the *condition* of this *expert rule*, it would be possible to know (infer) if the *Lathe.1 is free*, if the *ERIII.1 is free*, and if the *Table2P.1 has a Part upon Pos1* based on the awareness about the Equipment-HL states. More specifically, each *premise* of the *condition* (e.g. the first premise) can be concluded as true or false based on its knowledge (i.e. *Lathe.1 with State equal Free*) applied over the known state of the concerned Attribute (e.g. *Attribute Free of the Lathe.1 in True state*).

After that each premise is concluded as true or false, a boolean state of the *condition* of the rule can be found by a conjunction of the boolean states of the concerned premises. After that, an approbation decision about a rule may be directly made if: there is no conflict with other rules and its condition is determined to be true. This decision (i.e. the rule approbation) means to determine the appropriate moment to carry out the specifications of the action of the rule.

The action part of the rule presented in Figure 30 specifies that ERIII.1 must transport the part from the Table.3.Pos1 to the Lathe.1 and that Lathe.1 must be prepared to receive and process

<sup>74</sup> In RBS scope, a practice is the rules be connected to a frame system that defines their referenced objects (Jackson, 1990) (Rich et Knight, 1991). A frame is a representation of a complex element; it is identified by a name and consists of a set of slots. In fact, the Frame approach is similar to the Object-Oriented approach (Simão et Stadzisz, 2002). Resource-HLs with Attributes and Methods, as referenced in the Figure 30, can be understood as frames.

a part of type V2. In general, an action specifies a set of instigations that must be appropriately given to the Resource-HLs normally mentioned in the respective condition.

Metaphorically, the *rule* that reaches a *true* state expresses to the holarchy the suitable moment and the way to coordinate a considered *Resource-HL* set for reaching a determined cooperation. Briefly, a set of *rules* determines the when and how to carry out the control over *Resource-HL* cooperation. Therefore, the degree of agility of the control system is related to the expert knowledge inside the *rules* and to the inference process capacity to deal with this knowledge.

### 4.3. Control Solution Issues

The essence of dynamics of a control solution based upon rules is concerned with *the inference policy* used to carry out the inference process. Basically, this inference process consists in evaluating the Attribute states (i.e. facts) by using the knowledge of Premises that are the essence of Conditions of the Rules and in evaluating the truth or falsity of these Conditions based on the premises states (i.e. true or false).

A correct solution to control must have a good degree of reactivity, which is related to the computational complexity of an inference process (Lhoste et Pétin, 2001). Computational complexity has been also viewed as a relevant concern with respect to the inference process of Expert Systems (ES) or Rule Based Systems (RBS) (Russell et Norvig, 1995).

In RBS, the inference process is carried out by an Inference Engine (IE). If the goal were to design a small-scale ES, it would be trivial to develop a simple program, as a kind of IE, where in each inference cycle all *facts* would be *matched with all rules*. However, this approach results in an exponential computational complexity in respect to the rule number (Pan et al., 1998).

In IE for ES, it is important to avoid *the temporal redundancies* in the matching process of *rules* and *facts*, e.g. computations for matching the *facts* with the *premises* should be carried out only once in this process. It is also important to avoid the structural redundancies, e.g. knowledge about *premises* should be appropriately shared between objects representing different (rule) conditions (Pan et al., 1998).

When carrying out of IE based on search methods, it is possible to reduce the computational complexity (related to the number of rules) to polynomial time complexity by avoiding the temporal and structural redundancies (Forgy, 1982)(Pan et al., 1998). In fact, IE implementations in RBS classically use some type of search to match rules and facts (Russell et Norvig, 2003).

In spite of the common use of search methods, they are not appropriate for use within inference processes within a distributed application, such as one involving distributed holonic control. In fact, searches within inference processes in a distributed environment (where the facts are distributed) are implicitly complex. For example, there is a possibility of a large inference cycle due to the time needed for exchanging information. Moreover, the length of the cycle would be further increased if failures occurred within the communication process.

The drawbacks related to using search methods within an inference process for distributed control are better discussed in the next chapter. In conclusion, it would not be appropriate to apply the searches method (as typically used within an RBS) to the type of solution being developed here, since the application foreseen here is a holonic control, where the system is potentially distributed.

The inference process to be used within this rule-oriented control, beyond being very reactive so as to assure a correctness feature, must also present an execution policy allowing computational distributed control, for observing holonic issues. This means that a set of control rules could potentially be distributed.

Moreover, this same policy should also be applicable to non-distributed implementations. This is justified because the rules distribution can be partial, meaning that sets of rules can be physically distributed but each set runs locally in a sequential way or pseudo-distributed way (e.g. using treads).

In short, the control should be thought in terms of decoupled entities, allowing distributed, non-distributed, and mixed implementations, while preserving correctness and holonic features.

#### 4.4. Rules as Holons

A rule-oriented control could be composed of a set of agents, for instance one agent for encapsulating the knowledge of each rule (Simão et Stadzisz, 2002). Therefore, in this approach of rule and agent-oriented control, rules would not be considered as a set of information controls within a monolithic block.

This doctoral thesis proposes that each rule be effectively concerned to an agent or a soft-holon called a *Rule*, which keeps and independently manages its causal knowledge, allowing the decoupling of rules. For example, a Rule soft-holon having the knowledge of the expert rule presented in Figure 30 could *by itself* determine the boolean state of its *condition* and appropriately carry out the specifications of its *action*.

More specifically, that Rule (a soft-holon) could know *by itself* if the Lathe.1, ERIII.1, and Table2P.3 are in the expected states (respectively “state equal free”, “state equal free”, and “pos1 equal part-upon”) based on its awareness of the states of the concerned Resource-HLs. Therefore, the Rule could also know *by itself* its boolean state.

As the next step, in the case where the Rule reaches a true state (and it is without conflicts with other Rules), it could conclude still *by itself* that the *ERIII.1 must transport the Part* (over the Table2P.1 Pos1) and that *Lathe.1 must prepare itself to receive a Part V2* (e.g. download a specific Numeric Control program).

The knowledge of an expert rule is considered the essence of a Rule holon. A Rule serves to evaluate Attribute states in a decision part called Condition, and instigate Methods in a conclusion part called Action, as sketched in Figure 31. In fact, both Condition and Action are defined as subagents or subholons aggregated in the respective Rule, which carry out services that help the Rule to properly execute its functions.

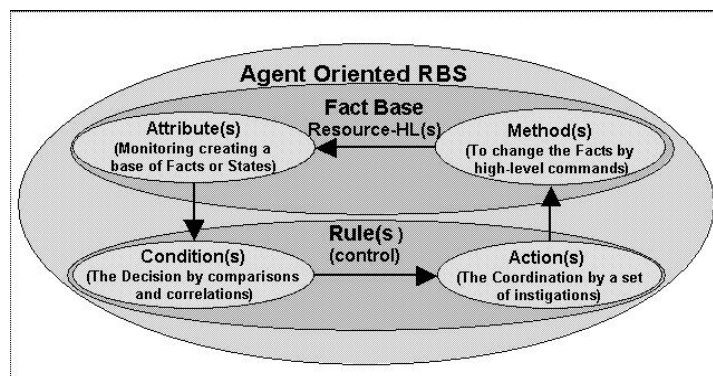


Figure 31: Rules as soft-holons.

A Condition, by itself, compares and/or correlates evaluation about the Attribute states of a set of Resource-HLs, normally Resource-HLs defined as co-operable in the respective Action. An Action, by itself, instigates Resource-HLs Methods (normally Methods of the Resource-HLs considered in the a respective Condition) in order to achieve the desired cooperation.

Each evaluation of the state of an Attribute could be performed by a specific subholon called Premise, with the aim of organizing the collection of the knowledge relevant for evaluating the Condition. Similarly, each instigation activity could be performed by a specific subholon called Instigation, with the aim of organizing the knowledge concerned with the Action.

Figure 32 illustrates the Condition of the Rule related to a non-empty set of Premise subholons and the Action of the Rule related to a non-empty set of Instigation subholons. To summarize, a Premise enables the evaluation of facts, represented as states of an Attribute, while an Instigation enables the instigation of a set of Methods.

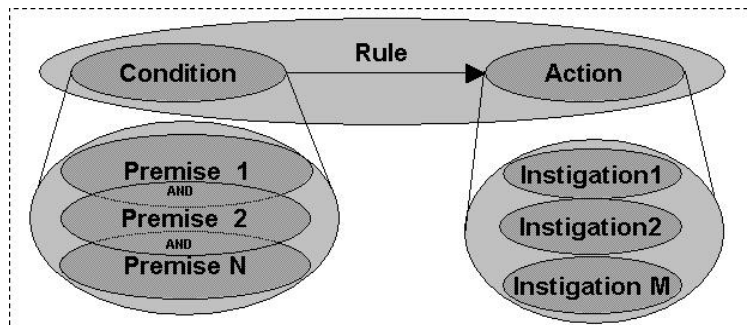


Figure 32: Control Entity.

Generically, Rules can be used whenever some determination about a set of Attributes states must be made in the holarchy resulting in activation of Methods. However, a more specific function of a Rule in the control is to know the moment that a group of Resources-HLs (e.g. Equipment-HLs) should collaborate in order to execute a sequence of operations, which is involved in a production step concerned with one or more types of products.

In fact, a *control Rule* considers itself as true when it determines the appropriate moment for the collaboration of a Resource-HL set. In short, a control Rule is responsible for instigating services within concerned Resource-HLs, in order to reached the desired cooperation, when two conditions apply: (1) it is in the state *true* and (2) it is not in conflict with other Rules.

#### 4.5. Rule Collaborators

This thesis proposes that a Rule be treated by sub-entities as presented in Figure 32, which can be called collaborators. These collaborators are responsible for specific functions related to the Rule. The initial intention in the use of collaborators is to propose an organized way to deal with the causal knowledge.

A Rule, as exemplified in Figure 33, is structurally composed of two subholons, the Condition and the Action. The Condition deal with decision-making while the Action deals with conclusion execution. In fact, the Condition deliberates about Attribute states of Resource-HLs by means of Premise collaborations and the Action instigates Methods of Resource-HLs to carry out services by means of Instigation collaborations.

*Premises* are subholons connected to Conditions, but they are not *components* aggregated within Conditions. A Premise collaborates in the Condition evaluation when it receives notification from one or even two Attributes about their states and it evaluates them. If the Premise reaches a new boolean value, it notifies the concerned Conditions. Each Condition, in



turn, calculates its boolean values by the conjunction of boolean values of connected Premises after the receiving of a notification from a Premise.

*Instigations* are subholons connected to Actions, but they are not aggregated within Actions. An Instigation collaborates in the coordination activity of an Action because it instigates one or more Methods to execute their services, providing the correct and necessary parameters (or arguments) for these Methods. Actually, the Action performs its function by activating the connected Instigations.

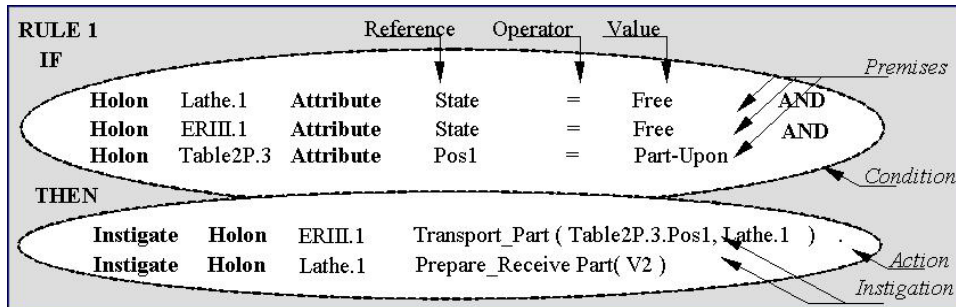


Figure 33: Example of Rule knowledge.

### 4.6. Rule Structure

Figure 34 depicts a holonic-class model. This model contains the classes of entities responsible for dealing with the knowledge of rules. The model presents the *Rule Holonic-Class* as well as *Holonic-Classes* for each of the Rule’s collaborators.

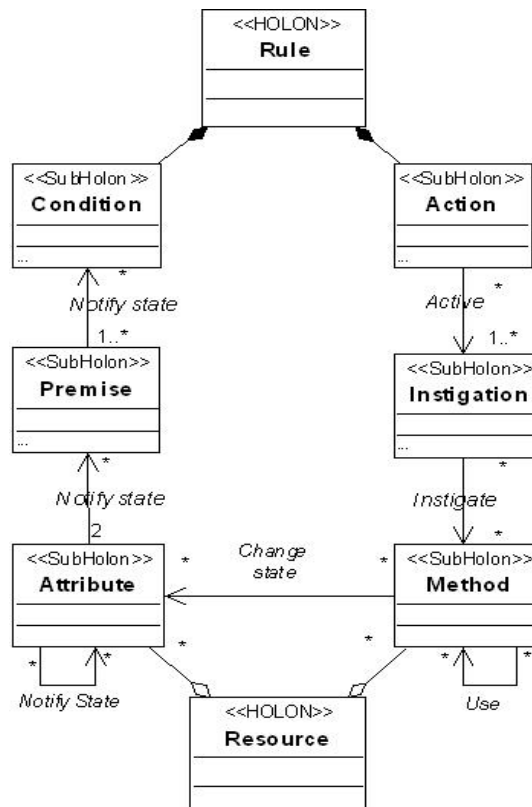


Figure 34: The structure of Rules and their collaborators.

The *Rule Holonic-Class* has an aggregation relationship to the *Condition Holonic-Class* (whose instances are Conditions) and another aggregation relationship to the *Action Holonic-Class* (whose instances are Actions).

A Condition is responsible for a fraction of all decisions related to the control of the Resource-HL cooperation. A Condition is related to a set of Premises, instances of the *Premise Holonic-Class*, which collaborate with it to help it carry out its responsibilities.

Each Premise knows the discrete value of an Attribute (received by notification) called *Reference*, a logical operator (for making comparisons) called *Operator*, and another value, called *Value*, which can be either a constant (see Figure 33) or the discrete value of another Attribute<sup>75</sup>. The Premise makes a logical calculation comparing the *Reference* with the *Value*, using the *Operator*.

An Action is responsible for a fraction of all coordination concerned with the control of the Resource-HL cooperation. An Action is related to Instigations, instances of the *Instigation Holonic-Class*, which collaborate with it to help it carry out its responsibilities (see Figure 33).

Each Instigation instigates changes in the Resource-HLs by means of Method activation, providing appropriate arguments to the Methods (being activated) when necessary.

#### 4.7. Summary of Inference Process

Figure 34 shows the diagram of *Holonic-Classes* for the proposed rule-based control. Note that these diagram presents that the knowledge of each rule be encapsulated within a Rule soft-holon and in related soft-holons called collaborators. These *Holonic-Classes* are defined in such a way that subholons used in evaluating the Condition of a Rule communicate by means of notifications. These features are concerned with the inference process explained in this present section.

A special feature of this rule-oriented control approach is an alternative to the inference process of Rules. It is proposed as an alternative to the standard RBS approach involving search methods. Unlike in the standard RBS approach, Rules are *not* updated by searching state information of Premises, and Premises are *not* updated by searching state information of Attributes.

This doctoral thesis proposes another inference process (illustrated in Figure 35), based on the collaboration of decoupled entities: Attributes notify only concerned Premises about states changed and Premises notify only concerned Rules about changed states, avoiding temporal redundancy, and thereby allowing a quick inference<sup>76</sup>.

Furthermore, the sharing of collaborations proposed in the model helps optimize the notification process because it eliminates structural redundancies. In short, the sharing of subholon collaborations (defined in the holonic-class diagram in Figure 34) consist of:

- A set of Premises may share Attribute collaborations (once theses Premises has the same Attributes as a *Reference*) and Conditions may share Premise collaborations (allowing the shared use of inferred information).
- A set of Instigation can instigate (at different times) the same Method and Actions can share Instigation collaborations.

As previously stated, the avoiding of structural and temporal redundancies is profitable in inference process established upon search methods. It is observed that the avoiding of

---

<sup>75</sup> The *Value* as a second *Reference* allows correlating values of Attributes.

<sup>76</sup> The awareness process about interested entities to be notified is carried out in the creation of the Rules. This is discussed in section 4.10.

structural and temporal redundancies is also profitable in an inference process establishes upon notification because it similarly avoids unnecessary interactions<sup>77</sup>.

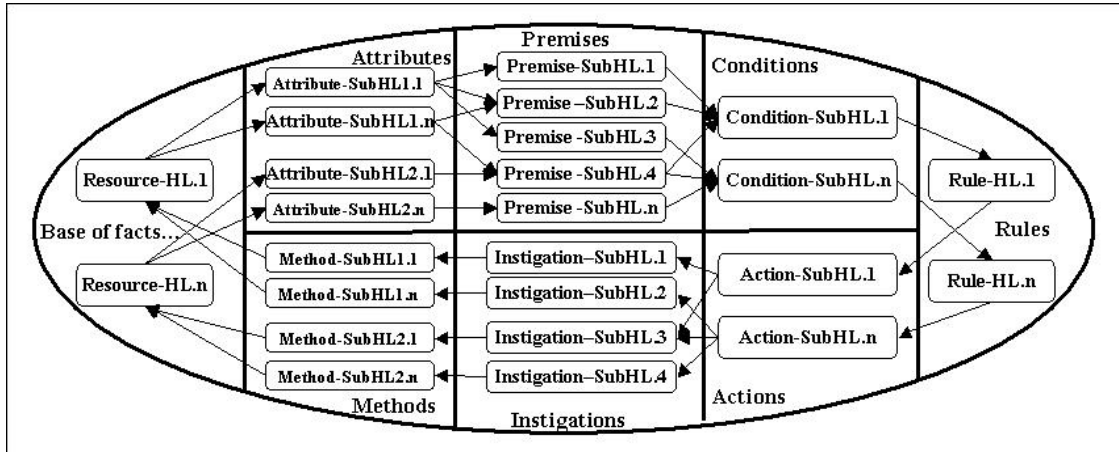


Figure 35: Notification mechanism.

## 4.8. Notification Mechanism

The Rules and their collaborators are decoupled agents that allow distributed and non-distributed implementations of the inference process. For an agent, implied in this inference process, the key point is to know the addresses of other agents interested in receiving its notification. An agent that notifies other interested agents (i.e. a notifier) has certain independence about the localization (i.e. local or remote) of the receivers.

The proposed architecture does not include an Inference Engine as an entity; the inference process is carried out by the collaboration of control entities by means of a notification mechanism, as illustrated in Figure 35. The proposed notification mechanism presents the following subtleties with respect to the inference process execution:

- Premises receive notifications of state change, i.e. new facts, from one or two Attributes<sup>78</sup>, once Attributes know which Premises have interest in their states (based on previously acquired information).
- After a Premise has received a notification with information about an updated state, it uses this information to make a comparison, i.e. a logical calculus, generating a boolean value for itself. The comparison is between the *Reference* and the *Value* using the *Operator*.
- If the new boolean value or state of the Premise is different from its previous value, the Premise notifies the interested Conditions about the current value. Each Condition that is notified uses this boolean value to perform or re-perform its logical calculus.
- The logical calculus of Condition is made by the conjunction of the boolean values of all connected Premises. If the result of this calculus is true, then the Condition puts the respective Rule in the state *true*.

The notification mechanism in this rule-oriented solution, comprising the sharing of agent collaborations, brings some advantages for discrete event control. Examples of advantages are the low computational complexity in the inference process and the actual functional

<sup>77</sup> In fact, the avoidance of temporal redundancy is implicit in this proposed approach.

<sup>78</sup> Each Premise is notified by its *Reference* and, when it is the case, by its *Values*.

independence of control entities by the reached entity decoupling, thereby allowing the control system distribution.

These advantages are detailed in this thesis, along with other relevant aspects of the solution. In order to discuss these advantages and aspects of the solution, the next section presents an ES-like, for controlling a simulated cell in ANALYTICE II, using the proposed solution. The ES-like consists of a set of Rules composing a control instance.

The control instance provides a demonstration of the feasibility of the solution, as well as a concrete example for illustrating the features of the solution. The control by means of Rules was implemented on the ANALYTICE II control-side using the C++ language, with the soft-holons being implemented as C++ objects.

## 4.9. Control Instance

### 4.9.1. Introduction

The proposed solution to holonic control was implemented for the resource-holonified cell in ANALYTICE II, presented in the previous chapter. The control instance is a kind of Holonic Supervisory Control (HSC) for the resource-holonified cell. This cell, which produces two virtual products (V1 and V2), is presented again in Figure 36.

The HSC, implemented as a rule-oriented and agent-oriented control, was composed of Rules for decision and Rules for coordination. The Rules for decision define when parts can come in or come out of the cell. The Rules for coordination define the appropriate moments for Resource-HL cooperation, aiming at achieve production steps. In addition, a policy to avoid conflict between Rules has also been defined during the HSC construction.

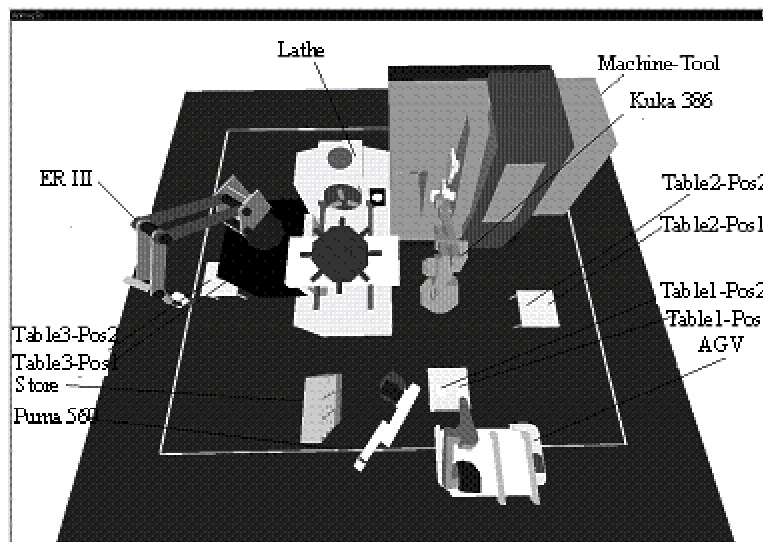


Figure 36: Virtual Machining Cell in ANALYTICE II.

### 4.9.2. Coordination Rules

A set of Rules was created for coordinating the cooperation between the Resource-HLs, thereby enabling the production of V1 and V2 parts. The Rules design was made focusing on a set of Resource-HL cooperation that allow reaching each operation defined in the process plans of each product-type.

The process plans for V1 and V2 parts are defined below, thereby detailing the process information related to the buffering position, which are information useful when creating the Rules. These process plans for V1 and V2 parts are respectively:

- { < Store [ pos 1, 2, 3, 4, 5, or 6 ] > < Table 1 [ pos 2 ] > < Machine-Tool > < Table 2 [ pos 1 or 2 ] > }
- { < Store [ pos 7, 8, or 9 ] > < Table 1 [ pos 2 ] > < Table 3 [ pos 1 ] > < Lathe > < Table 3 [ pos 2 ] > }

Each coordination Rule, concerned with a Resource-HL cooperation, considers a Transport-HL that transports a part from one place (i.e. a Store-HL or a Machining-HL) to another place.

Figure 37 presents an example involving interlocked Rules for transporting parts from the Store to Table1, position 1 or position 2, depending on the Rule. These Rules enable the performance of the first steps of the process plans for products V1 and V2, by transporting the parts in a determined sequence. Complementarily, Figure 38 presents other example of Rules, which enable completion of the subsequent steps defined in those process plans.

### 4.9.3. Rules in Conflicts

The Rules in Figure 37 were created in an interlocked way to avoid conflicts about shared Resource-HLs. For example, all those Rules use the Premise “Puma560.1 Status = Free”, but only an approved Rule that can effectively instigate a transport service in the Puma560.1 due to physical restriction.

<p>Rule A.1 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Busy Then Puma560.1 Transport(Store3x3.1. Pos1, Table2P.1. Pos1 )</p>	<p>Rule A.2 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Busy Then Puma560.1 Transport(Store3x3.1. Pos2 Table2P.1. Pos1 )</p>	<p>Rule A.7 If Puma560.1 Status= Free Table2P.1 Pos2 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos7 = Busy Then Puma560.1 Transport(Store3x3.1. Pos7, Table2P.1. Pos2 )</p>
<p>Rule A.3 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Busy Store3x3.1 Pos7 = Not_Busy Then Puma560.1 Transport(Store3x3.1. Pos3, Table2P.1. Pos1 )</p>	<p>Rule A.4 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Not_Busy Store3x3.1 Pos4 = Busy Store3x3.1 Pos7 = Not_Busy Then Puma560.1 Transport(Store3x3.1. Pos4, Table2P.1. Pos1 )</p>	<p>Rule A.8 If Puma560.1 Status= Free Table2P.1 Pos2 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Not_Busy Store3x3.1 Pos4 = Not_Busy Store3x3.1 Pos7 = Not_Busy Store3x3.1 Pos8 = Busy Then Puma560.1 Transport(Store3x3.1. Pos8, Table2P.1. Pos2 )</p>
<p>Rule A.5 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Not_Busy Store3x3.1 Pos4 = Not_Busy Store3x3.1 Pos5 = Busy Store3x3.1 Pos7 = Not_Busy Store3x3.1 Pos8 = Not_Busy Then Puma560.1 Transport(Store3x3.1. Pos5, Table2P.1. Pos1 )</p>	<p>Rule A.6 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Not_Busy Store3x3.1 Pos4 = Not_Busy Store3x3.1 Pos5 = Not_Busy Store3x3.1 Pos6 = Busy Store3x3.1 Pos7 = Not_Busy Store3x3.1 Pos8 = Not_Busy Then Puma560.1 Transport(Store3x3.1. Pos6, Table2P.1. Pos1 )</p>	<p>Rule A.9 If Puma560.1 Status= Free Table2P.1 Pos2 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Not_Busy Store3x3.1 Pos4 = Not_Busy Store3x3.1 Pos5 = Not_Busy Store3x3.1 Pos6 = Not_Busy Store3x3.1 Pos7 = Not_Busy Store3x3.1 Pos8 = Not_Busy Store3x3.1 Pos9 = Busy Then Puma560.1 Transport(Store3x3.1. Pos9, Table2P.1. Pos2 )</p>

**Figure 37: Rules to transport parts from Store for Table.1<sup>79</sup>.**

<sup>79</sup> The Rule soft-holons and collaborators are presented as causal knowledge in order to facilitate the control understanding. Surely, they could be presented as UML objects once they were implemented as C++ objects.

A conflict-resolution mechanism could be developed and implemented as a *conflict solver*. The conflict solver could perceive the conflicts between Rules involving Resource-HLs and resolve these conflicts, giving priority to only one of the conflicting Rules.

Such a conflict-resolution mechanism would facilitate the creation of Rules. For instance, the Rule A.9 would have only three Premises. The eight Premises that evaluate if positions of Store3x3.1 are empty, implemented only aiming at interlocked Rules, would not be necessary.

This implementation of a conflict-resolution mechanism is discussed in the next chapter. However, the interlocked Rules are efficient for avoiding conflicts and generating determinism in the control system anyway. Furthermore, these Rules (with *additional knowledge* for being interlocked ones) can serve as useful examples to illustrate Rule features.

#### 4.9.4. Rule Features

Only the set of Rules presented in Figure 37, executed in the controller instance over ANALTYCE II, allows the demonstration of some properties of the proposed solution. For example, it allows demonstration of the sharing of soft-holon collaborations (such as Premises and Instigations by the Rules) and the feasibility of the notification mechanism.

Furthermore, this set of Rules makes clear that Rules are similar. In fact, the Rules are very similar, with the only difference being the knowledge of each Premise (i.e. the *Reference*, *Operator*, and *Value*) and knowledge of each connected Instigation (i.e. the *referenced Method*), as well as the number of connected Premises and Instigations.

As previously discussed, (self) similar entities facilitate both the comprehension of the system, as the interactions within that system. Figure 38 presents another similar set of Rules. In fact, this second set of similar Rules enables performance of the remaining planned production steps. Furthermore, this set of Rules also provides confirmation of the feasibility of the features related to sharing of subholons collaboration, as well as of the notification mechanism.

<p>Rule B</p> <p>If</p> <p>KUKA386.1      Status= Free</p> <p>Puma560.1      Status= Free</p> <p>Machine-Tool.1   Status= Free</p> <p>Table2P.1      Pos1 = Busy</p> <p>Then</p> <p>KUKA386.1 Transport(Table2P.1 Pos1, Machine-Tool.1)</p> <p>Machine-Tool.1 Prepare_to_Receive(PartV1)</p>	<p>Rule D</p> <p>If</p> <p>AGV-Robot.1      Status= Free</p> <p>Puma560.1      Status= Free</p> <p>Table2P.3      Pos1 = Not_Busy</p> <p>Table2P.1      Pos2 = Busy</p> <p>Then</p> <p>AGV-Robot.1 Transport(Table2P.1 Pos2, Table2P.3 Pos1)</p>
<p>Rule C1</p> <p>If</p> <p>KUKA386.1      Status= Free</p> <p>Machine-Tool.1   Status= Part_Finished</p> <p>Table2P.2      Pos1 = Not_Busy</p> <p>Then</p> <p>KUKA386.1 Transport(Machine-Tool.1, Table2P.2 Pos1)</p>	<p>Rule E</p> <p>If</p> <p>ERIII.1      Status= Free</p> <p>Lathe.1      Status= Free</p> <p>Table2P.3      Pos1 = Busy</p> <p>Then</p> <p>ERIII.1 Transport(Table2P.3. Pos1, Lathe.1)</p> <p>Lathe.1 Prepare_to_Receive(PartV2)</p>
<p>Rule C2</p> <p>If</p> <p>KUKA386.1      Status= Free</p> <p>Machine-Tool.1   Status= Part_Finished</p> <p>Table2P.2      Pos1 = Busy</p> <p>Table2P.2      Pos2 = Not_Busy</p> <p>Then</p> <p>KUKA386.1 Transport(Machine-Tool.1, Table2P.2 Pos2)</p>	<p>Rule F</p> <p>If</p> <p>ERIII.1      Status= Free</p> <p>Lathe.1      Status= Part_Finished</p> <p>Table2P.3      Pos2 = Not_Busy</p> <p>Then</p> <p>ERIII.1 Transport(Lathe.1, Table2P.3 Pos2)</p>

**Figure 38: Rules to coordinate Resource-HLs.**

### 4.9.5. Decisional Rules

In this control instance, it is assumed that the Store is filled with rough parts for V1 and V2 products, in the correct predetermined position, always that the Store is completely empty. Thus, Rule 0 in Figure 39 is elaborated so as to signal the need for new rough parts. Normally, Rules outside of this Machining-Cell-HL scope, in other part of the holarchy, may consider the *Status* Attribute to decide when to reload the Store.

Similarly, it was assumed that parts on Table.3 Pos2 and on Table.2 Pos 1 or Pos 2 should be taken off the manufacturing cell. The proper moment to remove the parts is calculated by the Rule 1.1, Rule 1.2, and Rule 2, also presented in Figure 39.

The Rules presented in this section evaluate and change only Attributes within a particular Resource-HL. Therefore, each Rule could be aggregated in the respective Resource-HL. Likewise, all Rules presented in this chapter could be aggregated in a Resource-HL related to the machining-cell, i.e. a Machining-Cell-HL.

Additionally, the Machining-Cell-HL could have Attributes for dealing with its inputs and outputs states, which would be related to concerned Attributes of encapsulated Equipment-HLs (named Attributes of Store3X3.1, Table2P.2, and Table2P.3). This would allow more encapsulation of Machining-Cell-HL internal-holons by hiding them by means of higher-level Attributes.

<p>Rule A.0</p> <p>If</p> <p>Store3x3.1 Pos1 = No_Busy</p> <p>Store3x3.1 Pos2 = No_Busy</p> <p>Store3x3.1 Pos3 = No_Busy</p> <p>Store3x3.1 Pos4 = No_Busy</p> <p>Store3x3.1 Pos5 = No_Busy</p> <p>Store3x3.1 Pos6 = No_Busy</p> <p>Store3x3.1 Pos7 = No_Busy</p> <p>Store3x3.1 Pos8 = No_Busy</p> <p>Store3x3.1 Pos9 = No_Busy</p> <p>Then</p> <p>Store3x3.1 set Status(Empty)</p>	<p>Rule 1.1</p> <p>If</p> <p>Table2P.2 Pos1 = Busy</p> <p>Then Table2P.2 setPos1Status ( Part_Finished )</p> <hr/> <p>Rule 1.2</p> <p>If</p> <p>Table2P.2 Pos2 = Busy</p> <p>Then Table2P.2 setPos2Status ( Part_Finished )</p> <hr/> <p>Rule 1.3</p> <p>If</p> <p>Table2P.3 Pos2 = Busy</p> <p>Then Table2P.2 setPos3Status ( Part_Finished )</p>
--	--

Figure 39: Rules for signalize the reposition and removing of parts<sup>80</sup>.

## 4.10. Rule Composition

A Rule is created to reach some logic implication related to the considered system. For example, coordination Rule can know the appropriate cooperation moment for some Resource-HLs thereby concluding when to start of the cooperation between them. In this section, the creation of coordination Rules is explained in order to clarify details of the Rule composition.

Rule-creation involves focusing on a specific objective. For instance, a coordination Rule is created focusing on the control of the Resource-HL cooperation needed for performing a production step (e.g. *an operation x in a machine y*), which is established at least in the process plan for a particular type of product.

<sup>80</sup> In ANALYTICE II, the Instigations of Rule A.0 was replaced by the Instigation “Store3x3.1 create parts”, which also results in the state change of the Attributes related to the positions.

As a more specific example, a Rule could be created for reaching a production step < *operation into the Lathe.1* > defined in the process plan for products of type V2. This Rule would allow each product of type V2 to perform that production step when it has finished the previous production step < *buffering on Table2P.1 Pos 2* >.

A first step necessary for creating a coordination Rule is to identify the activities involved in performing the relevant production step. In the case of the production step < *operation into the Lathe.1* >, as specified in the process plan for V2-type products, these activities would be:

- The transporting of the part from Table2P.1 Pos 2 to the Lathe.1 via ERIII.1<sup>81</sup>.
- The preparing of Lathe.1 for the machining of V2-type parts.

The identified activities serve as the basis for defining the decision part of the Rule. The decision composition is firstly implied in the definition of the appropriate Premises for evaluating the conditions necessary for executing the activities. The Premises for the current example should verify if:

- Table2P.3 has a product deposited, using the Attribute Pos1 as *Reference*.
- ERIII.1 is free, using the Attribute Status as *Reference*.
- Lathe.1 is free, using the Attribute Status as *Reference*.

After the Premises have been defined, a Condition needs to be created whereby these Premises are connected. Once the decisional part is elaborated, the respective conclusion should be elaborated, also based on the activities previously identified. The conclusion composition is firstly implied in the definition of the appropriate Instigations. The Instigations for the considered example should request the following services:

- ERRIII.1 Transport Part from Table2P.3 Pos1 to Lathe.1.
- Lathe.1 Prepare to Receive Part of kind V2.

Following the definitions of the Instigations, an Action needs to be created whereby these Instigations are connected. Finally, the Rule aggregating the Condition and the Action must be created. In fact, the Rule for the example considered would be the same *Rule E* previously presented in Figure 38.

For the Condition composition, the determination needs to be made of whether the needed Premise already exists in the holarchy. If it already exists, it is connected to the Condition; otherwise it is created, connected to the Condition, and taken into account by the holarchy.

For the Action composition, the same idea is applied over Instigations. This technique permits the sharing of information and collaborations, which contributes in the dynamics of the holons interaction (i.e. the inference process) as presented in section 4.7.

All coordination Rules are created by following a creation process such as that outlined above, which includes the steps described above. Although the Rule-creation process outlined above is not an automated one, it is potentially realizable in a more automated manner (due to its organization), e.g. by using artificial actors or agents as a technological means.

For a more automated creation process, the agents could be called Product-Type-HLs once they would work using the knowledge from process plan related to each product-type. In the

---

<sup>81</sup> Each Processing-HL (e.g. Machining-HL) or Storage-HL could store and communicate which are the Transport-HLs that can reach it, i.e. which are the Transport-HLs that are able to do related physical interaction with them for loading or unloading parts. Similarly, each Transport-HL could store and communicate which are the reachable Processing-HLs or Storage-HLs. This kind of information would be useful for construction of coordination Rules.



case of Rules created by Product-Type-HLs, the control system expertise would be directly related to the skills of these agents for creating the Rules.

Another activity involved with Rule creation is the construction of the inference net (related to the evaluation of the Rules). This activity was effectively automated. This automation is related to the creation of the Premises and to the creation of the Conditions.

In the creation of a Premise, at least one Attribute is considered as its *Reference*. Once an Attribute has been referenced in a Premise, this Premise is automatically considered by the Attribute as interested in its state. In this way, the Attribute can determine which Premises are to be considered as interested, and notify each of them when the Attribute's state changes.

Similarly, when a Premise is connected to a Condition, the Premise automatically considers this Condition as interested in receiving notifications about its state. In this way, the Premise can identify all Conditions considered as interested, and notify them when its state changes.

This awareness that entities possess of other interested entities provides the basis for carrying out the inference process by means of notification. The notification net, which allows carry out the inference process, emerges in the control creation

In the proposed inference process, the inference information is taken from the formation of the Rules, rather than being discovered at execution time, as would be the case with inference processes based on search methods<sup>82</sup>.

#### **4.11. Rules and Agility**

At this point in the thesis, a solution to the composition of an architecture for the process-driven holonic control has been proposed. Both details and an application example of the proposed architecture have been presented. Some correctness features (such as performance and deadlock mechanisms) and holonic features (such as decoupled entities and human-friendly solutions) have also been discussed.

An important point about the proposed solution that still remains to be discussed is the topic of *agility*. A manufacturing system (MS) is considered as *agile* if response time is low in relation to the production demands. This thesis considers agility in relation to *production adaptability*, which allows the MS to maintain its agility in the face of changes.

*Production adaptability* refers to the ability to exploit flexibilities enabled in the MS. The flexibility could be exploited, for example, in order to optimize production, to deal with variable production, or to better tolerate system faults. Within the area of MS, it is possible to find different kinds of flexibilities. For instance, it is possible to have buffering flexibility, transport flexibility, or processing flexibility(Hartley, 1984).

The buffering flexibility and transport flexibility are concerned respectively with use of alternative buffering places and transport routes. Processing flexibility is related to use of alternative resources for processing and/or the ability to use these resources to produce different types of products (Carvalho, 2003).

Other, less evident, kinds of flexibility are also possible, such as flow flexibility. In the *scope* of flow flexibility, for example, the flow of production can be planned by grouping similar products and by launching them at on flexible dates, depending on a variable set of criteria, such as resource settings and production priorities (Bongaerts, 1998)(Sautter, 1991).

---

<sup>82</sup> Search methods are summarized, for example, in (Rich et Knight, 1991)(Russell et Norvig, 2003).

Production agility in the proposed solution is dependent on the knowledge “expressed” within the Rules. This knowledge is a factor in determining the degree of exploitation of flexibility (i.e. adaptability) possessed by the MS. In fact, agility is dependent on the knowledge (about the MS) of the set of experts who construct the Rules.

For example, a set of Rules could specify that all products of type V1 (between Pos 1 and Pos 6) must be removed from the Store before products of type V2 (between Pos 7 and Pos 9) are removed, which would be an alternative to the previously proposed approach (in section 4.9.3) to removing products. However, this would generate the worst possible use of the parallelism in that MS. The original set of Rules in section 4.9.3 allows a more equilibrated product removal, and production via an appropriate use of the available flow flexibility.

As the production agility depends on the knowledge of Rules, Rules can be improved to support changes in the system, with the goal of achieving adaptability while maintaining agility. An example of an improvement possible within the Rule knowledge is in the case of an introduction of a third product type (V1.2) from the same family as V1, to be machined by Machine-Tool.1.

V1.2 products could be placed between Pos 4 and Pos 6 of the Store, and then V1 products would only be placed between Pos 1 and Pos 3. Therefore, a process plan would be generated for products of type V1.2 and the process plan for products of type V1 would be changed. These process plans would be, respectively:

- { < Store [ pos 4, 5, or 6 ] > < Table 1 [ pos 2 ] > < Machine-Tool > < Table 2 [ pos 1 or 2 ] > }
- { < Store [ pos 1, 2, or 3 ] > < Table 1 [ pos 2 ] > < Machine-Tool > < Table 2 [ pos 1 or 2 ] > }

For supporting the production of parts of type V1.2, some rules could be modified, as proposed in Figure 40, with the aim of better use of the flexibilities allowed by the Resource-HL features. The changes in the Rules would be based on the defined process plan.

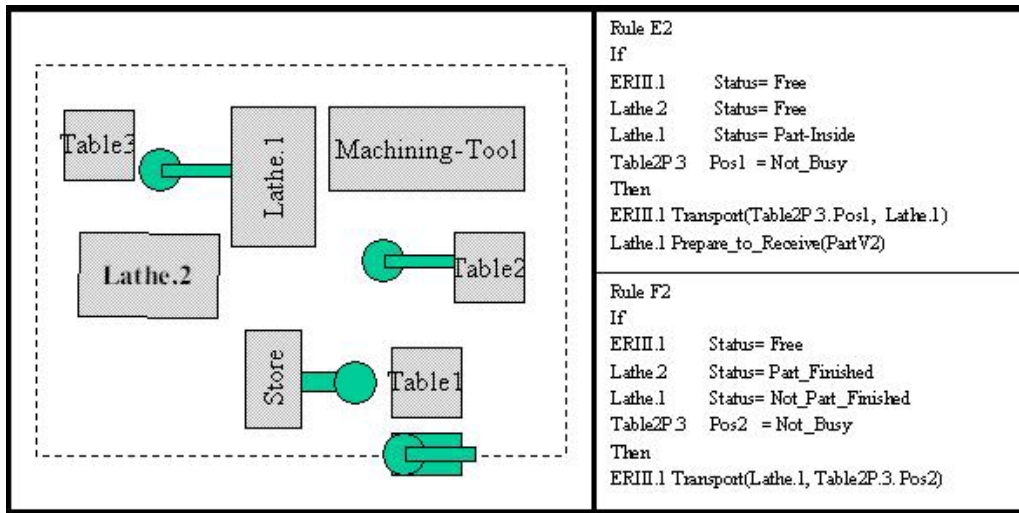
<p>Rule A.1 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Busy Then Puma560.1 Transport(Store3x3.1. Pos1, Table2P.1. Pos1 ) <b>Table2P.1 SetActualTypePart(V1)</b></p>	<p>Rule A.2 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Busy Then Puma560.1 Transport(Store3x3.1. Pos2 Table2P.1. Pos1 ) <b>Table2P.1 SetActualTypePart(V1)</b></p>	<p>Rule A.5 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Not_Busy Store3x3.1 Pos4 = Not_Busy Store3x3.1 Pos5 = Busy Store3x3.1 Pos7 = Not_Busy Store3x3.1 Pos8 = Not_Busy Then Puma560.1 Transport(Store3x3.1. Pos1, Table2P.1. Pos1) <b>Table2P.1 SetActualTypePart(V1.2)</b></p>
<p>Rule A.3 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Busy Store3x3.1 Pos7 = Not_Busy Then Puma560.1 Transport(Store3x3.1. Pos3, Table2P.1. Pos1 ) <b>Table2P.1 SetActualTypePart(V1)</b></p>	<p>Rule A.4 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Not_Busy Store3x3.1 Pos4 = Busy Store3x3.1 Pos7 = Not_Busy Then Puma560.1 Transport(Store3x3.1. Pos4, Table2P.1. Pos1) <b>Table2P.1 SetActualTypePart(V1.2)</b></p>	<p>Rule A.6 If Puma560.1 Status= Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Pos1 = Not_Busy Store3x3.1 Pos2 = Not_Busy Store3x3.1 Pos3 = Not_Busy Store3x3.1 Pos4 = Not_Busy Store3x3.1 Pos5 = Not_Busy Store3x3.1 Pos6 = Busy Store3x3.1 Pos7 = Not_Busy Store3x3.1 Pos8 = Not_Busy Then Puma560.1 Transport(Store3x3.1. Pos6, Table2P.1. Pos1) <b>Table2P.1 SetActualTypePart(V1.2)</b></p>
<p>Rule B.1 If KUKA386.1 Status = Free Puma560.1 Status = Free Machine-Tool1 Status = Free Table2P.1 Pos1 = Not_Busy <b>Table2P.1 TypePart = V1</b> Then KUKA386.1 Transport(Table2P.1 Pos1, Machine-Tool1) Machine-Tool1 Prepare_to_Receive(PartV1)</p>	<p>Rule B.2 If KUKA386.1 Status = Free Puma560.1 Status = Free Machine-Tool1 Status = Free Table2P.1 Pos1 = Not_Busy <b>Table2P.1 TypePart = V1.2</b> Then KUKA386.1 Transport(Table2P.1 Pos1, Machine-Tool1) <b>Machine-Tool1 Prepare_to_Receive(PartV1.2)</b></p>	

Figure 40: Rules improved to support a product variant.

Basically, in this case, more information about the process is included within the Rules, allowing them to carry out an adapted control, for exploiting possible flexibilities in the MS. A human actor, or even an artificial actor, could carry out the changes in the Rules. The main point is that the adaptation basically requires handling information and it is easily added to control entities at a high level in the system.

Another example of improvement of Rule knowledge within the control system is in the case where the system is enhanced with a second Lathe. This Lathe would be reachable by ERRIII, as presented on the left-side of Figure 41. In this case, the system would have more production flexibility, for example, useful for the machining of products of type V2. Thus, it would be necessary to construct other Rules to exploit this flexibility. These Rules are depicted on the right-side of Figure 41.

It is believed that some holons with specific knowledge and algorithms could automatically generate or change some types of Rules<sup>83</sup>. The Product-Type-HLs considered in the previous section is an example of this type of holon. These holons would serve to automatically achieve some adaptability for dealing with changes, trying to maintain or improve production agility.



**Figure 41: Rules to deal with more a Lathe-HL in the Environment.**

Beyond adding appropriate knowledge to the Rules, other means can be investigated for flexibility exploitation. One way that could be investigated is the use of other entities to activate/deactivate Rules or establish priorities in the Rules, if it is considered that a conflict solver has been established. For example:

- If only parts of type V1 and V1.2 must be produced, the Rules concerning parts of type V2 could be deactivated.
- If parts of type V1.2 must be produced with priority, the concerned Rules for taking them in the Store would have their priorities augmented.

This type of manipulation of the Rules is also understood as adaptation with the aim of increasing agility. The manipulation of the Rules could be carried out by other holons, such as holons related to flow management, which would know which products must be produced and in which priority.

<sup>83</sup> *Rules of formation*, with respect to classes of Rules, is discussed in (Simão et al. III, 2001)(Simão et Stadzisz, 2002)(Simão et al. II, 2003). These *Rules of formation* has been defined for searching certain automatism to the composition of Rules

## 4.12. Considerations

Rule Based System (RBS) and Expert System (ES) have inspired the solution to holonic control proposed in this thesis. In a holonic point of view, a useful property of a control such as an RBS is certain human integration since causal rules are a natural manner of human thinking. Additionally, in an appropriate user-friendly environment, experts could compose Rules expressing the appropriate knowledge without caring about a set of details in the informatics structure and dynamics generated in background.

In the proposed approach, control agility can be achieved by including appropriate knowledge in the Rules and by appropriately manipulating the Rules with respect to changes in their priorities or *validities*. Both Rule creation and Rule manipulation could potentially be performed by other holons playing the role of experts. These holons would be from other decisional systems, for envisaging a systemic integration

In order to achieve a rule-based solution to holonic control, an alternative way to carry out the inference process was proposed. The proposed inference process is based on decoupled entities (agents) for the causal knowledge, on the sharing of inferred information about factual knowledge, and on a notification mechanism for the inference about the factual knowledge by the causal knowledge.

An agent called Rule treats each rule as a specific entity. A Rule has its knowledge organized using collaborator agents. The collaborators allow Rules to share some inferred information, avoiding redundancies in the inference process. Also, the inference process is carried out by the notification of states or facts about Resource-HLs to the Rules, with only interested Rules being notified, using a net of collaborators.

The notification-oriented inference process replaces the standard search methods to reach the awareness of facts in the rules. The coupling of rules in the inference process carried out by searches methods can, for example, complicate the openness of the control system to reach distributed implementation, which is an aim in holonic control systems.

A Holonic Supervisory Control (HSC) was implemented using this proposed approach and executed using ANALYTICE II with holonified resources. The Rules and collaborators presented in this chapter were implemented for constructing this HSC and variants thereof. The HSC and its variants are functional for permitting the controlled cooperation of Resource-HLs by means of decoupled Rules.

The Rules were, in fact, implemented by a set of collaborative and connected entities, which cooperate by means of a notification mechanism, as specified. The awareness process of facts in the Rules has effectively happened via this notification mechanism, which allowed reaching deliberations in Condition parts and achieving consequent conclusions *expressed* as Action activations.

The architecture was implemented in a non-distributed way. However, it may also be applied within a distributed system because of the decoupling between Rules and their decoupling from the fact base, made possible by interaction via notifications. Independently of the implementation (i.e. distributed or not), only concerned entities are notified when a fact is changed. The behavior of a non-distributed implementation can be viewed as very similar to a distributed implementation (i.e. it *imitates* the distribute behavior).

The possibility of distributed and non-distributed implementations of the proposed holonic control solution is discussed in the next chapter. Anyway, the inference process adopted in this solution results in an inference process with very low computational complexity (a

concern in non-distributed implementations) and in the avoiding of unnecessary communication (a concern in distributed implementations).

This inference process optimizing notification is agreed with correctness features because it allows the system to be highly reactive. However, other questions about correctness still need to be dealt with. In this respect, mechanisms to solve determinism and conflict issues are discussed in the next chapter. At the moment, the determinism and conflicts of the Rules has been handled by means of interlocked Rules.

Another feature that is part of the criteria for the proposed solution is generality respecting certain applicability easiness. In reality, this solution is understood as generic due to the similarity or even fractal property (self-similarity) observed in the implemented instances, e.g. similar Rules and (self) similar Resource-HLs. Additionally, the Rule model in particular presents a trade-off between generality and easy applicability, which allows direct instantiation.

In this sense, the Rules comprising similar Conditions and Actions, each with respective similar collaborators, have been considered a pattern to decision and coordination in this control solution. Monitoring and command have also been patterned via Attributes and Methods of Resource-HLs (Simão et al., 2002). These two patterns are understood as a major computational pattern, an architectural pattern, for discrete control systems (Simão et Stadzisz, 2002)(Stadzisz et al., 2003).

Actually, in a more general sense, changing rules or instigating changes of facts is known as a general solution to much kind of controls in computer science literature, namely in the artificial intelligence area. The main difference between the approach proposed here and the approach discussed in the computer science (or artificial intelligence) literature is in approach proposed here both rules and correlated facts are effectively encapsulated in entities (agents) that collaborate to carry out a differenced inference process.

In point of fact, the key feature of the approach presented here is to define a differentiated and efficient inference process carried out by communicable control entities (inspired by RBS), and apply this approach to the handling of the causal relations of control over homogenized Resource-HLs (tested within a realistic simulation tool), as a means to achieve holonic control solution with openness to correctness features.

The next chapter explores solution properties related to holonics and correctness issues. It also considers other improvements that can help achieve the goals related to holonics and correctness. The subsequent chapter describes how to improve process-control solution described here by transforming it into an order-driven or product-driven.



## **Chapter 5**

---

Holonic Control





# Chapter 5 : Holonic Control

## 5.1. Introduction

The control solution proposed in this doctoral thesis permits to express the causal knowledge of control instances by means of rules, such as those used in Rule Based System (RBS). Thus, the essence of the control solution is related to the informational structure for handling the rule knowledge and to the adopted inference process. In fact, as previously presented, these concerns are related to the correctness and holonic features desired for the control solution.

In this chapter, the first section presents the structure of the *production system*<sup>84</sup> as the essence of RBS. This allows effectively understanding the architecture proposed for control as a type of RBS architecture, as well as the reasons that explain why standard solutions to inference process, for composing Inference Engine (IE), do not fit well in the objectives of this thesis.

In this thesis, a particular technique was defined to the inference process (in the scope of the proposed architecture) that can be considered a type of IE technique. This solution to inference process, based on notification, presents a set of properties useful for reaching holonic control systems comprising correctness features. The subsequent sections in this chapter are concerned to the attainability of these features, which are:

- The possibility of distributed, non-distributed, and mixed implementation thanks to the entity decoupling and low computational complexity of the inference process.
- The conflict identification and resolution, as well as the trade-off between determinism and reactivity, via additional symbiotic mechanisms also based on notification.
- The feasibility in the use of discrete event control (DEC) formalism, namely Petri nets (PN) that are compatible with RBS.

The main goal in this chapter is to present the openness of the proposed generic control architecture to the composition of proper instances of holonic control over Resource-HLs, like those simulated in ANALYTICE II. Actually, it is aimed to present that implicit properties of the architecture combined with additional mechanisms and formalisms allow composing holonic control with correctness features.

## 5.2. Rule Based Systems

### 5.2.1. Introduction

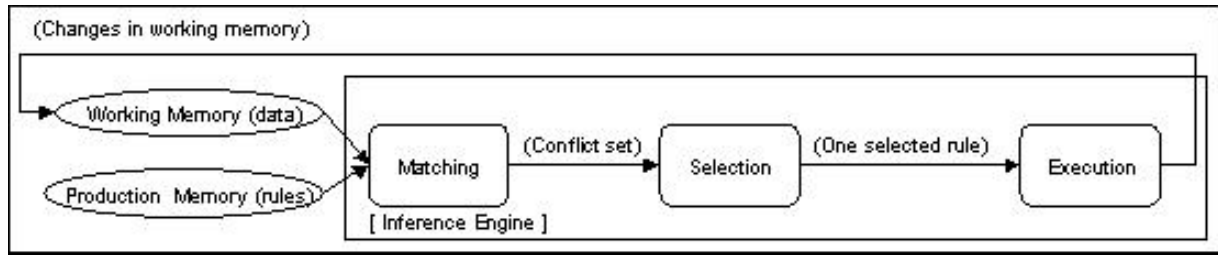
A lot of large-scale Expert System (ES), such as the CLIPS system from NASA, are founded on the concept of Production Systems (PS) from computer science (Pan et al., 1998)(Simão et Stadzisz, 2002). A common composition of PS is:

- A Working Memory (WM) for data or facts.
- A Production Memory (PM) for rules.
- An Inference Engine (IE) for inference process.

---

<sup>84</sup> Production System concerned to Production Rules in Computer Science.

An IE infers new facts (*productions*), by the matching of rules and existing facts, asserting them into the WM. The new facts can allow new inferences, and this cycle will finish when no further facts can be matched. The matching can be forward (i.e. to search the conclusion) or backward (i.e. to search the condition) (Jackson, 1990).



**Figure 42: The Cycle Operation of Inference Engine in a Block Diagram.**

An ES or RBS can be considered as a PS due to the operation cycle of its IE. This cycle consists of three steps called *matching*, *selection*, and *execution*, as presented in Figure 42 (Pan et al., 1998)(Rich et Knight, 1991)(Simão et Stadzisz, 2002). These steps are the nutshell of a PS:

- The *matching* step propagates the latest facts of the WM and finds which rules are enabled determining a conflict set, also called *agenda*.
- The *selection* step does the ordering of the rules in the agenda, using some parameters, and the rules with better punctuation are, in order, selected from the agenda<sup>85</sup>.
- The *execution* carries out the actions of the selected rules. These actions may either assert the new inferred facts into the WM or invoke functions.

Each instance of the architecture proposed in this work is also a kind of RBS due to their dynamics that presents matching, selection, and execution steps. However, the notification-based implementation differs from common implementations of IEs.

### 5.2.2. Inference Engine

An alternative inference-process solution could be used instead of the notification-based solution proposed in this thesis. An alternative solution could be one of the existing common policies for IE. In this respect, some policies had been established upon search methods, where the computational complexity has been a concern (Rich et Knight, 1991)(Russell et Norvig, 1995)(Russell et Norvig, 2003).

In fact, for RBS application in large-scale ES, it was needed advanced policies to build IE avoiding exponential explosion, like the very used Rete networks proposed by Forgy (Forgy, 1982). Rete (*net* from Latin) is a reference in advanced IE and it has been used in traditional ESs to improve the computational capacity of matching rules and facts (Pan et al., 1998)(Rich et Knight, 1991)(Russell et Norvig, 1995).

A Rete network eliminates the temporal redundancy, while the structural redundancy is eliminated for simple premises, but not for compound premises (Pan et al., 1998). In a simple premise an “attribute of object” is compared with a constant, while in a compound premise two “attributes of objects” are correlated by comparison (Simão et Stadzisz, 2002), as exemplified in Figure 43.

<sup>85</sup> Examples of parameters to order rules are the last date in which the rule was fired, the recency of concerned facts from the WM or the simplicity/complexity of the rule (Pan et al., 1998).

The Rete algorithm implements an optimized search process, placing all the components of rules in a tree-structured sorting network used to feature tests (Russell et Norvig, 1995). The kernel of many expert system frameworks designed to handle hundreds or thousands of rules (e.g. ART, CLIPS, OPS83, and ILOG Rules) use that technique or a variant thereof, because it has only a polynomial computational complexity (Pan et al., 1998).

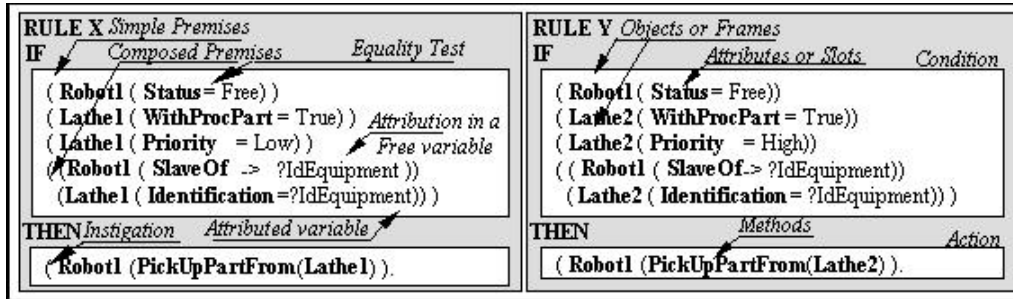


Figure 43: Examples of rules with simple and compound premises.

### 5.2.3. Avoiding Searches

This doctoral thesis envisages an agile control solution to HMS. Agile controls, such as holonic controls, are potentially distributed for agile reasons (Wyns, 1999). In fact, the control distribution is motivated by some advantages commonly referenced in distributed computational systems (Couloris et al., 2001), which are useful to keep the agility of the system. An example of distributed control advantage is the avoiding of the system breakdown even when part of the control system is not active any more.

The Figure 44 presents the case study from the previous chapter with Rules designed in a physical distributed way. Each set of grouped Rules is called a Control Island. In this case, for instance, if Island 4 breaks down, the production of parts V.1 could still be made, during certain time, because Rules E and F (on the Island 4) are not directly implied in the production of this type of part<sup>86</sup>.

The distribution of a computational system also allows other benefits, as certain robustness for fault treatment by means of the redundancy of entities (Couloris et al., 2001). For example, the Rules designed in Figure 44 could be replicated or *cloned* in others physical places aiming to replace the main or *matrix* Rules in inoperable state, for keeping the system operability.

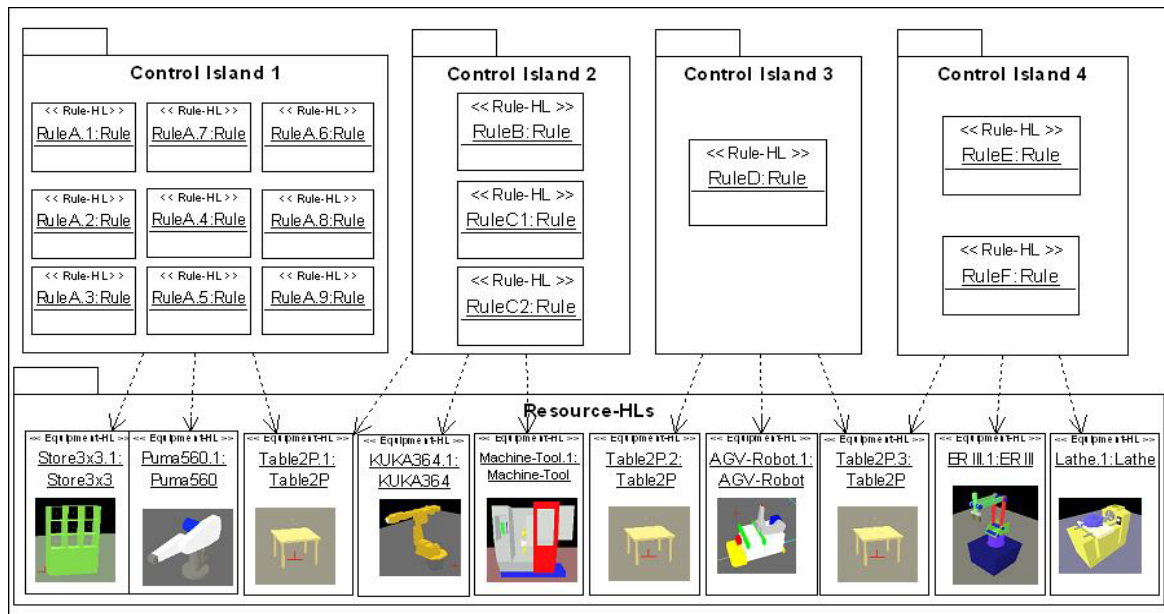
Hypothetically, each Rule *clone* would be synchronized with the *matrix* Rule and it would have its Action part completely deactivated. However, the Action would be activated when the Rule *clone* perceives that its *matrix* Rule is unexpectedly *dead*. Therefore, *clones* could keep the control functionalities when the *matrixes* are not functional.

In short, the implementation of a rule-driven control as an agile one should comprise an inference policy facilitating the distribution of control entities. However, this policy should also allow mixed implementation comprising non-distributed implementations because the agile control is not necessarily distributed in a complete manner.

The usual methods applied to carry out the Inference Engine (IE) in Rule Based Systems (RBS) use some kind of search to match *Rules* and *Facts*, even in the Rete case. However, it

<sup>86</sup> In the case that the Control Island 4 breaks down, to allow the uninterrupted production of parts V1, it would be only necessary to reorganize the Rules that transport parts from Store3x3.1 to Table2P.1.

would not be interesting an inference process carried out by searches in a rule-oriented control for HMS, which is potentially distributed.



**Figure 44: Distributed Rules on Control Islands.**

Actually, at least the base of facts (e.g. resources states) is implicitly distributed in Manufacturing System (MS). Searches in a distributed base of facts can be more complicated than in a centralized one. For example, depending on the implementation, some drawbacks could be:

- The increase in the time of inference cycle due to factors such as the time used to exchange information between network points and even to deal with failures in the communication.
- The increase in the network traffic in relation to the computational complexity of the search method used to the inference process.

Additionally, in the case of effectively distributed rules, it would be more difficult to carry out the inference process by means of search methods. In this case, for example, it would be also necessary to guarantee the coherence in the evaluation and changing of facts for all rules in the distributed environment<sup>87</sup>. This would imply in a type of IE that should also make searches in the distributed rule base, for aiming at suitable knowledge for guaranteeing the coherence.

Similarly, the implementation of robustness features such as the rule redundancy, where main and redundant rules must be synchronized, would be difficult in an environment with distributed rules inferred by search methods. Furthermore, other issues like load balancing (related to communication and processing performance) could be also hard once concerned rules should be placed together for facilitating the coherence in the inference process<sup>88</sup>.

<sup>87</sup> The coherence could be, for instance, the possibility of all rules concerned to a determined fact to have the same opportunity to evaluate its achievement.

<sup>88</sup> A way to support distribution could be all facts, about resources, be sent to a base of fact (as a blackboard) and rules be matched with the facts therein, using advanced techniques of IE (such as Rete). Furthermore, this control could be composed of a set of sub-ES in the same way, avoiding centralized control. However, also in this case, some problems remain even if factored. An example of problem is the potential inconsistency of changed facts that are related to many sub-ES. Another example of problem is the co-existence dependency or coupling of related rules in a same sub-ES. These problems cause (for instance) difficulties to on-line load balancing and to rule clones effectively synchronized. Summarily, the difficulty is in the distribution of the ES in sub-ES, especially hard when facts and rules can be interesting to many sub-ES.

The inference process defined in a generic architecture for holonic control should avoid search methods. The generic architecture desired to agile control must be open to distribution. This argument is a first motivation to propose an alternative inference process based on notification. Using notification, searches in a set of data (like lists or trees, or even distributed nodes) are not needed.

In the proposed approach, entities straightforwardly notify other interested entities, enabling more functional independence and reducing the overall need for communication between entities, which are profitable features in distributed systems. Furthermore, the notification of one entity to another does not care about the localization of the destination (i.e. local or remote) because the main issue is to know its contact address. Therefore, the inference by notification fits well from non-distributed to distributed implementations<sup>89</sup>.

Additionally, it is foreseen that an inference process founded in notification mechanism, without temporal and structural redundancies, may facilitate the proposition of control solutions that take into account conflict identification and resolution, reactivity versus determinism, low computational complexity, balancing of entity distribution, and entity redundancy.

### 5.2.4. Notification Principle

The notification concept was a key element to the development of the proposed inference process where both rules and correlated facts are effectively computational entities. In the case of facts, facts extremely correlated (e.g. mutual exclusive boolean values or many states that are mutually exclusive) are encapsulated in and handled by an agent called Attribute. These facts, states or values related to an Attribute define its possible discrete states, such as generically presented in Figure 45.

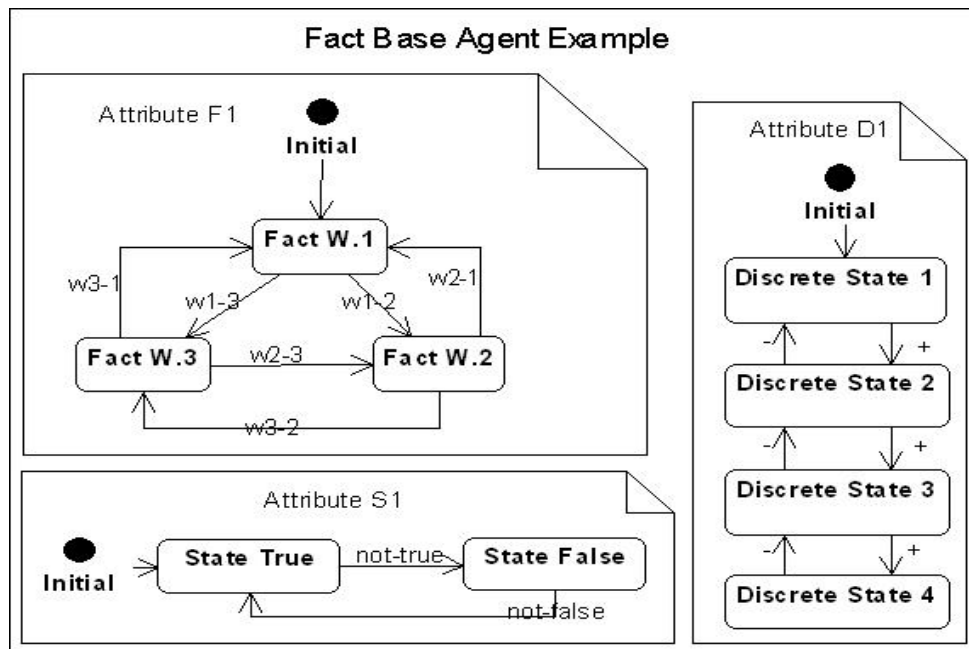


Figure 45: Generic examples of Attributes inside of a FBA.

A set of Attributes that makes reference to a same entity, in an observed system, can be, in turn, aggregated into a **Fact Base Agent (FBA)** by becoming its Attributes. For example, in

<sup>89</sup> In the case of distributed control implementation, common technologies like CORBA can be used. This type of technology abstracts the technical distribution layer, like the exchanging of messages between objects.

the case of an industrial plant over supervision and control, all Attributes related to a specific machine (e.g. status and temperature Attributes) would be included within a specific FBA (Simão et Stadzisz, 2002). In fact, the FBA is a generalization of the Resource-HL *concept* made to present the solution in a more generic manner.

As previously defined, the Attribute states are the facts observed via notification by the entities that handle the rules, i.e. the Rules and their collaborators. In short, Attributes notify Premises and Premises notify Conditions of Rules as instantiated in Figure 46. When a Condition of a Rule is proved, the respective Action can be fired. When the Action is fired, its concerned Instigations are activated. The related Instigations may instigate Methods that may directly or indirectly change the Attribute states (Simão et Stadzisz, 2002).

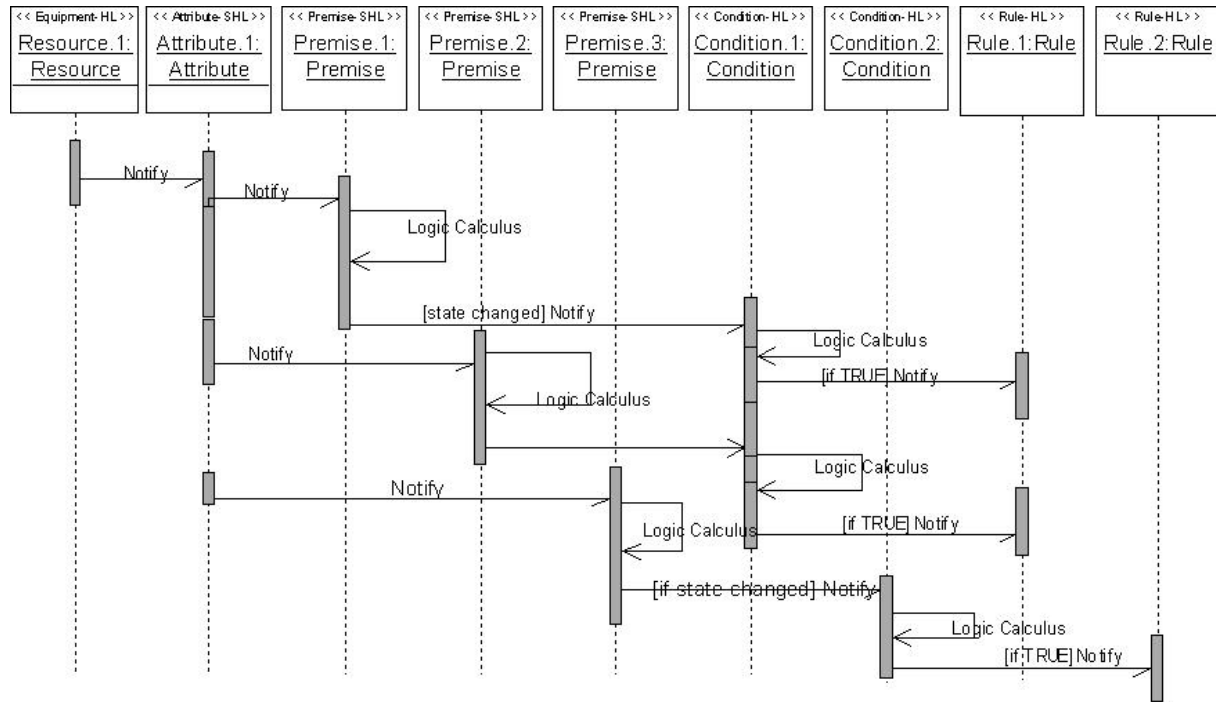


Figure 46: Sequence diagram to exemplify a notification chain.

In the Expert System point of view, the inference process is carried out by the collaborative interrelation of agents in a connection graph with shared information and based on notification, as presented in section 4.7. The notification graph avoids temporal redundancy, while the sharing of agent information avoids the structural redundancy in both simple and compound Premises<sup>90</sup>.

The proposed inference process is a quick one because it does not exist searches (e.g. in lists or trees), but messages propagation from Attributes, through Premises, to Rules, reaching Actions when possible. A new inference takes place only when a new fact occurs and only the *Premises* and *Conditions* affected are reevaluated. All the previous logical calculus remains valid (Simão et Stadzisz, 2002).

In this inference process approach, the concerned elements are functionally independent agents. This independency is due to the notification mechanism, which also allows an implicit openness of the solution to distributed implementations. Figure 47 presents the classes of related elements that carry out the inference process based on notifications. Effectively, these classes are presented in a more detailed way than in previous chapter. In short, they are the kernel of the proposed control architecture.

<sup>90</sup> The Structural redundancy is completely avoided because each Premise can reference two Attributes.

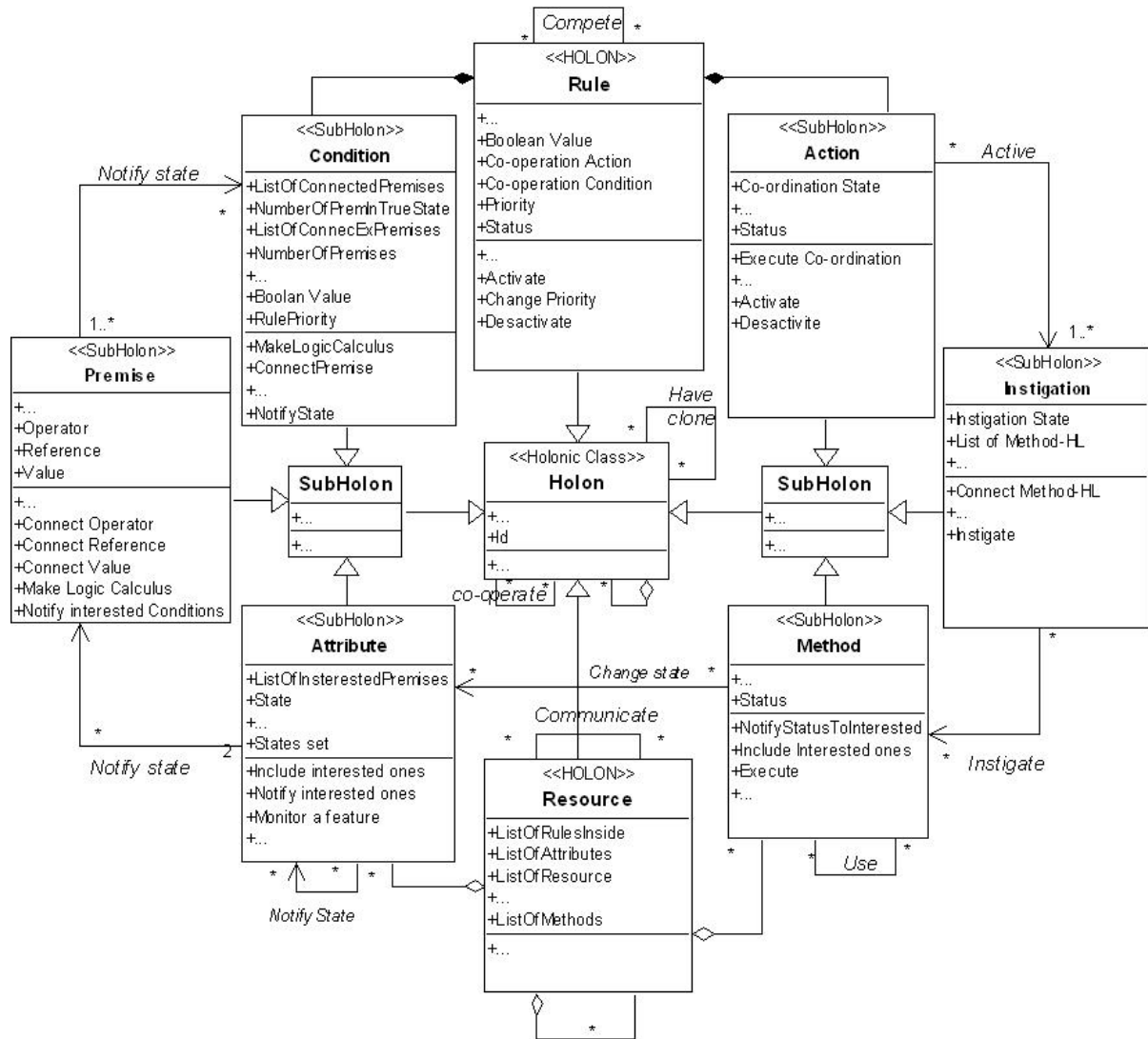


Figure 47: Main classes in the control architecture.

### 5.3. Computational Complexity

#### 5.3.1. Introduction

The proposed rule and agent-oriented architecture for control is *conceptually distributed* in the sense that each agent has its own behavior and communicates by notification in a decoupled way. This *conceptually distributed* architecture can be implemented from a real distributed way (distributed-objects) to a non-distributed way (as a type of object-oriented algorithm) being, anyway, the performance of implemented systems a potential concern.

In the case that the architecture is implemented in a distributed way, the system performance is related to the number of communications carried out and to the degree of execution parallelism. The degree of execution parallelism is directly related to the real distribution degree of agents (objects-like), while the number of communication is optimized by definition within the proposed architecture. Actually, the inference process proposed in this architecture

(established upon a notification mechanism) is appropriate for distribution because only necessary communications are carried out<sup>91</sup>.

In the case that the architecture is implemented in a (total or partial) non-distributed way, it can be important to take into account the computational complexity to guarantee functionality and good reactivity, i.e. to guarantee an appropriate performance. In fact, it is expected that non-distributed implementation of the architecture be reactive since each virtual resource (i.e. Resource-HL soft part) handles and notifies only related facts and the associated notification mechanism generates low computational complexity.

The virtual resource, even within a total non-distributed implementation, receives the signal from a respective resource, via a network, and then sensitizes only the concerned Attributes. After that, each sensitized Attribute potentially sensitizes control entities, but only the appropriate ones using its *notification branches* in the *notifications graph*. It is expected that the computational complexity be a low one is this approach.

### 5.3.2. Asymptotic Analysis of the Complexity

The non-distributed implementation of the proposed architecture has only polynomial computational complexity to each cycle of inference, related to the number of notifications made, in the worst case<sup>92</sup>. This polynomial complexity can be demonstrated by means of a computational complexity analysis.

The computational complexity analysis of the architecture can be made using asymptotic analysis (Russell et Norvig, 1995). For carry out this analysis, it is necessary to take into account the number  $n$  of notifications between ‘Attributes and Premises’ and between ‘Premises and Rules’ in each inference cycle.

Only as reference, in the case of an algorithm implemented based on search mechanism, the analysis of the computational complexity, called  $O()$ , would be made taking into consideration the number  $n$  of comparisons made between ‘Attributes and Premises’ and ‘Premises and Rules’. In truth, to know the number of notifications implies to know the number of comparisons.

In the worst case of the proposed *algorithm*, all Attributes would notify all Premises and then all Premises would notify all Conditions of Rules. Therefore, the function of worst case  $T_{\text{worst}}(x) = x^3$ , where its parameter  $x$  is the number  $x$  of notifier elements and its result is the number  $n$  of notifications or comparisons. This implies an  $O(n^3)$  complexity to the decisional inference process proposed in the worst case.

However, in the application domain where this algorithm is used, i.e. manufacturing system control, this worst case is not realistic because, for example, all Attributes of Resource-HLs would be not connected to all Premises. Another more proper way to analyze the complexity is to consider the average case.

The complexity analysis to an average case is started analyzing the fire of a notification (or sensitization) of an Attribute in a general manner. The main variables concerned to an Attribute notification (due to its state changing) is done by the Equation 5.

---

<sup>91</sup> The avoidance of (structural and temporal) redundancies and some advances in the notification mechanism (presented in the following section) may enable interesting properties to the distributed control, e.g. the balancing of the entity distribution and entity redundancies to robustness features. These aspects are discussed in the subsection 5.4.4.

<sup>92</sup> An inference cycle consist in a certain number of notification that must be done and finished before to start another certain number of notification related to the changes determined by these previous notifications.



$$FB_{AT}() = NumPremises + NumRules$$

**Equation 5: Complexity in the Attribute notification.**

The  $NumPremises$  is the sum of connected Premises to that Attribute and  $NumRules$  is the sum of connected Rules to each Premise counted in  $NumPremises$ . Therefore, if it is simply considered each inference cycle as the sensitization of an Attribute, an average is possible:  $T_{medium}(x) = (FB_{AT.1}() + \dots + FB_{AT.w}()) / w$ , where  $w$  is the number of all existing Attributes<sup>93</sup>. The result of this average is a constant, which implies an  $O(n)$  complexity.

### 5.3.3. Complexity and Time

The complexity analysis in a non-distributed implementation can be more precise when considering other Attributes concerned in the moment of fire of each Attribute notification. The main variables concerned to the sensitization of an Attribute (AT) in an instant  $i$  are defined in the Equation 6.

$$F_{AT}(i) = NumPremises_{(i)} + NumRules_{(i)} + F_{AT}(i-1), \text{ with } i \geq 1$$

**Equation 6: Complexity in the Attribute notification taking into account the time.**

The constituents of the Equation 6 have the follow interpretation:

- $NumPremises_{(i)}$  is the sum of connected Premises to AT, at the instant of time  $i$ .
- $NumRules_{(i)}$  is the sum of connected Rules to each Premise counted in  $NumPremises_{(i)}$ , at that same instant of time  $i$ .
- $F_{AT}(i-1)$  is the amount of complexities about the Attributes sensitized before AT, respectively in the *instants* from  $i-1$  until  $0$ , that are not treat yet<sup>94</sup>.

The part  $F_{AT}()$  of the equation presents some particularities:

- $F_{AT}(0)$  can be interpreted as the remainder complexity of the Attribute AT being considered. Anyway, it is considered that  $F_{AT}(0) = NumPremises_{(0)} + NumRules_{(0)}$ , by comprising the worst case.
- $F_{AT}(i)$  complexity is only  $NumPremises_{(i)} + NumRules_{(i)}$  when there not exist Attributes sensitized before the AT sensitized at the instant  $i$  (or when the Attributes sensitized before AT are already dealt with).
- $F_{AT}(i-1)$  has a tendency to short average when the  $FB_{AT}()$  of Attributes have a short average<sup>95</sup>.
- $F_{AT}(i-1)$  have also a tendency to short average if the algorithm is applied to a context where the facts are changed in an independent and time-driven way, as in the case of resources in manufacturing system.

The number  $\beta$  of Premises and Rules non-concerned to the equation  $F_{AT}(i)$  is indifferent to the complexity in the state-changing treatment of the concerned sensitized Attribute at the instant  $i$ . The exposed equation let to present the solution proposed as a complexity to each Attribute

<sup>93</sup> It was established that in the sequential implementation an approved Rule were executed only when there was not Attribute sensitization anymore.

<sup>94</sup> The instants would be counted in  $i$  when Attribute notifications would be accumulated.

<sup>95</sup> It is considered that Rules are well formed, e.g. without infinite Rule approbation.

sensitization, at an instant  $i$ , only between  $(K + 0)$  and  $(K + \alpha)$ , with  $K = NumPremises_{(i)} + NumRules_{(i)}$  and  $\alpha = F_{AT}(n-1)$ , where  $\alpha$  has a tendency to short or even none complexity in the studied application domain.

## 5.4. Control Correctness

### 5.4.1. Introduction

At this moment, the control solution has been detailed. Some features of the generic architecture have been discussed, thereby the architecture present a *convergence* in direction to a control solution with holonic and correctness features, e.g. the openness of the solution to distribution due to decoupled entities and the low computational complexity within non-distributed implementations. However, there also are other issues to be discussed as the clear openness of the solution to implement mechanism for conflict identification and solving, reactivity versus determinism, and certain robustness features.

### 5.4.2. Conflict Issues

A relevant issue to be implicitly dealt with in the proposed control architecture is the conflict identification and resolution (Simão et Stadzisz, 2002). In fact, the conflict identification and resolution is a classical problem in the discrete control (Chaar et al., 1993)(Wyns, 1999). Basically, a conflict happens when two different activities depend on a same-shared resource that must be exclusively used.

In the holonic case, conflicts are related to shared Resource-HLs. An example of shared Resource-HL is a Robot-HL that can be useful to two distinct Resource-HL cooperation-activities, in different time slices. A more specific example of shared Resource-HL is the Puma560.1 that is useful to many distinct Resource-HL cooperation-activities as presented in section 4.9.

In the proposed control solution, each Resource-HL cooperation-activity is controlled by a specific Rule. Therefore, Rules for cooperation involving a shared Resource-HL are potentially conflicting ones. In the case that these Rules are not created in an interlocked way, it is necessary to define a mechanism to identify and solve conflicts between them.

For the development of mechanisms to identify and solve conflicts between Rules, it is defined a special type of Attribute for shared Resource-HLs. An Attribute of this special type, called an *Exclusive-Attribute*, is an *exclusivity expression* of the use of a shared Resource-HL. Also, an Exclusive-Attribute (e.g. a *Status Attribute*) can be the *Reference* of Premises<sup>96</sup>, being Premises with this *Reference* type considered *Exclusive-Premises*. These specialized types of Attributes and Premises are modeled in the class diagram in Figure 48.

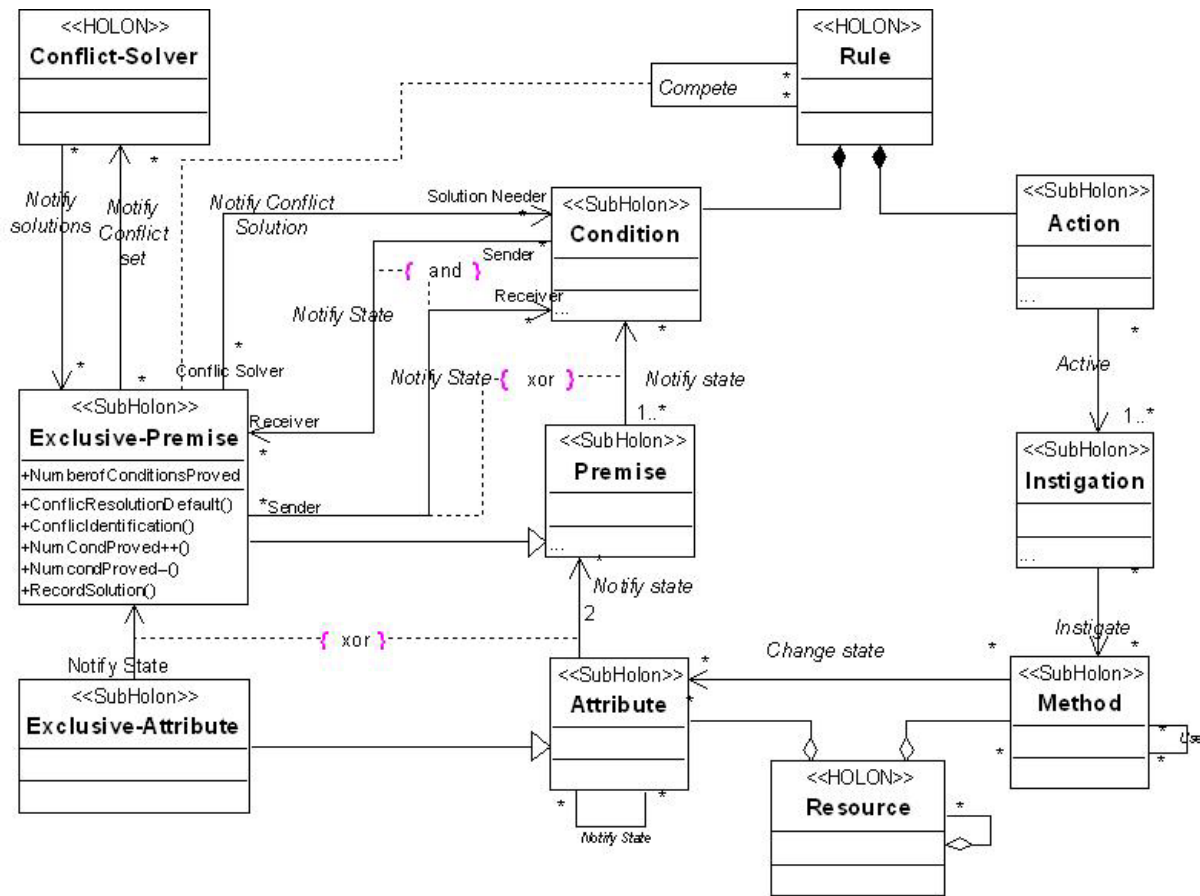
An Exclusive-Premise collaborates in the approbation of potential conflicting Rules and in the treatment of conflict issues between them (Simão et Stadzisz, 2002). In fact, a conflict is established when Conditions of Rules in *true* state have a same connected Exclusive-Premise (Simão et al. II, 2003).

In order to identify a conflict, each time that a Condition with connected Exclusive-Premise makes its logic calculus, it notifies the Exclusive-Premise about the reached state (i.e. *false* or *true*), as instanced in Figure 49 following the model presented in Figure 48. After each

---

<sup>96</sup> A Premise is composed of a *Reference*, an *Operator* and a *Value*, cf. defined in the section 4.6. The *Reference* of a Premise is an Attribute that notifies its state to that Premise.

notification about a Condition state, the Exclusive-Premise has a counter that is appropriately incremented or decremented. In short, if the state is *true* the counter is incremented else it is decremented.



**Figure 48: Class Diagram for Rule Conflict Identification and Solving.**

The counter represents the number of Conditions that the Exclusive-Premise has collaborated to be approved (Simão et Stadzisz, 2002). The Exclusive-Premise identifies a conflict state when the counter is greater than one. After that, the Exclusive-Premise can solve the conflict by itself or it can notify a Conflict-Solver-HL to solve the conflict (Simão et al. II, 2003).

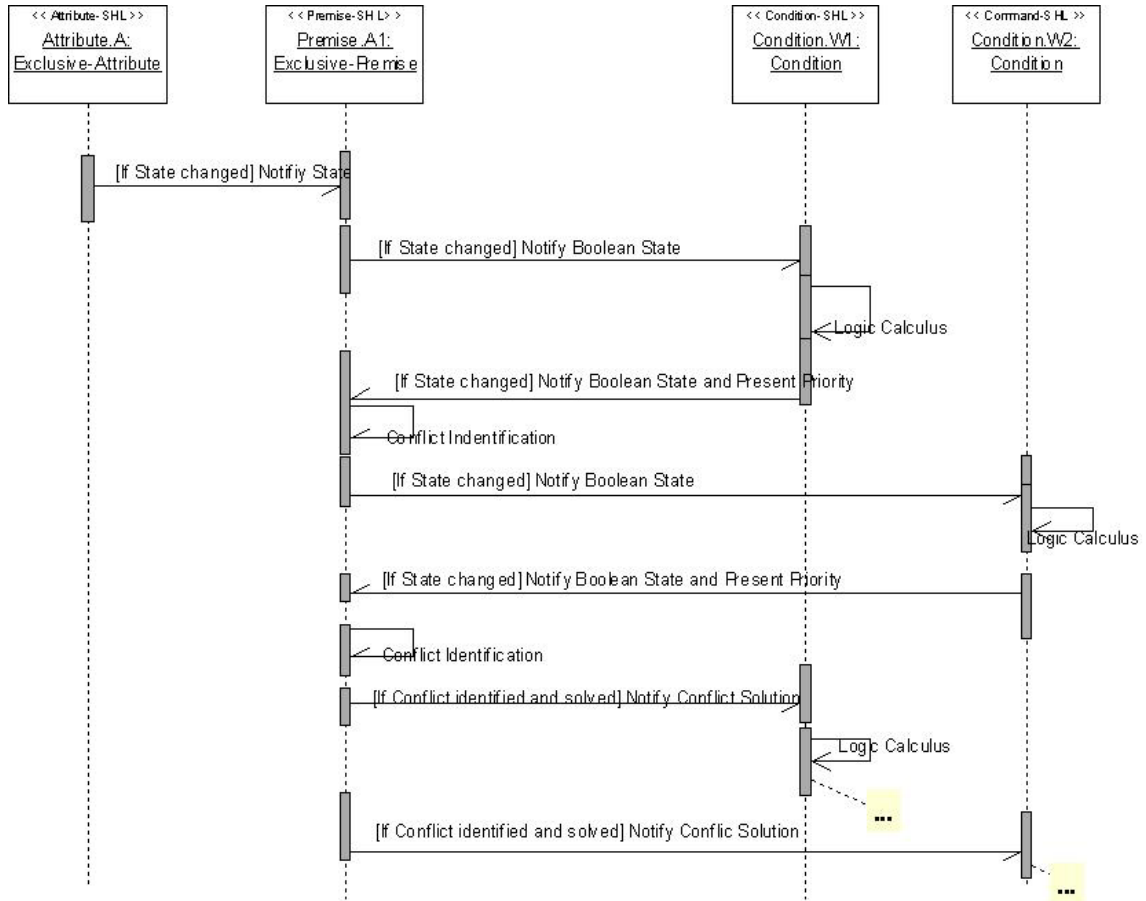
When the Exclusive-Premise solves the conflict by itself, it uses the Condition priority to make the decision<sup>97</sup>. If some ones of the Conditions have a same priority as the highest, a default policy is applied, such as a random choice of a Condition. However, a better idea would be to use a specific determinist policy to make the decision, e.g. first Condition that notified the Exclusive-Premise in a sequential implementation. Moreover, this decision could even be registered and reused in next time this same conflict happens, aiming to avoid indeterminism.

When the conflict is solved by means of a Conflict-Solver-HL, the Exclusive-Premise uses the solution proposed by the Conflict-Solver-HL to decide the conflict. The Conflict-Solver-HL could be, for example, a Holonic Flow Management (HFM) playing the role of an on-line scheduler. This allows certain systemic integration whereby other systems can interact in some HDCS decisions.

<sup>97</sup> The priorities of Rules can be defined in a way that avoids deadlock and allows productions with a desired behavior. Therefore, the priorities in the Rules are an alternative to interlocked Rules when a mechanism for conflict identification and resolution is present.

After the conflict solution, the selected Condition has the *true value* confirmed by the Exclusive-Premise and then being effectively approved, while other related Conditions have the *true value* disapproved from the Exclusive-Premise, and consequently they are disapproved too (Simão et Stadzisz, 2002)(Simão et al. II, 2003).

The proposed mechanism for conflict issues may be considered as appropriate one, firstly because it does not charge the inference process since it only generates a few additional notifications. Furthermore, the implementation of this mechanism can facilitate the control composition (e.g. avoidance of interlocked Rules) and the mechanism can be also emerged in the Rule creation (if the Attributes are identified as exclusive or non-exclusive).



**Figure 49: An instance of the dynamics during conflict identification and resolution.**

### 5.4.3. Reactivity and Determinism

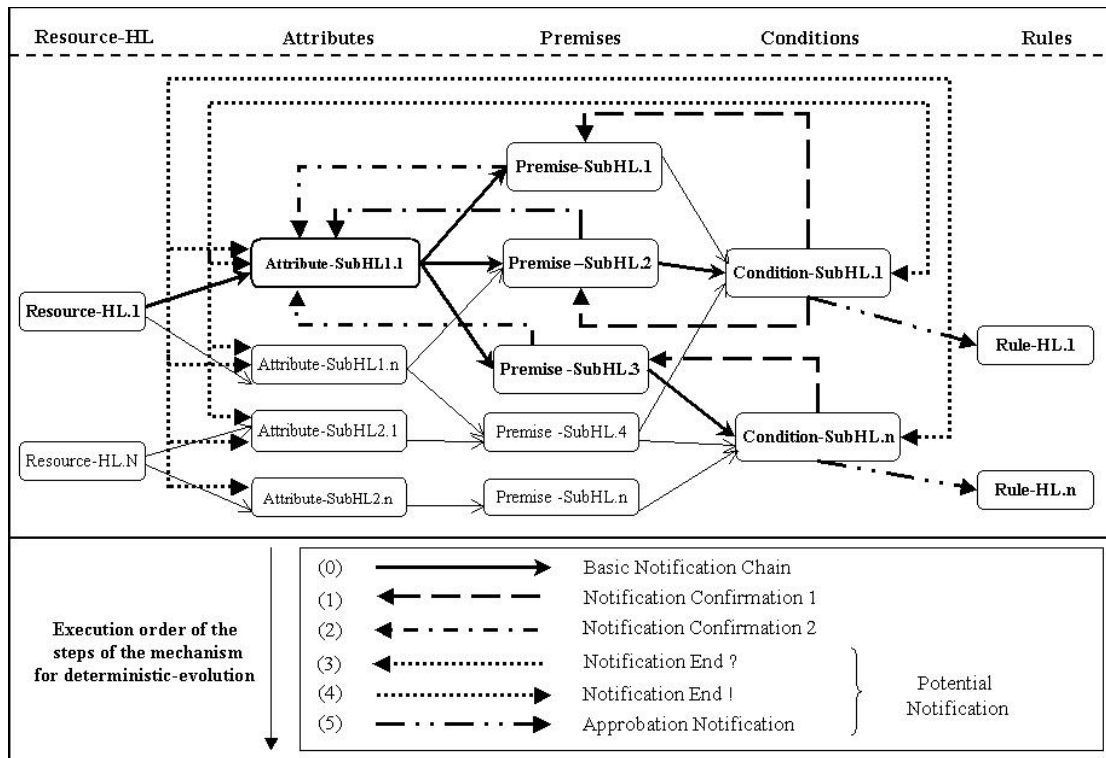
Beyond the conflict issues, another issue related to the carrying out of discrete control is to find a trade-off between reactivity and determinism (Lhoste et Pétin, 2001)(Pan et al., 1998). The reactivity is related to the control response time while the determinism is related to the control deterministic-evolution and deterministic-decision:

- The deterministic-evolution means that all entities of causal relation (e.g. Rules) must have the same opportunity to evaluate concerned changed states arriving at a deterministic-decision.
- The deterministic-decision means that decision taken must be the same in a deliberation involving the same evaluated information. The deterministic-decision is handled in the conflict resolution, for example, using the solution to conflicting Rules presented in the previous section.

In search-oriented algorithms, the trade-off between deterministic-evolution and reactivity is a problem because all possible affected causal relations should be found and evaluated, implying in time consuming to search and then less reactivity. In the proposed inference process, a low dependence between the reactivity and deterministic-evolution treatment can be reached because there are not searches but only notifications between directly concerned entities.

In this section, it is proposed a mechanism for deterministic-evolution in the proposed inference process where Attributes wait for notification confirmation from Premises and Premises wait for confirmation notification from Conditions (Simão et al. II, 2003)(Stadzisz et al., 2003). This mechanism concerns both distributed and sequential implementations and comprises a set of steps, based on notification, as exemplified in Figure 50. These steps are:

- (1) Each Condition that was notified confirms the receiving and using of the notified information to the notifier-Premise.
- (2) Each Premise that was notified confirms the receiving and using of the notification information to the notifier-Attribute.
- (3) Each approved Condition asks for each Attribute referenced in its collaborative Premises if it has finished its notification activity.
- (4) Each asked Attribute notifies the interested Conditions about its *notification end*.
- (5) Each approved Condition with all necessary confirmations about *notification end* notifies its respective Rule about its approbation.



**Figure 50: Representation of the Mechanism for Deterministic-Evolution.**

This mechanism gives the base to guarantee that all interested Rules are updated about new facts because the Conditions are forced to wait for all evaluations, as sketched in Figure 50 and instanced in Figure 51. Furthermore, this mechanism can be also emerged in the instantiation of the Rules, being its computational complexity (to each Attribute sensitization) increased only by some constants, still keeping the reactivity.

Also, additional improvements to solve specific questions are possible within this mechanism, such as for dealing with communication failures in distributed control implementation. In this case of communication failures, some policy can be foreseen, for example to notify again, to the case that any entity has not made its confirmation after a given time.

Another example of improvement in the treatment of communication failures could be an intermediary step before step 1. This step would be: (a) each Premise that was notified by the Attribute pre-confirms the notification; and (b) each Attribute (with pre-confirmations) re-notifies all Premises with a “flag” allowing their potential notification to Conditions.

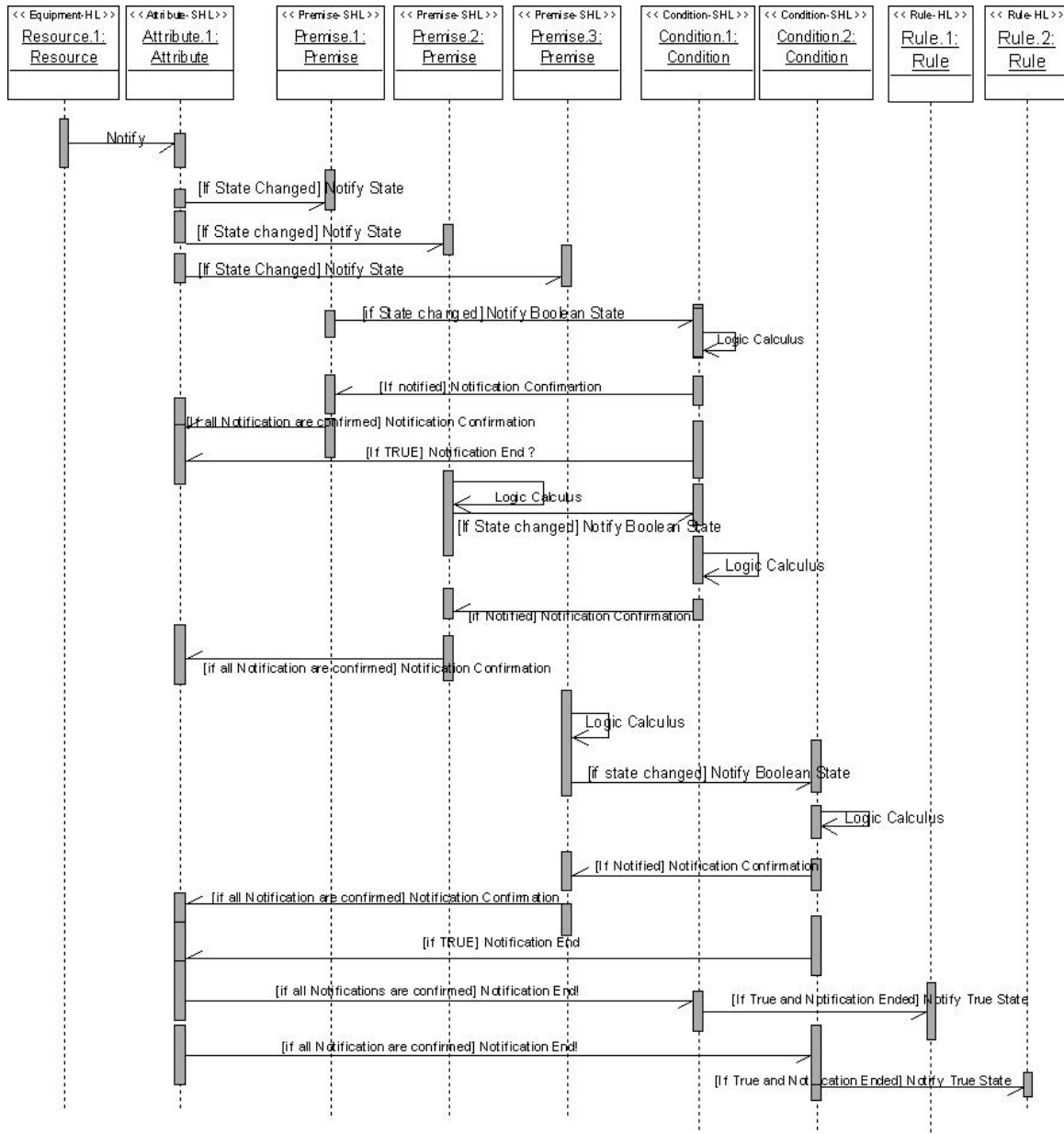


Figure 51: Instance of control entities considering the deterministic-evolution.

#### 5.4.4. Robustness Features

A holonic control is foreseen as distributed to have some robustness features. For instance, the real distribution of the decoupled control prevents all lost of control even if part of the processing environment breaks down. In fact, the robustness features are related to the failure tolerance and avoidance and then they are also related to the correctness features. In this

sense, for example, the proper computational performance and conflict treatment aids to avoiding failures, keeping certain robustness in the control.

In the scope of the control robustness, it is still considered the features of failure tolerance like the entity redundancy for substitute main entities in inoperable or inappropriate states. An example of entity redundancy is the *clones* of Rule presented in section 5.2.3. In the case of *cloned* Rule, it would be created a Rule with the same knowledge of a *matrix* Rule, but having its Action deactivated. The Action of a Rule *clone* would be activated only when it would substitute an inactive *matrix* Rule.

In fact, the mechanism of determinism can allow implementing *cloned* Rules for reaching a robustness feature. By the application of this mechanism, the cloned Rules would have the same inferred information than the matrix Rules in a synchronized way. However, an additional and simple dynamics between Rules would be still necessary to reach the Rule *cloning*. This dynamics is need to reach a certain aware between *cloned* Rule and *matrix* Rules.

In this dynamics, always that a *matrix* Rule had changed its state, it would notify its *clones*. The *clones* would wait for the *matrix* signal. However, in the case of no signal during a specific time after the *clone* approbation, it would ask to the matrix about its status. If an appropriate answer is not effectively achieved, a preferential *clone* would activate its Action and assume the *matrix* role keeping the control functionality.

Still, each *cloned* Rule (and its collaborators) should be placed in other distribution points from the respective *matrix* Rule (and its collaborators) to avoid a same environmental error over both *clones* and *matrixes*. The cloning in distributed control reinforces the need for good balancing of entity distribution in the distributed processing environment.

It is expected that the proposed decoupled control facilitates the balancing task, once constituent entities are functionally independent. The good balancing can help in the robustness firstly optimizing communication and even processing, and also guaranteeing redundancy of entities in different places from their *matrixes*.

This section was neither presented to classify the solution proposed as robust for all possible instances nor to formalize issues about entity redundancy and balancing. This section was presented to clarify that the proposed control solution is open to correctness and holonic features potentially allowing openness to robustness features.

## 5.5. Formalism for Control Expression

### 5.5.1. Introduction

In this section, additional concerns about correct and holonic features are discussed. These concerns are related to the formalism in the synthesis of DEC (Discrete Event Control) and the link to automatic synthesis techniques of DEC. More specifically, a well-known DEC formalism, the Petri net, is investigated in this context.

Petri net (PN) is a generic tool that allows modeling discrete distributed system in a graphical way and simulating its behavior playing tokens in its graphical structure. Additionally, in some classes of PN, it is possible to make some mathematical analysis of system properties. The PN principles are presented in (Cardoso et Valette, 1997) and in the related bibliography.

Petri nets are a formal notation for modeling and analyzing distributed and concurrent systems, and have been widely applied in many fields. There have been more than 6000 publications on PNs (He et al., 1999). Furthermore, PNs are considered a family of

formalisms that provides a framework or working paradigm useful for design and operation in production systems issues, as in manufacturing system control (Silva et al., 1998).

Broadly speaking, a major contribution of PN to that field is to provide a family of formalisms sharing basic principles in a consistent way. *Although for each purpose or degree of detail the adequate formalism would be chosen from the family, the transformation from one formalism to another could be sound, if not formal or even automatic* (Silva et al., 1998).

In this section, it is presented the compatibility between the developed control solution and PN. This would allow more an item of correctness to the proposed solution, meaning openness of the solution to a proper formalism for modeling and designing distributed systems, as manufacturing control, which is also used in the field of control automatic synthesis.

### 5.5.2. Rules and Petri Nets

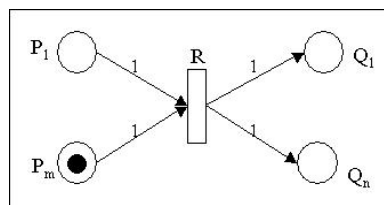
It has been shown that ordinary PN can be considered a RBS in propositional logic and that Predicate-transitions nets (high-level PN) can be considered a RBS in first order logic (Bako et al., 1990). In this sense, *He et al.* present that various graphical techniques were proposed for analyzing various types of structural errors for RBS and that PN are adaptable to this issues<sup>98</sup> (He et al., 1999).

An elementary example of PN is presented in Figure 52. Basically, a PN is composed of transitions that have places connected by means of (input and/or output) oriented arrows. A transition can be fired when each place connected by input-arrows has a foreseen token number (Cardoso et Valette, 1997)(Simão et al. II, 2003).

After the transition fire, the input-places lost a specific number of tokens and the output-places receive another specific number of tokens (Simão et al. II, 2003). The tokens in the places are like a base of fact while transition  $T_1 \dots T_n$  (with the input and output arrows) are like rules  $R_1 \dots R_n$  (with condition and actions) (Simão et al. II, 2003).

A rule  $R$  has the following general form  $P \rightarrow Q$ , where  $P$  and  $Q$  are respectively called antecedent and consequent. In a more detailed way, the rule  $R$  can be defined as  $P_1 \wedge \dots \wedge P_m \rightarrow Q_1 \wedge \dots \wedge Q_n$ . The conclusion or part of the conclusion  $Q_1 \dots Q_n$  from a rule  $R$  can be, for example, the antecedents or part of the antecedents  $P''_1 \dots P''_m$  of another rule  $R''$ . A fact concluded can also be the change of a fact from the antecedent, e.g.  $P_1$  or  $\neg P_1$ .

For each rule  $R$ , an equivalent structure can be finding within a Petri net, due to existent mappings (Etessami et Hura, 1991)(He et al., 1999)(Nazareth, 1993). For example, in Figure 52 this PN represents a rule  $R$  that, if true, the antecedent facts are changed (negated) and new facts are generated<sup>99</sup>. The equivalent rule  $R$  would be  $P_1 \wedge P_m \rightarrow \neg P_1 \wedge \neg P_m \wedge Q_1 \wedge Q_n$ .



**Figure 52: A Rule in Petri net representation.**

<sup>98</sup> Examples of structural errors are inconsistency (conflict rules), incompleteness (missing rules), redundancy (redundant rules), and circularity (circular depending rules) (He et al., 1999)(Nazareth, 1993).

<sup>99</sup> The rule  $R$  is in a false state or, in other words, the Transition  $R$  is not enabled.



### 5.5.3. Petri Net Player

The Bako's studies also consider that PN describes causal relations as RBS (Bako et Valette, 1990). Moreover, the Inference Engine (IE) of a RBS is considered like a PN player in these studies<sup>100</sup>. Bako has proposed a controlled RBS as a PN where a rule is attached to each transition of the PN (i.e. high-level PN). In this approach, the PN player is a variant of the Rete algorithm for IE (Bako et al., 1990)(Cardoso et Valette, 1997).

By considering the previous studies, a control instance that is expressed by means of rules and has an inference process like an IE, could be alternatively expressed by means of PN. Therefore, this compatibility between PN and RBS makes control architectures, based on RBS, open to PN formalism. However, there exist concerns about the *distance* between the PN control model and the control implementation.

A first concern is that a high-level control model of MS, designed with a PN, considers each MS resource in a high-level without care about the resource details, such as state monitoring and command. This first problem can be solved by means of resource homogenization and integration at high-level, namely by means of Resource-HLs. A Resource-HL encapsulates the resource details and enables these details in a high computational level.

When carrying out a control as the playing of high-level PN, the Attribute states of Resource-HLs could be used as marking of *Places*, e.g. the states *free* or *busy* of an Attribute Status could be the marking of a *place* called Status. Also, Methods could be activated in respect to the reached marking of *Places*, letting the Resource-HL in the correspondent states, carrying out the control command as a type of interpretation in the PN.

Another concern is that the IE implementations have been made using search methods to match facts (*marking in the places*) and rules (*transitions*) while the PN playing is conceptually based on sensitization of transitions (*rules*) in relation only to the marking of concerned places (*facts*). This difference can generate certain *distance* between the PN expression and its carrying out.

Opportunely, the proposed control architecture presents the matching of Rules by means of notifications only about the states of the concerned elements, which is similar to the sensitization concept in PN playing. Therefore, it is considered that the proposed control architecture is suitable to the execution of control instances designed by means of PN.

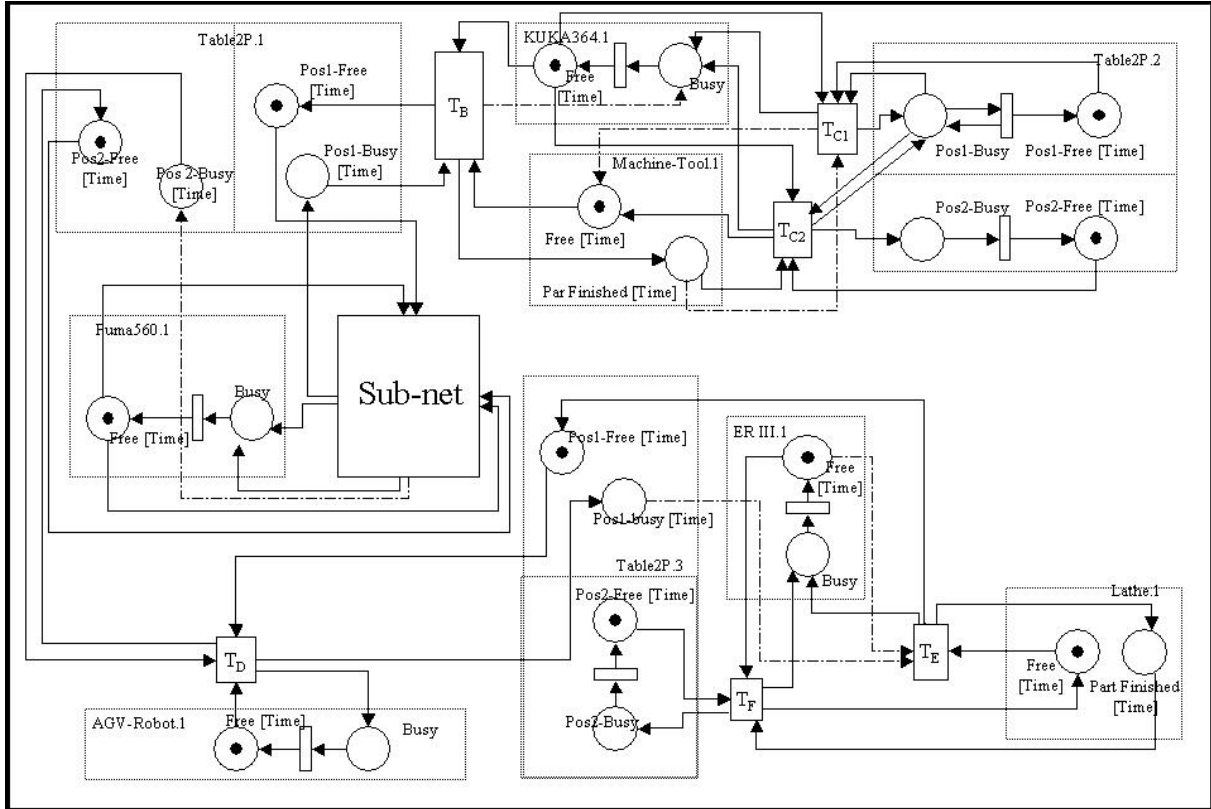
The next section presents a PN model and its "playing" by means of an instance of the control architecture. This next section aims at a more concrete example of the intrinsic compatibility between PN and the proposed control architecture. It is expected that this compatibility opens the rule oriented control architecture to the PN formalism in a clear manner, thereby creating a link to concerned methods of automatic synthesis of control.

### 5.5.4. Petri Net Playing

The PN presented in the Figure 53 and Figure 54 was used to model the control applied over the Resource-HL cooperation simulated in ANALYTICE II and described in the previous chapter. Actually, this PN can be understood as played by the Rules and other holons from the architecture instance. In this sense, each Rule is like a *Transition* and each set of related *Places* is like an Attribute of an Equipment-HL. The related *Places* are delimited by means of rectangles in the PN, being the marking of related *Places* the state of an Attribute.

<sup>100</sup> Basically, a PN player allows carrying out the dynamics in the PN firing the enabled transitions.

The inference process carried out by means of notifications between the control elements allows *emulating* the actual PN playing. In the changing of the Attribute state (*marking changing*), the related Rules (*Transitions*) are notified by means of related Premises (*Arcs*). Therefore, each *Transition* (Rule) can know if an interested *Place* is marked or not, representing if an Attribute state is reached or not.



**Figure 53: A Control designed using a Petri net.**

The described process allows emulating the transition sensitization because each *Transition* can know if the marking need to its firing is reached only receiving notifications about the state of related *Places*. In terms of the Rules, this means to know about its true or false state with respect to the Attributes referenced in its Premises. After the Rule (*Transition*) fire, Instigations promotes (by means of Methods) changes in the state of some Attributes generating other sensitizations.

The PN played by the control instance also presents a P-temporization of some Places<sup>101</sup>. The P-temporization of a Place is related to the time that a Resource-HL is executing a Method. In order to emulate the P-temporization, the Attribute changed to this kind of Method could not make notification that can approve a Rule but should make notification that can disapprove a Rule. This notification to disapprove Rules represents the Transitions perceiving the token lost in the mutually exclusive Place in relation to the Place that received a token.

In order to emulate this P-temporization, each instigated Method of Resource-HL puts a respective Attribute in an intermediate state and internally generates the necessary command to the emulated piece of equipment. The intermediate state of an Attribute is one that does not allow any Rule (*Transition*) be approved because that reached state does not approve any Premise, but it allows Rules be disapproved because the state changing disapproves Premises.

<sup>101</sup> In the start of a P-temporization, the changing of the marking Place does not sensitize concerned Transition, they are sensitized only after the defined time of temporization.

Once the Method finishes its execution (i.e. the operation of the emulated piece of equipment is finished), it puts the concerned Attribute in an *effective state*. The Attribute in an effective state carries out the appropriate notifications potentially reaching approbation of Rules (*Transitions*). This mechanism allows keeping the marking place during the time of the Method execution, for emulating the P-temporization.

Summarily, in the sense of PN playing, after each *Transition* (Rule) execution the related *Places* properly *received or lost tokens*. Internally, the states of Attributes were changed and commands for emulated equipment were done, like an interpreted PN. An interesting point is that, at PN level, the interpretation is hidden inside of Resource-HL. The resulting marking of Places were confirmed in the monitoring of simulated-equipment in ANALYTICE II.

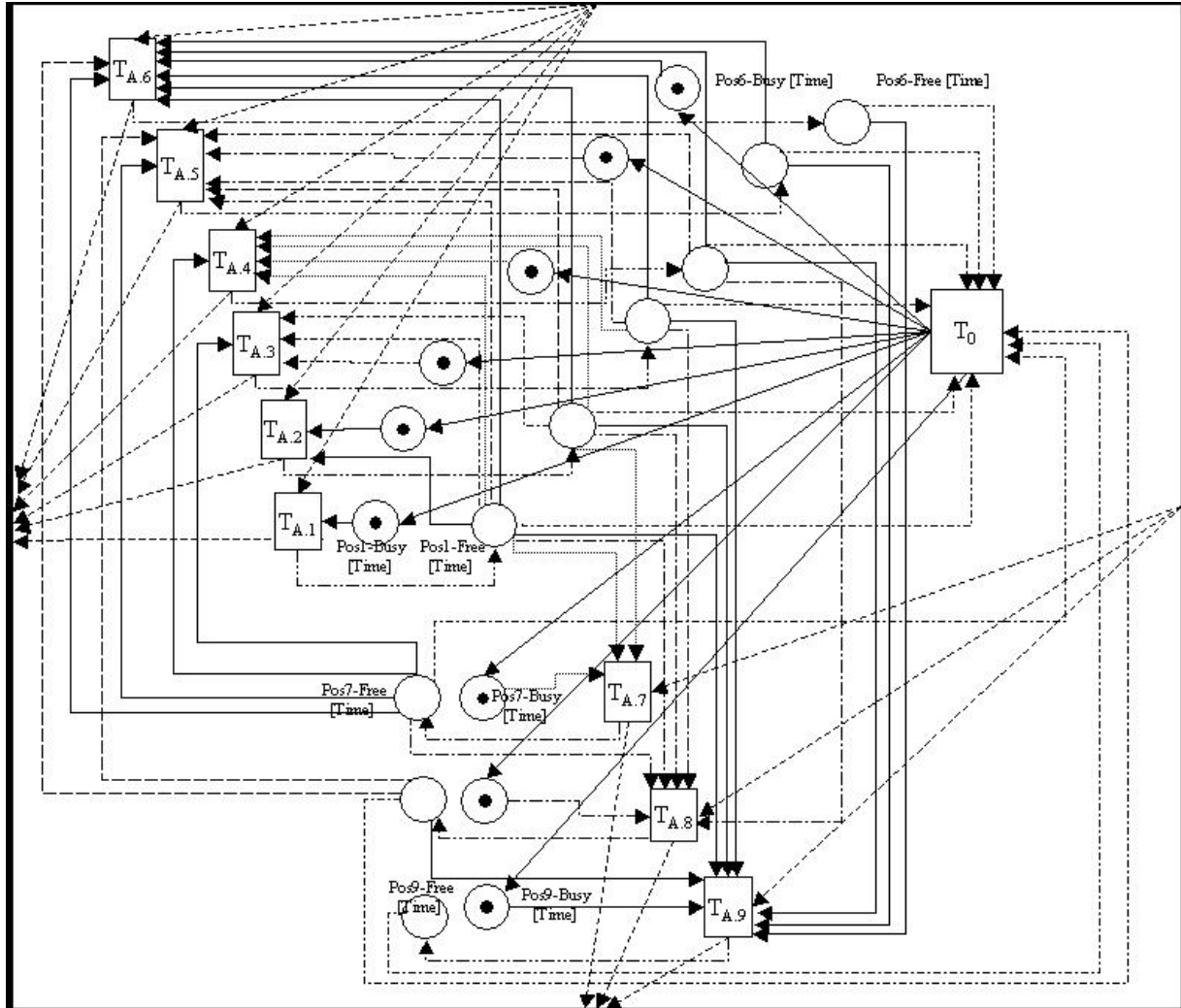


Figure 54: Complementary subnet<sup>102</sup>.

### 5.5.5. Petri Nets and Solution Generality

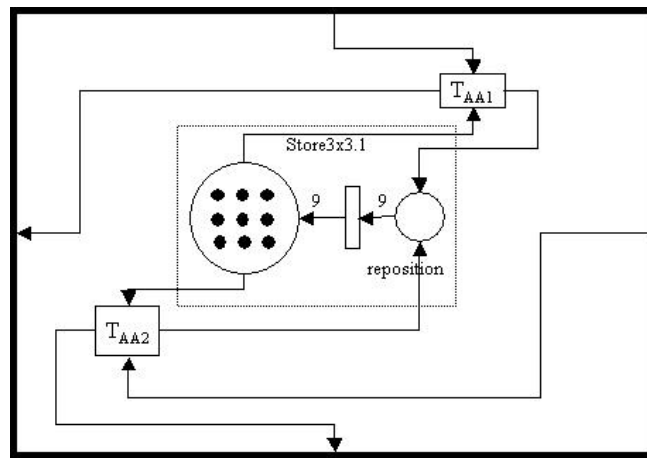
The compatibility between PN playing and the proposed inference-process thinking allows classifying the proposed control architecture as a proper solution to PN player, as well as the PN as a proper modeling tool to its control instances. Complementary, when a control

<sup>102</sup> This sub-net is the equivalent net to the interlocked Rules related to the transport of parts from Store.1 to Table2P.1. The sequence of transition fire, when the entries are enabled, is \$T\_{A.1}\$, \$T\_{A.2}\$, \$T\_{A.7}\$, \$T\_{A.3}\$, \$T\_{A.4}\$, \$T\_{A.8}\$, \$T\_{A.5}\$, \$T\_{A.7}\$, and \$T\_{A.9}\$.

architecture can be understood as a PN player, or effectively is a PN player, this compatibility can be an artifice to reflect about its generality.

Once an instance of the control architecture can play a PN, with the fundamental structures (parallelism, conflict, etc), this indicates the generality of the solution to carry out coordination in Discrete Event System (DES) as much as PN is used to model coordination within DES<sup>103</sup>. In this sense, the experiment presented in the previous section present the fundamental structures of a PN, being the conflict an exception because the PN was applied to a control case<sup>104</sup>.

Additionally, an alternative implementation was made to the subnet in the PN presented in Figure 53. The implementation was made using only two Rules, as presented in Figure 55. However, it was established an order to tokens (parts) exit of the place Store.1. Still, other alternative to the token taking off could be by means of a random policy, without consider the control determinism in this scope (Simão et al. II, 2003).



**Figure 55: An alternative to the subnet in Figure 54.**

A PN player is under development at LSIP using an adaptation of the C++ code used to play or carry out the control example presented in the last chapter. For this PN player, it is foreseen the implementation of the entities from this architecture in separated threads, looking for certain parallelism in the implementation, aiming to consider the time within *Places* and *Transitions*.

This tool can be seen as an intersection of researches developed at LSIP surrounding the execution of Discrete Event Control (DEC) and synthesis methods by means of Petri-nets. Still, a first application related to ANALYTICE II would be to appropriately connect the player to the simulator. This would allow modeling the control with PN and carrying it out as an interpreted PN at graphical level.

### 5.5.6. The Agility and Petri Nets

When a control instance is modeled using a PN model, as an engineering step, it is possible to play this PN for observing its behavior with respect to the agility. For example, the PN presented in Figure 53 could be played or analyzed to observe if there exist flexibility or production alternative of a part type, reachable by the designed control, when some preferential resource is busy.

<sup>103</sup> Actually, any PN similarly presents the same type of structures or entities, entity relation, and entity dynamics.

<sup>104</sup> The control architecture, in relation to its generality, has been also applied as an inference engine to fuzzy system in the scope of some initial and parallel research efforts (Simão et al. I, 2003).

In this sense, when a system receives an additional resource (as presented in section 4.11), it can be considered in the designed control. For example, in Figure 56, the PN is changed to consider a new Resource-HL (the Lathe.2), where the flexibility to the new scenario can be observed playing the enlarged PN. In terms of control, this means more a Resource-HL (*Places*) and Rules (*Transitions*) to better exploit the new enabled flexibility.

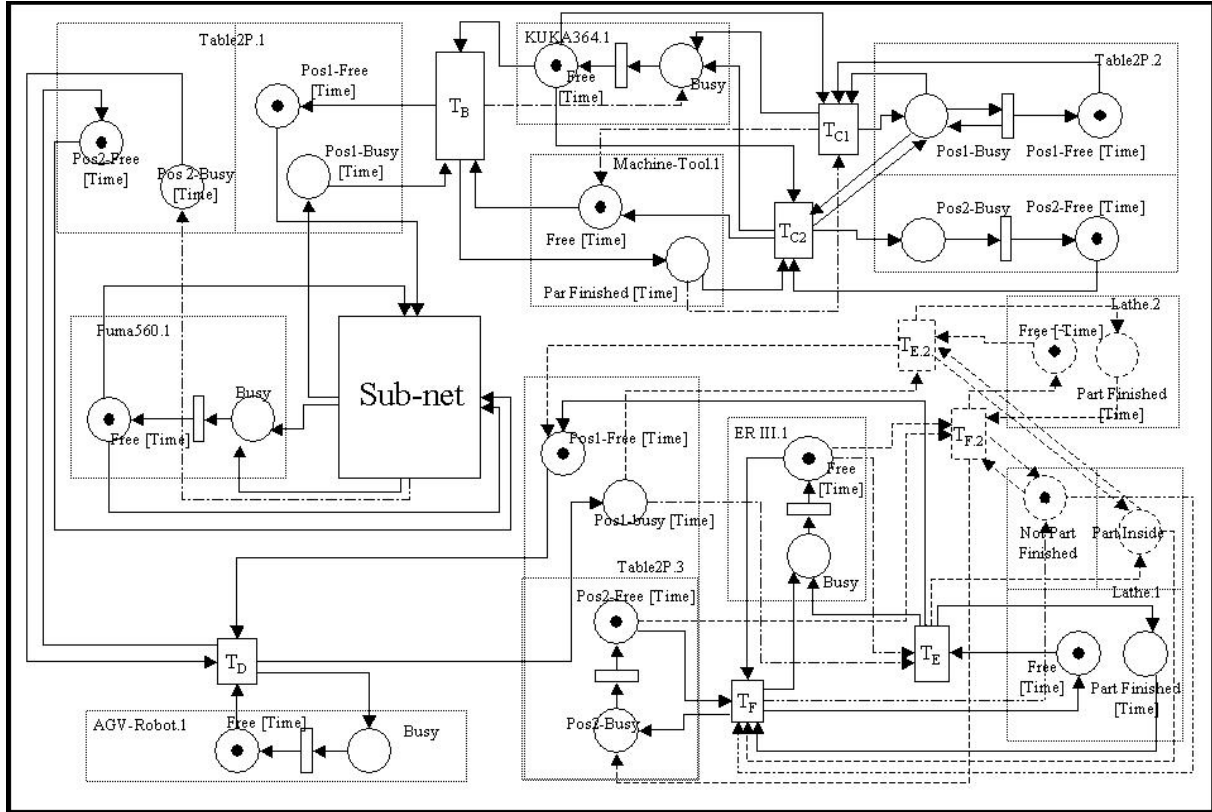


Figure 56: Control Improved.

Moreover, if a control system was really implemented taking into account a proper PN model, it would have the flexibility observed in the PN model, before the implementation, by playing the PN<sup>105</sup>. In an opportune manner, the proposed architecture presents a certain *tendentious nature* for instantiating systems that running as a PN being played, because its essence is like a PN player that works very similarly to the way conceptually conceived to PNs be played.

### 5.5.7. Player Architecture

In this section, in order to better fit PNs as a suitable way to model instances of the proposed control architecture, the architecture is also interpreted as a player architecture for standard PN in a nearer way of PN playing. Furthermore, it is expected that this interpretation of the control architecture be useful for better understanding the execution of its instances likewise the PN playing and to *approximate* the PN playing and the concept of control execution proposed in this work.

For the purpose of applying the proposed architecture as a player to standard PN, the architecture should be interpreted as follow:

<sup>105</sup> Also, the set of possible analysis enabled could be applied to each Petri net that model a control instance.

- Each transition is handled as a Rule (now called *Transition*) where: (a) the set of input-arrows are the Condition (or *Input-Arrows*); and (b) the set of output-arrows are part of the Action (or *Output-Arrows*).
- Each place is handled as a FBA<sup>106</sup> (now called *Place*) that has: (a) only one Attribute called *Token-Number*; and (b) only two Methods called *Put-Token* and *Get-Token*.
- Each transition input-arrow is handled as a Premise (*Input-Arrows-Pre*) where: (a) the number of tokens needed is the *Value*; (b) the *Token-Number* of a *Place* is the *Reference*; and (c) the *Operator* is always a comparison operator.
- Each transition output-arrow is handled as a type of Instigation (*Output-Arrow-It*) that: instigates a *Put-Token* using as parameter the token number of the respective output-arrow.
- Each transition input-arrow is also handled as a type of Instigation (*Input-Arrow-It*) that: instigates a *Get-Token* of a *Place* using as parameter the token number of the input-arrow.
- Each second part of the Action (*Output-Arrows*) is composed of a set of *Input-Arrow-Its*.
- Each FBA could be also seen abstractly as a set of related *Places*.

This interpretation of the control architecture describes, in fact, a restrictive mapping to a PN player architecture. It is expected that standard PNs can be played by instances of this architecture. For example, the safety PN presented in Figures 53 and 54 is playable by an instance of the PN player architecture, if the time in the places is not considered. Additionally, the mapping could be improved to allow playing other classes of PN, as temporized PN, if these classes are not already allowed in the proposed generic architecture of control.

In the case that the architecture takes into account temporization in transitions (i.e. T-temporization), each approved *Transitions* (i.e. Rules) should fire (i.e. *Output-Arrow* execution) only after a predetermined time. Still, in the case that the architecture takes into account temporization in places (i.e. P-temporization), the *Token-Number* (an Attribute) of each sensitized *Place* (i.e. FBA) should notify the appropriate *Input-Arrow-Pre*s (Premises) only after a determined time that its state (i.e. token number) has been changed.

The PN player architecture considering P-temporization is appropriate to simulate control once the *Places* within instances could represent times that Resource-HLs need to execute its operations. Furthermore, the instance could be even used to control of HMS if the *Places* are actually related to Resource-HL and if each *Put-Token* execution internally activates a set of suitable Methods of the Resource-HL.

The *Place* would be really enabled to *sensitize* only after the time that the respective Resource-HL was working. At the end of the Methods execution, the *Put-Token* method would receive an *end signal* to enable the *Place* sensitization. By this way, the control would be carried out in a hidden manner in the PN execution, like a type of interpreted PN. In fact, the derivation of the control architecture to a PN player architecture *returns* to control application but nearer of the PN nature.

---

<sup>106</sup> Fact Base Agent (FBA), equivalent to a Resource-HL, defined in section 5.2.4.

## 5.6. Summary

The control solution proposed in this work has some suitable properties for composing holonic control. These properties comprise integration issues including certain human integration. In fact, the composition, changing, and understanding of Rule knowledge are cognitively natural to the human thinking. Human experts can adapt the control system, manipulating the Rules with certain easiness, based on production needs and on their expertise about the Manufacturing System (MS)<sup>107</sup>.

The human integration is related to the systemic integration searched in HMS, where the system entities synergistically work. Besides the human integration, a larger systemic integration is potentially reachable by the proposed solution. This solution, even if self-contained, is also an open and generic architecture for control issues. This openness has some *branches*, being the integration with other decisional system architectures an example.

Actually, the proposed architecture for control allows composing auto-contained control instances, as well as integrated instances that interact with other decisional entities of the MS. The interactions could be collaborations to improve the efficiency of the controlled system. A way to interact with a control instance can be to act on the Rules, for example, changing their priorities or validities, for aiming to reach a more appropriate MS behavior in a specific context.

In this open control solution, the potential knowledge in the Rules and the potential systemic integration can allow reaching agility objectives, as the exploitation of MS flexibility. The solution presents the basic elements to compose instances of agile control over Resource-HLs like those tested in ANALYTICE II. However, the agility depends on the skills of the designers (human or/and artificial tools) to compose control instances with appropriate knowledge and interactions with proper entities.

This solution does not only present the generic elements to compose agile control for Holonic Manufacturing System (HMS), but also present these elements respecting a trade-off between generality and applicability. For instance, Rules are directly instantiated only giving the appropriate knowledge to Premises and to the Instigations. Also, Premises and Instigations are created only if preexistent ones are not appropriate otherwise pre-created Premises and Instigations are just connected.

The structures related to a Rule (Condition and Premises-Attributes as well as Action and Instigations-Methods) are similarly generic and applicable to a diversity of cases in different scopes. The Rules can be used in the holarchy for any causal relation treatment concerned to elements externally patterned via Attributes and Methods. A clear application example, differing from control, could be in fault supervision where faults would be inferred based on Attribute states and consequent actions would be reached by Method instigations.

The concept of Rule entity and their collaborators implicitly respects the compromise between the “all” (*holos*) and the “specific” (*on*) in holonic system. In the same sense, the inference mechanism is generic (always using notifications) and directly applied and emerged from Rule instantiations in the specific scope. In short, this solution presents a *meta* feature involving generality and applicability, being considered a meta-model.

The control meta-model establishes instances in a connected graph of agents, i.e. the Rules and their collaborators, which implements causal relations reaching an inference process via cooperation based on notifications. Summarily, a control agent notifies only appropriate other

---

<sup>107</sup> The knowledge of Rules would be manipulated in a proper environment comprising a user-friendly interface.

agents about new pertinent information using the receiver addresses. Thus, an agent has certain independence about the receiver localization in terms of local or remote address.

The notification between agents allows even imitating the distributed behavior within non-distributed implementation. In both cases, distributed and non-distributed implementation, only concerned agents receive notifications from another agent when new information is inferred. Therefore, the meta-model also presents *holonism* with respect to the types of implementation, varying from distributed to non-distributed systems and having low computational complexity and optimized communications.

Moreover, the notification mechanism allows reaching other correctness features, as additional mechanisms for conflict and determinism treatment and also the expression of the control by Discrete Event Control (DEC) formalism. In this sense, additional mechanisms were proposed for improving the control solution, but still allowing from non-distributed to distributed instances. In fact, it is expected that these mechanisms facilitate the treatment of concerns of distributed systems, like the distribution balancing and entity redundancy.

A mechanism for conflict issues, based on notification, was proposed for symbiotically working with the notification-driven inference process. This mechanism is established upon the previous information about shared Resource-HLs. Also, a mechanism for deterministic-evolution was defined. It was defined by means of an additional notification chain improving the inference process.

These two mechanisms can be emerged with the basic inference process in the creation of the Rules. The two mechanisms can also allow deterministic instances keeping the reactivity due to the implicit optimization in the inference given by the proposed solutions. Actually, the notification concept allows proposing a non-complex solution to control concerns both in distributed and non-distribute environment.

Additionally, the notification-driven mechanism is similar to the playing ideally thought to Petri net (PN). The solution proposed is a way to *carry out* causal relations in a decoupled or distributed way while the PN is a tool to model distributed causal-relations. This compatibility is opportune because causal rules can be considered a so soft formalism to express the control, PN is largely used in DEC, and the solution proposed can still be a way to *reduce the distance between* PN model and its implementation.

It is expected that this compatibility still give other benefits to reach proper holonic controls. A control correctly designed by a PN and effectively played by an instance of the generic architecture could have agility properties previously observed playing the PN. Furthermore, the PN is a formalism applied in the automatic synthesis of controllers creating a link between the proposed solution and the automatic synthesis school.

A reasonable hypothesis is that this solution proposed could still be used, in some way, to deal with certain system-classes of DEC in which the PN and causal relation (e.g. rules) are used as modeling tool. In this sense, even if self-contained solution, the meta-model proposed is an open solution to interpretations, particular improvements, systemic improvements, systemic integration, and so forth.

Anyway, within the scope of this thesis and based on the exposed arguments, the presented meta-model is considered open to correctness and holonic features whose essence has been tested within ANALYTICE II. Nevertheless, a pertinent explicit improvement to the meta-model, proposed for process-driven HDCS, would be a *conformation* to product-driven HDCS. This *conformation* is considered in the next chapter.



**Chapter 6**

---

Product Driven Control



# Chapter 6 : Product Driven Control

## 6.1. Introduction

The present automated identification (Auto-ID) technology enables accurate timely information about a specific item to be stored, retrieved, and communicated. This information can be even used to assist in automated decision-making and controlling relevant to that item (Kärkkäinen et al., 2003)(McFarlane et al., 2002). In this sense, *information and, beyond, a form of technical 'Intelligence' embedded into the manufacturing systems components and even inside the product themselves, are playing a prominent role as pivotal technologies, which makes possible to address agile Business to Manufacturing (B2M)* (Morel et al., 2003).

Effectively, Auto-ID systems, which involve at least the automated retrieval of the identity of objects, are becoming a reality for monitoring items moving through the manufacturing supply chain (García et al., 2003). However, the manufacturing deployment of Auto-ID technology is unlikely to be its first industrial application as there are immediate benefits in warehouse and retail environments to be achieved in a relatively straightforward manner (McFarlane et al. 2003).

Anyway, many of the often-cited long-term challenges for manufacturing-customer responsiveness (e.g. ready mass customizability and small volume/high variety order management) can be addressed by the combined application of Auto-ID technology and distributed intelligent control. In this context, the Auto-ID and some control smartness *cohesively* related to a specific product has been called intelligent or smart-product, whose technical viability has been studied and foreseen via experiments (McFarlane et al. 2003).

The main idea in product-driven control is that each smart-product suitably drives its own production considering and using the enabled resources. The smart-product, beyond some coherence between informational and physical flows, allows certain decoupling both in control information about each specific item in the production and in the production demand and execution. These features are useful to holonic objectives as previously presented in this thesis. Concordantly, the smart-product has been understood as a holon (Gouyon et al., 2004).

In the case of product-driven holonic control, the control feedback is incremented with the smart-product holon information, as depicted in Figure 57 about the feedback evolution from a process to a product-driven control (McFarlane et al. 2003). This feedback control, involving Smart-Product-HLs, takes place inside of each Resource-HL in different recursive levels, like an Equipment-HL, a Process-Cell-HL, or an Area-HL. A specific example would be in the context of the holonic Machining Cell presented in the chapter 4.

In the presented holonic Machining Cell, the monitoring and actuation (transport, storage, or machining) are performed by Equipment-HLs, while the control (i.e. the decision and coordination about Resource-HL cooperation) is carried out by the Holonic Supervisory Control (HSC). In a product-driven case, the HSC would not work only in reaction to the states that are notified from Equipment-HLs, but also in reaction of the “desires” (i.e. production need) that are notified by the Smart-Product-HLs inside.

In this chapter, it is firstly made a brief discussion about the Smart-Product-HL implementation focusing in the scope of ANALYTICE II. However, the main objective of this chapter is to improve the generic solution, previously proposed, for supporting the product-driven holonic control based on Smart-Product-HLs. Summarily, the main idea is to

use Rules to organize and regulate the collaboration of Smart-Product-HLs and Resource-HLs, brings *mutual* advantages over rule-driven control and product-driven control.

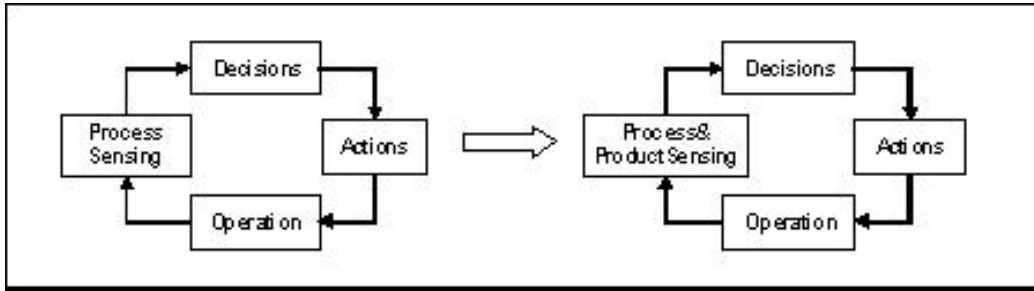


Figure 57: Feedback control evolution.

## 6.2. Smart-Product-Holon

The Smart-Product-HL implementation can be more concentrated in its soft part or in its hard part. The soft part can be a software agent and the hard part can be a communicable electronic component (smart-tag) connected to the physical product. A present technology to smart-tags is based on the RFID (radio frequency identification) technology.

In the RFID case, it is possible at least to identify each object and implement all its smartness in a software agent letting the Smart-Product-HL implicitly integrated in the informational world. On the other hand, some smartness in the hard part of Smart-Product-HL could give benefits, such as an implicit coherence of informational and physical flows.

In this work, it is considered the software implementation of the Smart-Product-HL smartness. Nevertheless, the main point is the Smart-Product-HL concept as an *atomic entity*, in the HMS, thought in terms of holons negotiating at high-level in the computational world. On the control side, the implementation place of the internal smartness of Smart-Product-HL is hidden, being the important feature its interaction skills at computational world.

Smart-Product-HL can be implemented in ANALYTICE II. However, before understanding the potential simulation of Smart-Product-HL, it is needed to understand the ANALYTICE II product-emulation. At the moment, a simple computational object deals with each emulated-product at plant level. The object for emulated-product has a geometric cube associated for its physical representation in the graphical animation of the simulation. In ANALYTICE II, there is not effectively implemented the product auto-id yet.

However, in each “physical” interaction of a product with a piece of equipment, ANALYTICE II knows that a specific object representing a product is in that interaction position. In fact, the module plant ambient of ANALYTICE II that is responsible for this monitoring (see section 3.3 and 3.4). In each physical interaction, the plant ambient could get an Id of this product (e.g. the object address) and generate a message to be sent to the virtual-net. In this case, in the control side, the respective soft part could be updated allowing a connection about the product-flows.

This type of implementation could be important if errors about the product-position in ANALYTICE II could be generated. Therefore, mechanism in the soft-holon could perceive the inconsistency between the position expected and the real position. However, in ANALYTICE II, errors in the product flows are not generated yet. The unique error possible about parts is an incorrect physical interaction. In this case the simulation is stopped and advices are sent to the user.

As the flows inconsistencies are not generated, in the product-driven control experiments it is considered only the soft part of the Smart-Product-HL. In fact, the physical part (the product) is also simulated, but it is not directly connected to the soft part. Anyway, it is not in the scope of this research concerns about the specific implementation of the flows coherence related to Smart-Product-HL in ANALYTICE II.

The main concern about Smart-Product-HL in this work is to propose a holonic control solution allowing product-driven implementation, firstly in the scope of ANALYTICE II, where the Smart-Product-HL is considered as an *atomic entity* reachable at computational world<sup>108</sup>.

### 6.3. Rules and Smart-Product-Holons

A Smart-Product-HL has production need in the context of its manufacturing. Each production need is metaphorically named as a *desire*. The desires are basically the operations of resources defined in a process plan. In a process plan, it is defined the production steps (operations in resource) for a type of product.

The Smart-Product-HL should find a set of satisfier-Resource-HLs to carry out each desire. Opportunely, as previously presented, a coordination Rule is created exactly to know which are the Resource-HLs able to carry out an operation, via cooperation, related to a set of product type. Furthermore, a Rule knows how and when this set of Resource-HLs can effectuate this cooperation. Therefore, it is envisaged that each Smart-Product-HL could use a set of related Rules to reach its production desires, i.e. operations need.

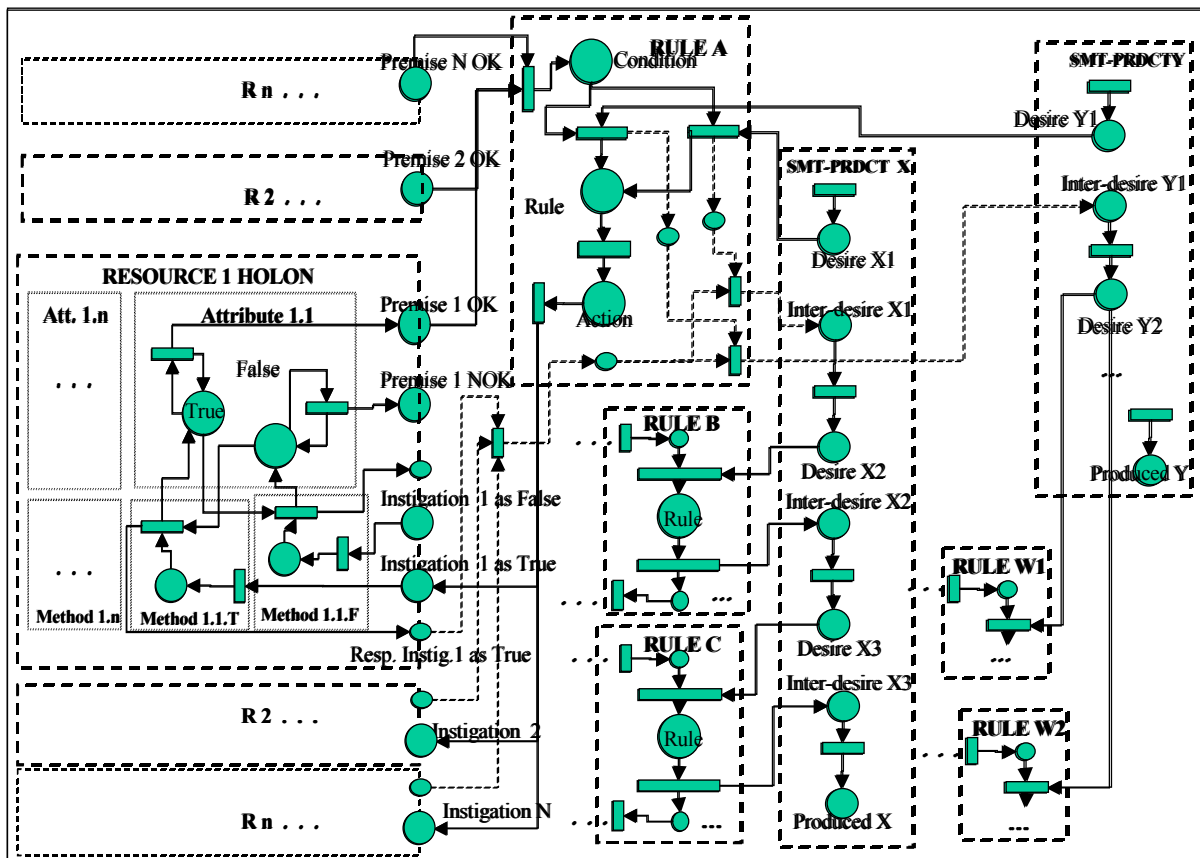


Figure 58: Dynamics concerned to Smart-Product-HLs, Rules and Resource-HLs.

<sup>108</sup> The Resource-HLs are also seen as atomic entity in the control side. However, the implementation of a *consistence process* between soft and hard-parts of Resource-HL were imperative for carrying out functional experiments.

The appropriate Rules for a Smart-Product-HL could be known by means of a Product-Type-HL. A Product-Type-HL has been thought for some automatism in the Rule composition, based on the production steps from a process plan<sup>109</sup>. Therefore, a Product-Type-HL could also know what Rules are related to each operation and then each Smart-Product-HL started would know the appropriate Rules by means of a related Product-Type-HL.

Anyway, once a Smart-Product-HL has been informed about the appropriate Rules (by some manner), it could appropriately select and allocate a Rule (among the possible ones) for reaching each desire. In this context, an approved Rule would depend also from the allocation of a Smart-Product-HL to execute its Action.

In the product-driven case, the Rule is equally responsible for requesting services from Resource-HLs during its execution. However, now these requests to Resource-HLs, for instigating their cooperation, are direct concerned to an identified Smart-Product-HL. A generic model presented in Figure 58 sketches the interaction foreseen for these entities.

A Smart-Product-HL, after its initialization, would select and allocate a Rule among foreseen ones to make its first operation. After the Resource-HL cooperation has been made, it would choose another Rule for its second operation and so on. The allocation would be made by a notification between the Smart-Product-HL and the Rule.

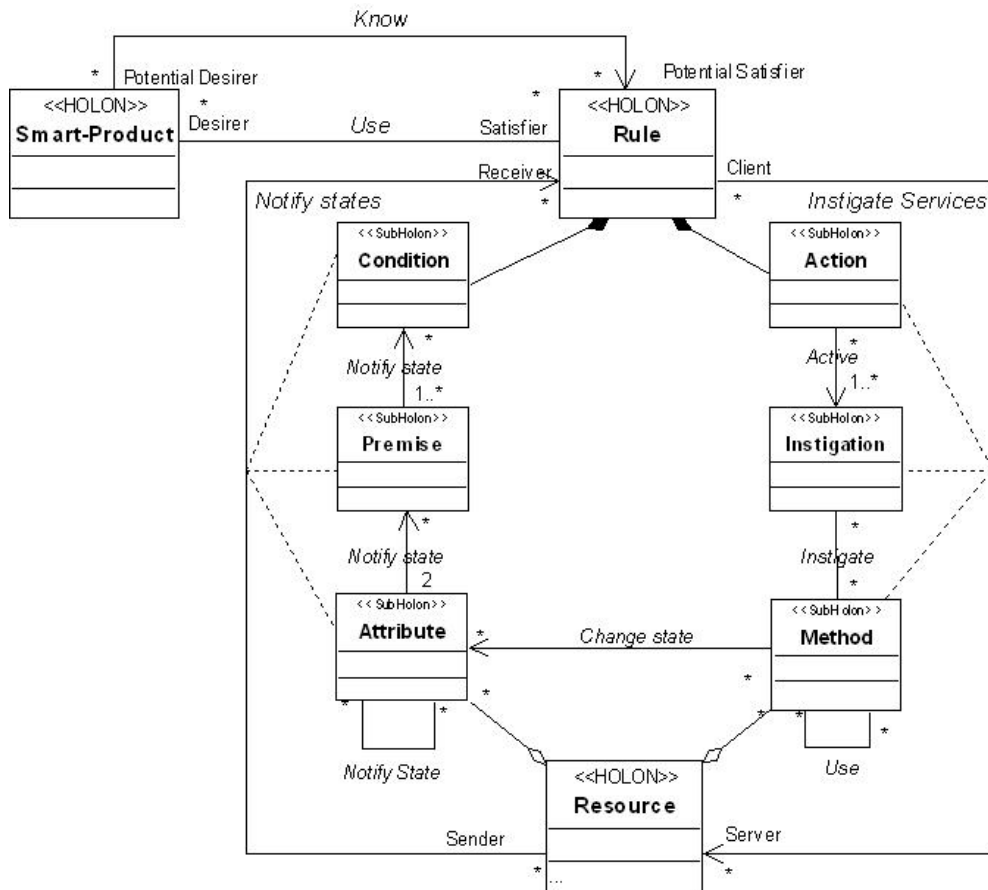


Figure 59: Rule and Smart-Product Holonic-Classes<sup>110</sup>.

<sup>109</sup> Product-Type-HL has been considered in section 4.10.

<sup>110</sup> In the Figure 59, the three intermediate *Holonic-Classes*: *Attribute*, *Premise* and *Condition* are interpreted as a relation *Notify State* between *Resource* and *Rule*. Likewise the three intermediate *Holonic-Classes*: *Method*, *Instigation* and *Action* are interpreted as a relation *Instigate Services* between *Rule* and *Resource*. In the Smart-Product-HL viewpoint, the important is to know what Rules control what Resource-HL cooperation. The control way is hidden at Smart-Product-HL level.

In this proposed approach, Smart-Product-HLs do not negotiate directly with Resource-HLs, these negotiations are organized and regulated by means of Rules. The Rules allows, for example, identifying conflicts in the use of resources and avoiding temporal and structural redundancies in the negotiations.

In this product-driven control, a Rule function is to know the exact moment that a Resource-HL set can collaborate to execute an operation sequence interesting for a Smart-Product-HL set. Therefore, in this context, a Rule can be seen as decoupling mechanism about negotiations between a set of co-operable Resource-HLs and a set of Smart-Product-HLs that are potentially desirers of their cooperation, for reaching a determined operation.

A specific relation *Use* between *Rule* and *Smart-Product Holonic-Classes* is then proposed in Figure 59, where it is observable the independency of Smart-Product-HLs and collaborators of Rules. In this model, it is defined that Smart-Product-HLs would reach services of Resource-HLs using Rules.

## 6.4. Control Instance

The product-driven holonic control was partially tested in the same case of the manufacturing cell presented in chapter 4, comprising the simulation of the production of two part types. The test was made over the transport of parts from the Store to the Table 1. However, for this experiment, the Rules before defined to that transport were substituted by other two Rules, presented in Figure 60.

Rule A.1 If Puma560.1 Status = Free Table2P.1 Pos1 = Not_Busy Store3x3.1 Status = Not_Empty Then Puma560.1 Transport(Store3x3.1.Pos?, Table2P.1.Pos1)	Rule A.2 If Puma560.1 Status = Free Table2P.1 Pos2 = Not_Busy Store3x3.1 Status = Not_Empty Then Puma560.1 Transport(Store3x3.1.Pos?, Table2P.1.Pos2)
---	---

**Figure 60: Rules to transport part from Store3x3.1 to Table2P.1.**

The two Rules defined are more concise than the previous ones. These two Rules were elaborated aiming to profit some advantages possible by the use of Smart-Product-HL. The Smart-Product-HL that allocates one of these Rules must inform its Store position as a parameter to the Rule. After that, this parameter is transferred by the Rule, via its Actions, to the appropriate Instigation.

The conscience of the Smart-Product-HL about its position must be allowed by someday. A way could be the Store3x3.1 to know, when the Smart-Product-HL arrives, what product is therein, e.g. using captors supporting Auto-ID for a real project. After that, the Store3x3.1 would still notify this Smart-Product-HL about its position by means of a mailer. In ANALYTICE II, for a simulation purpose, when the part is put in the Store the respective Smart-Product-HL is informed about its position.

In the experiment carried out, it was observed that the products could be putted in any place of the Store, because each Smart-Product-HL allocates and sets the correct Rule to arrive in Pos 1 or Pos 2 of Table2, depending of its type. This allows an independence of pre-allocated position in the Store, i.e. it could be putted n ( $n < 10$ ) products V1 and m ( $m < 10-n$ ) products V2 in the Store, allowing a better buffering utilization.

Smart-Product-HLs could even compete for the allocation of a specific Rule, e.g. three Smart-Product-HLs in the Store3x3.1 that compete for using a Rule to be transported to the

Table2P.1. The conflict can be solved by many ways, as by production priority of Smart-Product-HL or/and by advice from another entity like HFM (Holonc Flow Management).

The priorities have been used, in the experiment carried out, for deciding what Smart-Product-HLs could allocate one of the Rules. The Smart-Product-HL priorities were related to the arriving order in the Store. The first Smart-Product-HL that arrived had the better priority, while the last Smart-Product-HL that arrived had the worst priority.

In this case, there is no more the need of interlock between Rules or conflict solver because the Smart-Product-HLs that defines their order to be taken off. This means more adaptability because even if a type of product could not be produced, the other could be produced independently, without Rules resetting.

In this product-driven experiment, the number of Rules was reduced and Rules simplified because part of control information come from the Smart-Product-HLs. In fact, some information that was previously extracted from the process (in the process-driven control), by means of a declarative way in the Rules, now comes from the Smart-Product-HLs.

In the subsequent steps of the production of each product, in the Manufacturing Cell considered, there not exist competition between Smart-Product-HLs. The system setting does not present flexibilities for having competitions. Therefore, the experiment could be completed by Smart-Product-HLs allocating the Rules previously defined.

However, in the case of products type V1.2 supposed in section 4.11, the Smart-Product-HLs can also present some advantages. In this sense, for the Smart-Product-HL be advantageous, some changes in concerned Rules should be carried out to profit its knowledge, as presented in Figure 61.

<p>Rule B'</p> <p>If</p> <p>KUKA386.1      Status= Free</p> <p>Puma560.1        Status= Free</p> <p>Machine-Tool.1   Status= Free</p> <p>Table2P.1        Pos1 = Busy</p> <p>Then</p> <p>KUKA386.1 Transport(Table2P.1 Pos1, Machine-Tool.1)</p> <p>Machine-Tool.1 Prepare_to_Receive(Part X?)</p>	<p>Rule D'</p> <p>If</p> <p>AGV-Robot.1      Status= Free</p> <p>Puma560.1        Status= Free</p> <p>Table2P.3        Pos1 = Not_Busy</p> <p>Table2P.1        Pos2 = Busy</p> <p>Then</p> <p>AGV-Robot.1 Transport(Table2P.1 Pos2, Table2P.3 Pos1)</p>
<p>Rule C1'</p> <p>If</p> <p>KUKA386.1      Status= Free</p> <p>Machine-Tool.1   Status= Part_Finished</p> <p>Table2P.2        Pos1 = Not_Busy</p> <p>Then</p> <p>KUKA386.1 Transport(Machine-Tool.1, Table2P.2 Pos1)</p>	<p>Rule E'</p> <p>If</p> <p>ERIII.1          Status= Free</p> <p>Lathe.1          Status= Free</p> <p>Table2P.3        Pos1 = Busy</p> <p>Then</p> <p>ERIII.1 Transport(Table2P.3 Pos1, Lathe.1)</p> <p>Lathe.1 Prepare_to_Receive(Part Y?)</p>
<p>Rule C2'</p> <p>If</p> <p>KUKA386.1      Status= Free</p> <p>Machine-Tool.1   Status= Part_Finished</p> <p>Table2P.2        Pos1 = Busy</p> <p>Table2P.2        Pos2 = Not_Busy</p> <p>Then</p> <p>KUKA386.1 Transport(Machine-Tool.1, Table2P.2 Pos2)</p>	<p>Rule F'</p> <p>If</p> <p>ERIII.1          Status= Free</p> <p>Lathe.1          Status= Part_Finished</p> <p>Table2P.3        Pos2 = Not_Busy</p> <p>Then</p> <p>ERIII.1 Transport(Lathe.1, Table2P.3 Pos2)</p>

**Figure 61: Rules to the co-ordination of Resource-HLs.**

In this redefined set of Rules, when a Smart-Product-HL of type V1 allocates the Rule B', the parameter related to the *Instigation Prepare\_to\_Receive (Part X?)* would be given on the fly. This means that, it would not be necessary to redefine a new Rule B.1' as in the previous



process-driven control, because the existent Rule B' would be correctly parameterized by Smart-Product-HLs of type V1.2 or V1.

## 6.5. Smart-Product-Holon Issues

In the previous section, some profitable features of the Smart-Product-HLs were highlighted in a symbiosis with the rule-oriented control. This section presents other advantages of this symbiosis as well as other inherent advantages of the Smart-Product-HLs.

Generally, a Smart-Product-HL can also have more than one Rule as alternative. This depends on the alternatives determined in the process plan. For instance, in the case of the supposed Lathe.2 in the simulated cell (presented in section 4.11), Smart-Product-HL of V2 type would have two Rules as alternative to reach the desired lathing.

The Smart-Product-HL can select and allocate one of Rules (maybe the best) to execute its desire. The choice of a Rule by a Smart-Product-HL can be based on its own knowledge, on the knowledge acquired by other Smart-Product-HLs or on the advice from some holon, such as the HFM responsible for optimizing the production flow.

For instance, always that a Smart-Product-HL had used a set of Rules to reach a desire, it could get the time used. This information would be transmitted to the respective Product-Type-HL that would use it for generating some quality parameter (e.g. time average) about that set of Rules. Therefore, other related Smart-Product-HL could use this information at the future, from the respective Product-Type-HL choosing appropriate Rules<sup>111</sup>.

Smart-Product-HLs, beyond the potential benefits to flow management in the use of Rules, could be useful also in fault tolerance scope. A specific example is a non-finished Smart-Product-HL that arrives in a place (e.g. buffering place) where it does not find a Rule to allocate. This could be perceived by the inactivity over that respective product, or even the Smart-Product-HL could put itself in abnormal state.

In that case, as an example of adaptability feature, it could choose Rules that would allow it arriving in an unloading position, avoiding the production blocking<sup>112</sup>. Additionally, it could also notify a related Fault Supervision System that, for example, could ask for human intervention.

Smart-Product-HLs can be also helpful for reaching consistencies in the production, also related to fault detection and tolerance. For instance, after the Smart-Product-HL arrives in the scope of a Processing-HL (e.g. Machine-Tool.1), it could confer if the setting is correct because errors can happen, e.g. product interception or data transfer failure.

In the case of errors in the Processing-HL setting, the Smart-Product-HL could notify the Fault Supervision System. Moreover, the Smart-Product-HL could try resetting the Processing-HL to support its production or it could use appropriate Rules to exit of the system, searching for some adaptability.

Furthermore, the Smart-Product-HL could also present other advantages non-directly concerned to the proposed product-driven architecture, namely advantages to the traceability issues. For traceability, for instance, after the Smart-Product-HL be processed in a Resource-

---

<sup>111</sup> This strategy described is similar to the "Ant Theory" presented in (Valckenaers et al., 2003). Any way, a deeper specification of the methodology to this kind of chooses would bypass the unclear frontier of HDCS functions arriving in the HFM scope, which is a kind of on-line scheduler.

<sup>112</sup> Sets of Rules allowing a Smart-Product-HL reaches an unloading position, from any position in the system, could be defined for fault tolerance purposes.

HL, both holons could register information about the operation carried-out. This information stored could have a lot of uses, e.g. production statistics, quality measurement, and product recalls.

These properties highlighted about Smart-Product-HLs interactions with Resource-HLs have been envisaged or even demonstrated in experiments described in the literature aforementioned. The contribution of the present work to this field is the organization and openness to correctness features to these interactions by means of Rules.

The idea of organizing the cooperation of Resource-HLs by means of Rules also allows avoiding unnecessary repetitions of negotiations each time that an operation has to be performed in the holarchy. In fact, the Rules may already foresee a set of possible cooperation in the manufacturing system, because they are created based on Resource-HLs capabilities and the process plans of Product-Type-HLs.

Actually, in the conception of a HDCS, as well as in the conception of each HMS subsystem, it is important to optimize the number of negotiations carried out by the holons. The requisites demanded by a holarchy, implying in the negotiations and cooperation of many holons, may generate a great number of interactions. For instance, in the case of a holarchy with smart-product holons presented in (McFarlane et al. 2003), it is noted an inflation in interactions related to this kind of holons<sup>113</sup>.

In relation to the solution before proposed, for process-driven holonic control, this *upgrade* to product-driven orientation contributes in the reduction of Rule information and even Rule quantity. Actually, part of information need to control is given by Smart-Product-HL, which avoids putting equivalent information in the process for exploiting it in the Rules. Moreover, the solution is improved with the intrinsic advantages of Smart-Product-HLs.

## 6.6. Holons Interaction

In this section, it is exemplified holon interactions in the product-driven control foreseen by means of some models. This models would be applicable in the Holonic Manufacturing Cell studied considering a *strong* implementation of Smart-Product-HLs. In the models, it is instanced some presented features about Smart-Product-HLs.

A first model is presented in the *Holonic-Sequence-Diagram* of Figure 62. In this *Holonic-Sequence-Diagram* it is modeled the coordination of Resource-HL cooperation by way of a Rule, which depends upon a Smart-Product-HL desire to carry out the coordination.

The designed Smart-Product-HL drives a production step, at a high level, using a Rule to arrive in its goal. It allocates the Rule E', for aiming to arrive in the Lathe.1, due to a lathing desire. The Smart-Product-HL gives, throughout the allocation process, the appropriate parameters to the Rule carries out the coordination.

After the execution of Rule E', the Smart-Product-HL explains its machining desire to the Lathe.1 (e.g. *a perforation with smooth, remove, and inside finalization*) to confirm the parameters previously gave to the Rules. Additionally, the Smart-Product-HL could aid the

---

<sup>113</sup> Furthermore, if considered some practical aspects, this interaction will be carried out over a network (field bus, TCP/IP, etc), where it is equally important reduce the number of interaction in the network. This would collaborate to the service quality, in agreement with the systemic approach.

Equipment-HL with specialized information, e.g. pointing a specific CNC or even the need for a specialized tool<sup>114</sup>.

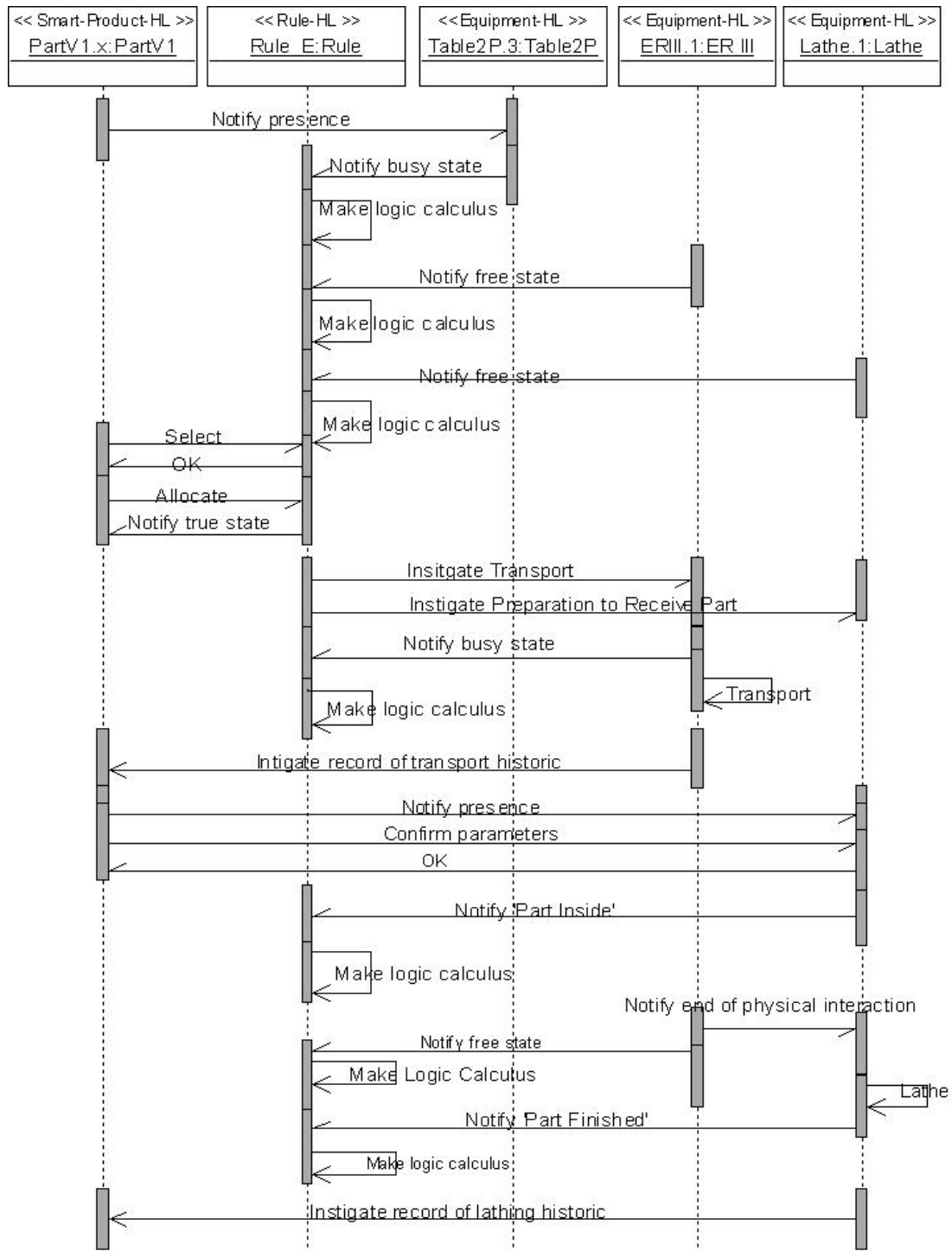


Figure 62: Smart-Product-HL allocating Rule E’.

In fact, this type of interaction described permits reactions using machine flexibility, via re-setting. For example, when a Smart-Product-HL must be processed in a non-determined order, the Equipment-HL can be directly reset with information from the Smart-Product-HL. The

<sup>114</sup> Holons like Lathe.1 could also perceive, after some repetitive interactions, the default setting related to a product type in a specific state of production, such as the default need of Smart-Product-HLs of type V2 that arrive in the Lathe.1. However, the Smart-Product-HL could notify another setting, if it extraordinarily does not want the default setting.

information could be provided at the moment that it allocates a concerned Rule or even at the moment that it arrives within the Equipment-HL.

In this presented example, at the end of the physical operation over a product, the Lathe.1 could instigate the recording of concerned information in the respective Smart-Product-HL, focusing traceability (historic) aspects. In this case, the Machine-Tool.1 could release that Smart-Product-HL only after the information be effectively recorded. Generally, in a Smart-Product-HL interaction with a Resource-HL, they could change relevant information for holonic issues, as agile control and tracing information.

Another model of possible interaction is the Rule B' being used for a Smart-Product-HL, as presented in Figure 63. It is observed, by comparison of diagrams in Figure 62 and Figure 63, that the dynamic of the interaction is very similar in the two models presented.

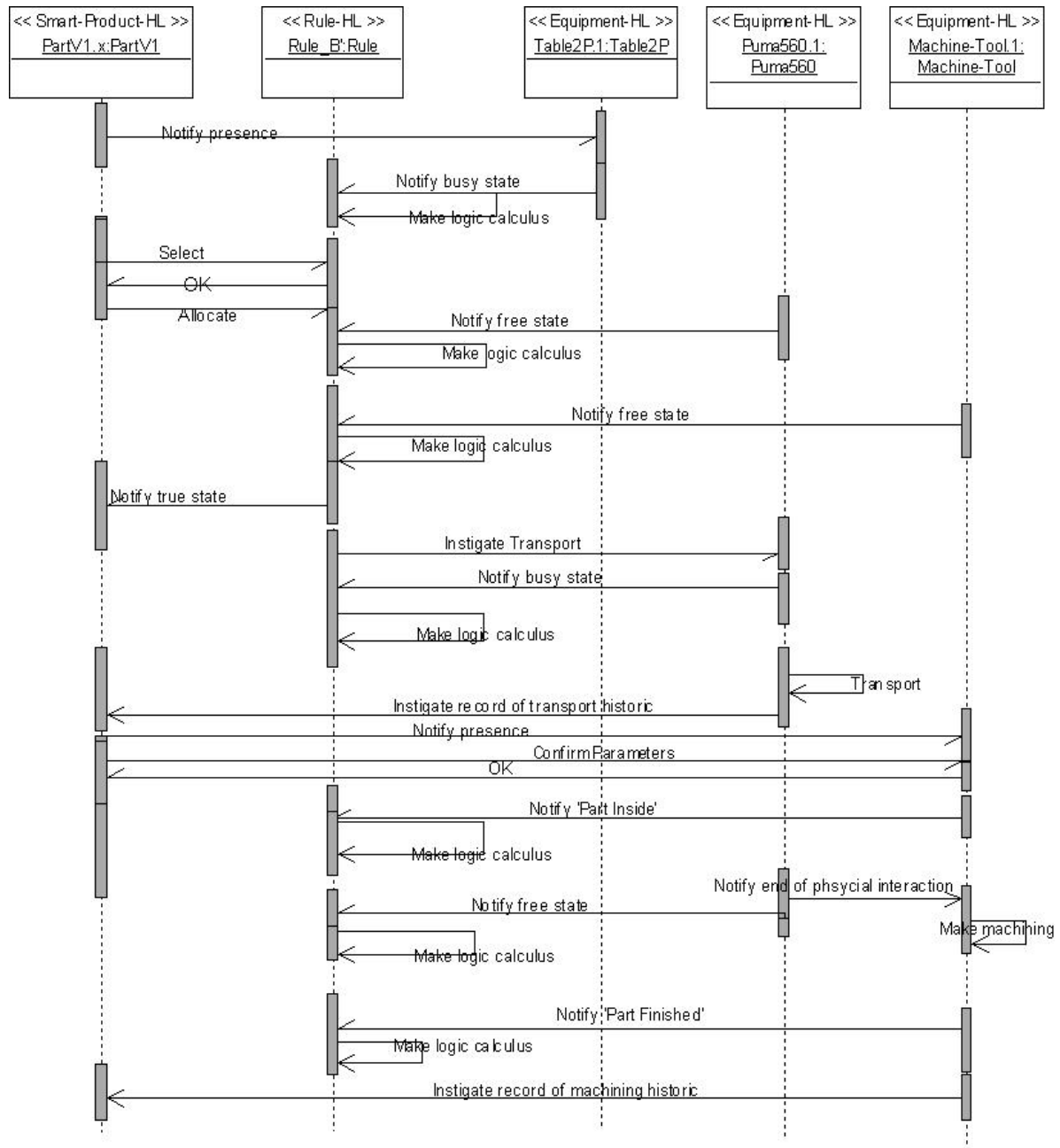


Figure 63: Smart-Product-HL allocating Rule B'.

When a Smart-Product-HL enters the Rule E', this means that it has arrived on the Table2P.1. In the moment that execution of this Rule starts, its Action instigates the KUKA386.1 to carry out the transport to the Machine-Tool.1 and the Machine-Tool.1 to prepare, itself, to receive the parameterized part-type.

Once the Smart-Product-HL arrives within the Machine-Tool.1, it confirms its need to Machine-Tool.1 (e.g. *Perforation of tree holes with groove and engrave*). After the Machine-Tool.1 execution, it instigates the machining-information recording in the Smart-Product-HL.

### 6.7. Holonic Control Architecture

Rules can be referenced as a black box when related to Smart-Product-HLs, such as made in the two models presented in the previous section, because each Rule works with weak dependency about Smart-Product-HLs. This dependency consists only in the allocation of a Smart-Product-HL to activate the Action of the Rule, when it is in true state and ready to execute, and, for times, some parameters for its Instigations.

The *Rule Holonic-Class* is presented in a higher level, i.e. without the collaborator *Holonic-Classes*, in the holonic-class diagram in Figure 64. This diagram was conceived aiming at a better understanding about Rules and other holon relationships in this context of the product-driven HMS proposed.

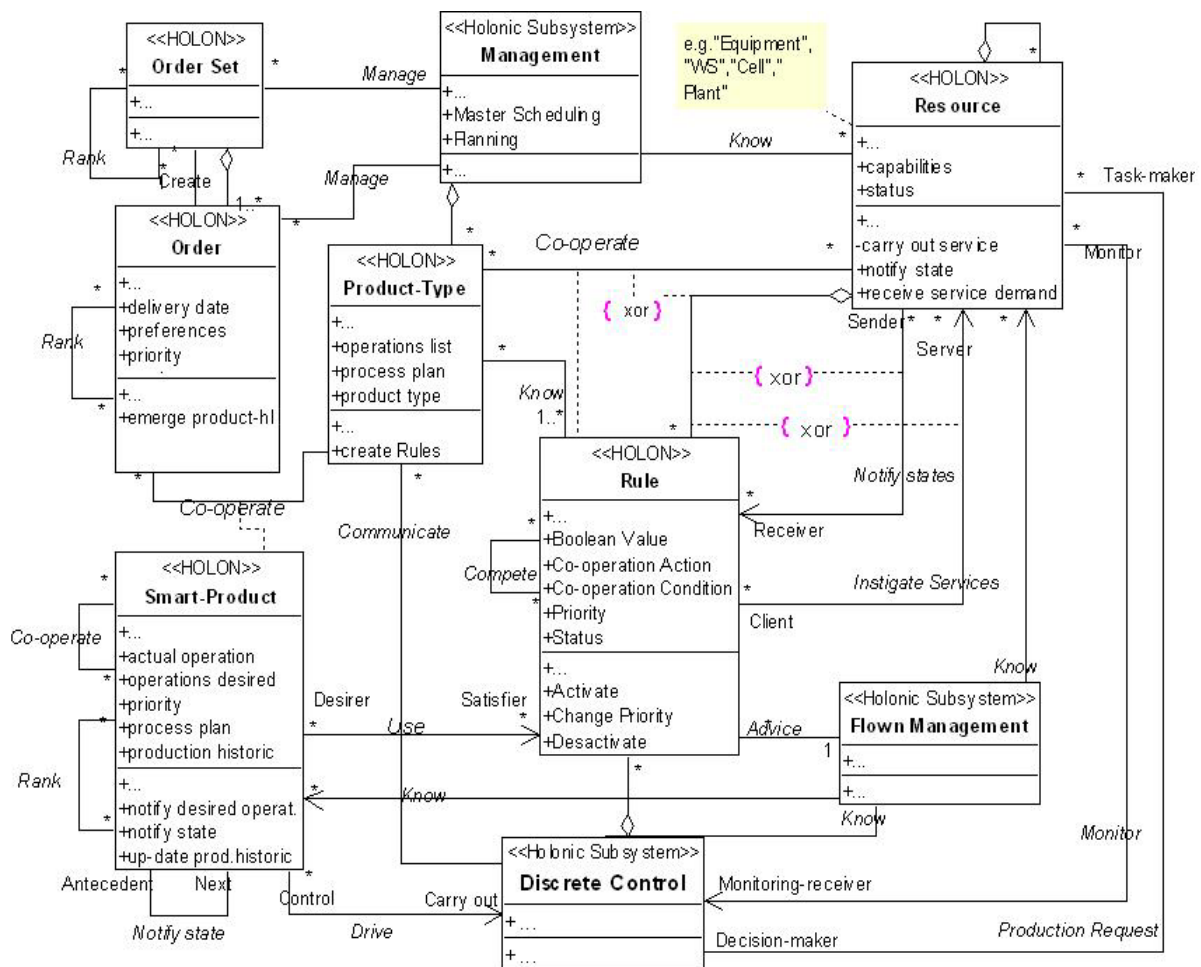


Figure 64: Holarchy with Smart-Product-HLs, Rules, and Resource-HLs.

In the holonic-class diagram presented in Figure 64 is redefined, for example, the relation *Use* between *Rule* and *Smart-Product Holonic-Classes*, as well as it is defined the relation *Know*

between *Rule* and *Product-Type Holonic-Classes*. Also, some other relations are therein redefined, as the *Notify State* (between the *Resource* and *Rule Holonic-Classes*) and the *Instigate Services* (between the *Rule* and *Resource Holonic-Classes*).

Additionally, the Holonic Class Diagram in Figure 64 presents the follows relations in the holarchy: (a) relations between the HDCS class and some class of holons, namely Smart-Product and Resources; (b) the relation between HDCS and the HFM; (c) the important aggregation relation between HDCS and Rule; (d) the relation “origin” of Smart-Product from an Order and Product-Type relation; and (e) the relation “origin” of Rule from a Product-Type and Resource relation.

In fact, in this diagram, it is also expected to present a notion of the complexity found in a HMS, as well as to present the main defined classes of entities and their relationships to deal with part of this complexity in an open architecture.

## 6.8. Rules Creation

A Smart-Product-HL could know the appropriate Rules to use by means of a related Product-Type-HL, cf. previously presented. However, Product-Type-HLs could effectively be more than an information provider to Smart-Product-HL, they could be even implied in the creation of Rules.

Each Product-Type-HL considers a process plan and also additional information defining what set of Resource-HLs is able to carry out each defined operation (desire). This information could be used by Product-Type-HLs in the Rules creation, as stated in the interpretation of Fusaoka’s equation<sup>115</sup> presented in Equation 7.

D	=	Method of Resource-HLs
		Attributes of Resource-HLs
U	=	Unknown Rules
G	=	A Rule for each operation relate to a set of Resource-HLs

**Equation 7: Rule Creation.**

Each Product-Type-HL could create Rules (respecting the Equation 7) likewise defined in section 4.10 as a response to the incognito  $U^{116}$ . However, the intention of this section is not present the method to creation of Rules, but only formalize the openness of the solution to this automatism, reinforcing that this presented solution (a self-contained control meta-model) is open to systemic integration.

## 6.9. Summary

In this chapter it was proposed a way to improve the generic architecture to take into account product-driven aspects. It was discussed contributions of a smart-product orientation to the solution before proposed. Also, in the other sense, it was discussed contributions to the rule and agent-oriented approach in the context of product-driven control.

In a product-driven control, basically, smart-products search for a set of resources to carry out each one of its production need. Opportunely, the cooperation of each Resource-HL set, to carry out a production step related to a set of product-types, is organized and regulated by

<sup>115</sup> The Fusaoka’s formulation consists in:  $Dynamics \wedge Unknown\ Control\ Rules \supset Goal$ , here summarized as  $D \wedge U \supset G$ .

<sup>116</sup> Still, each Product-Type-HL could only considerate appropriate Rules when they are previously created in the holarchy.

means of Rules in the previously proposed solution. Thus, the *upgrade* to a product-driven solution was direct. In short, Smart-Product-HLs individually allocate appropriate Rules to carry out their production desires.

The solution improvement results in an architecture for product and rule-driven control. In this solution, it is established that part of control information is given by the Smart-Product-HL and other part by the Rules. In relation to the process and rule-driven control, the experiment and models carried out allow observing certain transfer of some process knowledge before putted in Rules to the Smart-Product-HLs.

The improved architecture brings some *mutual* advantages in the rule-driven control and product-driven control points of view. For instance, in the product-driven case the Rules permit certain organization and regulation of the control, while in the rule-driven case this approach can allow reduced number of Rules and even smaller Rules than a process-driven control.

Also, Smart-Product-HLs, by themselves, can give more agility and fault tolerance possibility to the control implementations. The potential Smart-Product-HL benefits are compatible with a rule-oriented control approach for reaching holonic goals. However, as some liability, the appropriate composition and the appropriate information keeping for Smart-Product-HLs, even if effectively feasible, seem to be non-easy in technical terms.

The architecture proposed for product and rule-driven control is considered generic. In fact, this reached architecture also follows an orientation to (self) similar entities with similar interactions. For example, the news elements in this architecture, i.e. Smart-Product-HLs, are similar at interface level. In this sense, each Smart-Product-HL, similarly to any Smart-Product-HL, allocates Rules for reaching its production desires.

Summarily, in this thesis, it was presented a solution to control meta-model. The solution properties allow considering it as a meta-model for holonic control open to compose instances with correctness features and systemic integration. The solution is also versatile because it allows composing holonic control systems both in process and product-driven approaches.

The next chapter additionally presents three correlated case studies of HDCS over Resource-HLs simulated in ANALYTICE II. The first case study is related to a machine-cell, the second to an assembling-cell, and the third to a plant environment. These case studies are more a way to present the generality of the architecture as well as to present different manufacturing domains and contexts of the solution application.





## **Chapter 7**

---

Case Studies



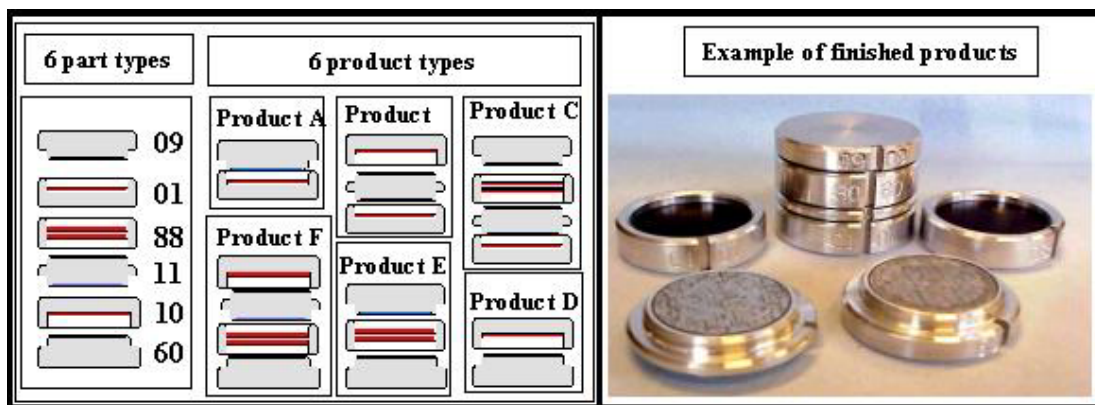
# Chapter 7 : Case Studies

## 7.1. Introduction

The proposed solution to control has been considered as generic due to the (self) similarity in groups of instances that compose each control system. In this chapter, it is presented some case studies aiming to still exemplify the generality of the proposed solution. The examples are concerned to a training environment, called AIPL-PRIMECA, related to the CRAN (*Centre de Recherche en Automatique de Nancy*) and UHP (*Université Henry Poincaré*)<sup>117</sup>.

AIPL-PRIMECA aims put trainees and trainers in a realistic Business (ERP) to Manufacturing (MES) integration context over a production environment. At AIPL-PRIMECA, there exist sub-environments to the training of specific subjects, such as machining and assembling.

Some sub-environments at AIPL-PRIMECA are correlated generating a dynamic similar to that found in real enterprise. For example, there is a sub-environment where it is machined 6 types of base parts from a product family. These parts serve as base ones to compose six types of products assembled in another sub-environment. Both types parts and products are presented in Figure 65.



**Figure 65: Product produced at AIPL-PRIMECA area.**

A model of a plant or *area*, depicted in Figure 66, was made based on the dynamics found in the AIPL. In short, the assemblage product would be demanded in a just-in-time way, generating an appropriate chaining of production about base-part and also about rough material for base-part manufacturing.

Three cases of holonic modeling and design/simulation within ANALYTICE II were developed based on that model. The first and second cases are related to process-cells presented in the model, respectively the automatic-cell (e.g. for machining parts type 01) and the assembling-cell (e.g. for assembling products type A). The third case is related to the presented area comprising substantial elements of that model.

Actually, these three testing efforts are presented as an additional experimental way to still demonstrate the generality and reusability, in diversified contexts, of the proposed solution to holonic control.

<sup>117</sup> *Ateliers Inter-établissements de Productique Lorrain*, see <http://www.aipl.uhp-nancy.fr/lorraine/>.

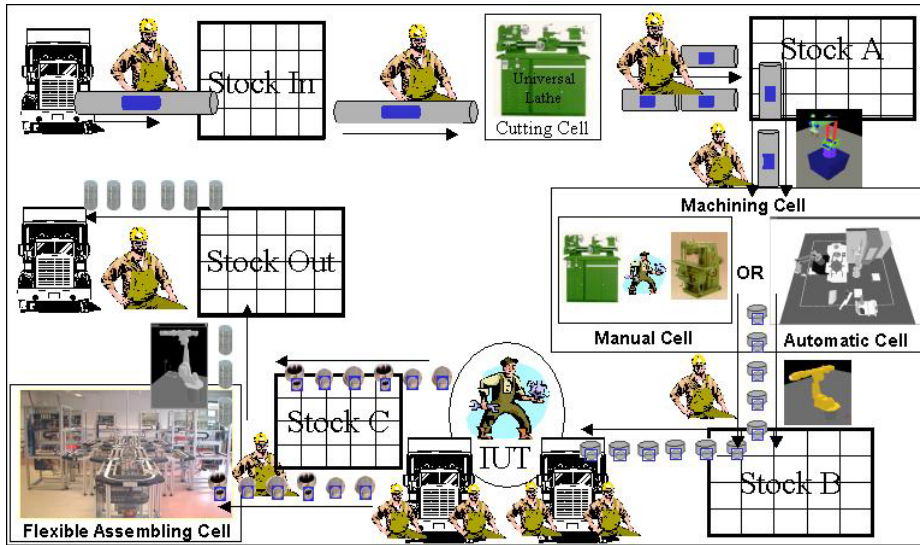


Figure 66: A similar AIPL-PRIMECA context.

## 7.2. Machining System

### 7.2.1. Introduction

The first study case is about the same aforementioned machining cell, but modeling, and partially designing and simulating another control instance. This control instance would be for producing other types of parts. The machining cell would produce parts as those presented in Figure 65. In fact, these types of parts are produced at scope of AIPL within similar machines to those simulated in the designed cell (i.e. lathe and machine-tool), but without the designed automatic transport system.

This designed system has machine flexibility to produce the four types of parts (09, 01, 88 or 11) by changing the setting of the machines (e.g. CNC and tools). However, it almost does not have production flexibility in the sense of alternative machines to manufacture those types of parts (i.e. weak routing flexibility). Also, the system has some buffering flexibility because parts can be buffered in the Store 3x3 or on the Tables 1, 2 or 3. The possible *part states* within the designed system are modeled in the state diagram presented in Figure 67<sup>118</sup>.

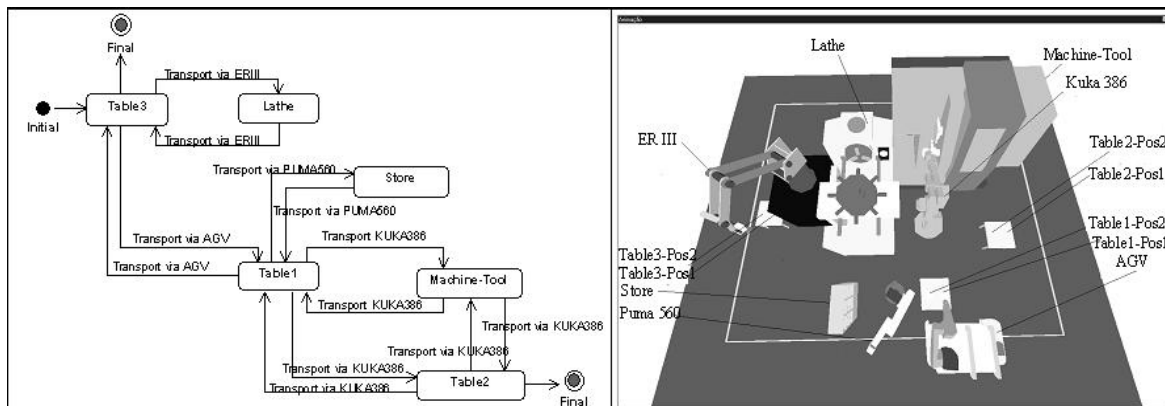
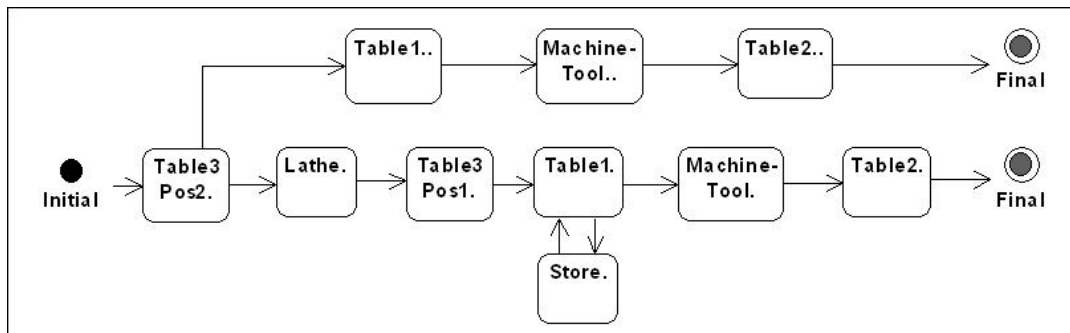


Figure 67: Possible production ways.

<sup>118</sup> These states are those allowed by the physical system, but some of them can be prohibited to determined parts by their process plans.

The types 09, 01, 88, and 11 are a family of parts. The shape designed to this family of parts *demand*s lathing and milling operation. The lathing operation would be the first to be carried out. Thus, the first production step could be the *buffering on table 3* whose subsequent step should be *operation in the Lathe*. The operation carried out by the Lathe could be also carried out by the Machine-Tool. However, they should be rather carried out in the Lathe due to technical reasons.

Each part would be produced following a respective process plan, which should respect the production ways defined by the state diagram above. For example, in Figure 68, it is presented the process plan to Parts 01 and 09 by means of another state diagram, respecting those delimited production steps. The process plan comprises the alternatives of machining and also the alternative of buffering, being the main production way centered in the state diagram.



**Figure 68: Process Plan for the Part from type 01 or 09.**

### 7.2.2. Process-driven Control Simulation

A control instance of the holonic meta-model proposed was effectively created and applied. It was created and applied over the simulated and holonified-machining-cell, within ANALYTICE II, to simulate the production of parts type 01. The Rules, the essence of the control, were created based on the main process plan alternative designed in the previous subsection. The Rules created are presented in Figure 69. These Rules had allowed the machining simulation of a real product by controlling the Resource-HL cooperation.

In this simulation, it was considered the realistic time for the carrying out of operations in machining resources. During the simulation, some statistical results were taken. Briefly, the system was productive in 83,68% of time. This result means only that loading and unloading time between some resources must be optimized, i.e. the internal capacity of Resource-HLs. In the control viewpoint, the control realized its function, i.e. to control the Resource-HL cooperation.

This control was carried out in a process-driven way, where it was not considered flexibilities in the production. In fact, this experiment involving a control instance was made firstly to carry out a simulation related to a real academic product in ANALYTICE II and mainly to demonstrate again that the hypothesis of (self) similar entities facilitate the reuse and application of the solution, in agreement with statements of the concerned bibliography.

Anyway, the control could be implemented with alternative Rules envisaging a better exploitation of flexibilities in the holonified-machining-cell. The flexibility exploitation could be related to:

- The machining flexibility, e.g. to produce the other part types.
- The buffering capacity, e.g. to treat failure or priority of parts.

- The low routing flexibility, e.g. to reach production alternatives.

In the case of adaptability by means of flexibility exploitation, as already argued, a product-driven control would be more profitable. The next subsection exemplifies these potential improvements for product-driven control in the considered holonic-machining-cell.

<p>Rule M1</p> <p>If</p> <p>ERIII.1      Status= Free</p> <p>Lathe.1      Status= Free</p> <p>Table2P.3   Pos1 = Busy</p> <p>Then</p> <p>ERIII.1 Transport(Table2P.3 Pos1, Lathe.1)</p> <p>Lathe.1 Prepare_to_Receive(Part01)</p>	<p>Rule M4</p> <p>If</p> <p>KUKA386.1      Status= Free</p> <p>Machine-Tool.1      Status= Free</p> <p>Table2P.1      Pos1 = Busy</p> <p>Then</p> <p>KUKA386.1 Transport(Table2P.1 Pos1, Machine-Tool.1)</p> <p>Machine-Tool.1 Prepare_to_Receive(Part01)</p>
<p>Rule M2</p> <p>If</p> <p>ERIII.1      Status= Free</p> <p>Lathe.1      Status= Part_Finished</p> <p>Table2P.3   Pos2 = Not_Busy</p> <p>Then</p> <p>ERIII.1 Transport(Lathe.1, Table2P.3 Pos2)</p>	<p>Rule M5</p> <p>If</p> <p>KUKA386.1      Status= Free</p> <p>Machine-Tool.1      Status= Part_Finished</p> <p>Table2P.2      Pos1 = Not_Busy</p> <p>Then</p> <p>KUKA386.1 Transport(Machine-Tool.1, Table2P.2 Pos1)</p>
<p>Rule M3</p> <p>If</p> <p>AGV-Robot.1      Status = Free</p> <p>Table2P.3      Pos2 = Busy</p> <p>Table2P.1      Pos1 = Not_Busy</p> <p>Then</p> <p>AGV-Robot.1 Transport(Table2P.3 Pos2, Table2P.1 Pos1)</p>	<p>Rule M6</p> <p>If</p> <p>KUKA386.1      Status= Free</p> <p>Machine-Tool.1      Status= Part_Finished</p> <p>Table2P.2      Pos1 = Busy</p> <p>Table2P.2      Pos2 = Not_Busy</p> <p>Then</p> <p>KUKA386.1 Transport(Machine-Tool.1, Table2P.2 Pos2)</p>

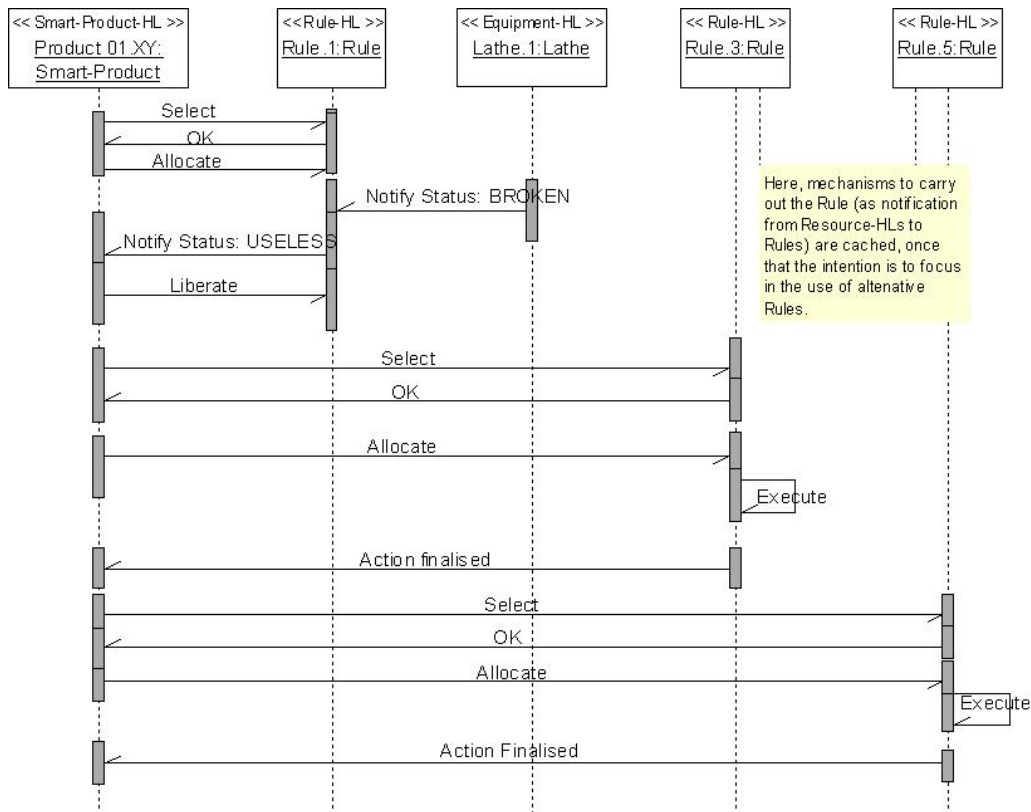
**Figure 69: Similar Rules for a control instance.**

### 7.2.3. Product-driven Control Modeling

In the improvement of the control instance presented to a product-driven control, the use of Smart-Product-HLs could allow the exploitation of the machining flexibilities once each Smart-Product-HL could set the Processing-HL. The use of Smart-Product-HLs could also allow using alternative Rules, which means to use routing flexibilities. A Smart-Product-HL can perceive if a main desired Rule is not enabled and then use a foreseen alternative Rule.

In the case studied, for example, if a Smart-Product-HL for Part 01 is on the Table2P.1 for allocating the Rules1 and if the Rule M1 becomes impossible due to a Lathe.1 status such as *Break*, then the Smart-Product-HL should try another alternative Rule. An alternative Rule would be the Rule M3 and then the Rule M5 for reaching the Machine-Tool.1, in agreement with its process plan. This dynamic is depicted in the holonic sequence diagram in Figure 70.

On the other hand, the *holarchy* can forbid the use of alternative Rules. For instance, in mechanical terms, it is not so interesting to make the ‘lathing’ in the Machine-Tool. However, in some specific situation, the Rule could be enabled by some entity in the holarchy as an exception. For example, if the Lathe is broken in a moment that only two parts should be produced to finish a large lot (that must be delivered in few time), the alternative Rule could be approved by the HFM (Holonc Flow Management).



**Figure 70: Alternative Rule.**

Other flexibility that should be explored by a product-driven control is the buffering flexibility. For example, in the case of failure of the Machine-Tool, the product on the Table.2 could be transferred to the Store. Therefore, the production in the Lathe could happen normally, until the Store be full. Maybe, during this time the Machine-Tool could be repaired. Still, the buffering flexibility could be useful for ordered production where the products must be produced in certain rank.

Normally, Smart-Product-HLs would be produced in the appropriate rank foreseen in the master scheduling, in the Holonic Scheduling. But, in abnormal case, maybe it would be necessary to reorder the exits. An example is a Smart-Product-HL that arrives in the Machining Cell before another previous one. In this case, it could use appropriate Rules to be processed and after that it could use appropriate Rules to be buffered in Store3x3.1, waiting for the previous processing be made.

When a Smart-Product-HL goes out of system, it would notify its successor-Smart-Product-HL. Once a Smart-Product-HL is *authorized* (by a notification) and processed, it could exit of the system, else it would stay in the system. A Smart-Product-HL processed (when convenient) could ask to its previous Smart-Product-HL its state. If the response were “delayed” it would wait in the system. If the response were “problem” or “no response” then the Holonic Fault Supervision (HFS) should be notified<sup>119</sup>.

#### 7.2.4. Summary

The experiment carried out over ANALYTICE II was about a real and academic product. In the experiment it was created a new instance of the control meta-model. The Rules for

<sup>119</sup> As HDCS reaction (adaptability for fault-tolerance), another Smart-Product-HL can be asked to substitute this problematic one.

composing the process-driven control were easily instantiated due to the similarity of the Rules and trade-off between generality and applicability in the meta-model of the Rules.

Additionally, in this section, a possible improving of the control instance to a product-driven one has been considered. Also, some interaction of the product-driven control with other decisional systems has been described. These considerations have been made to present the potential of the architecture for the creation of instances adaptable by exploiting flexibility via appropriate cooperation.

Actually, the main difficulty of this experiment was to extract data for statistical analysis. This data extraction was motivated by curiosity about the performance of the machining cell designed in ANALYTICE II. Really, a need in ANALYTICE II tool is the development of data extractor and analysis environment.

Anyway, in the scope of this work, the important point is to observe the functionality of the control entities. It is not necessary to observe the quality of the manufacturing design developed and simulated in ANALYTICE II, even if this is important for tool applications in a general sense.

### 7.3. Flexible Assembling Cell

#### 7.3.1. Introduction

The second case study is concerned to a real Flexible Assembly Cell (FAC). This FAC is presented on the right side of Figure 71. This section, firstly, presents the FAC and its holonic interpretation. After that, a smart-product oriented Holonic Supervisory Control (HSC) is presented to the Holonic FAC (HFAC) simulated within ANALYTICE II.

The FAC is an assembly cell, integrated in the manufacturing area of the training AIPL-PRIMECA site<sup>120</sup>. The FAC can assembly the six types of academic products (previously presented in Figure 65) from parts of six types buffered in Workstations (WSs).

This FAC enables flexibility in the routing of product assembly through six WSs: one WS for loading parts on pallets circulating on a conveyor, four similar WSs to make products by part assemblage, and one to unload products assembled on pallets. The WS layout is presented on left side of Figure 71.

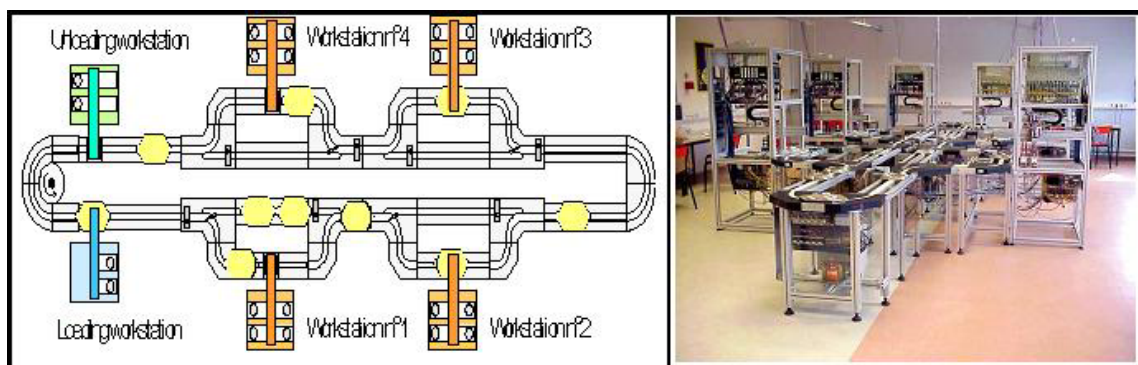


Figure 71: Picture of Flexible Assembly Cell (FAC).

<sup>120</sup> Some FAC research works are integrated in a CRAN research project related to the “Product-driven Controlled Manufacturing Systems”. The goal of this project is to define, develop, and deploy automation methods for the product-driven control and management of industrial processes in order to ensure the traceability of both goods and services along the products life cycle.



FAC architecture is conceptually considered as a HMS for research works while its technical organization, for training works, enables only to store products data in a digital memory embedded on each pallet carrying four products. Automatic Identification of pallets enables to control the routing of products in assemblage, using Conveyor switcher, through the WS controllers networked by a field-bus.

### 7.3.2. Holonic FAC

In this case study, the FAC is considered completely holonic (HFAC) and products *are* comprised in Smart-Product-HLs. The base parts buffered in WS are related to passive Smart-Product-HLs (i.e. they do not drive the production) while the product assembled by base parts are related to active Smart-Product-HLs (i.e. they drive the production).

Also, in the Holonic FAC (HFAC) resources are comprised in Resource-HLs. The Figure 72 presents the Resource-HLs that compose the Holonic FAC (HFAC) and even the class hierarchy in which the *Holonic-Classes* of these instances are found.

A *Holonic-Class WS-FAC* is proposed whose instanced Resource-HLs are WorkStation.0, WorkStation.1, WorkStation.2, WorkStation.3, WorkStation.4, and WorkStation.5, respectively related to Loading WS, WS1, WS2, WS3, WS4, and Unloading WS.

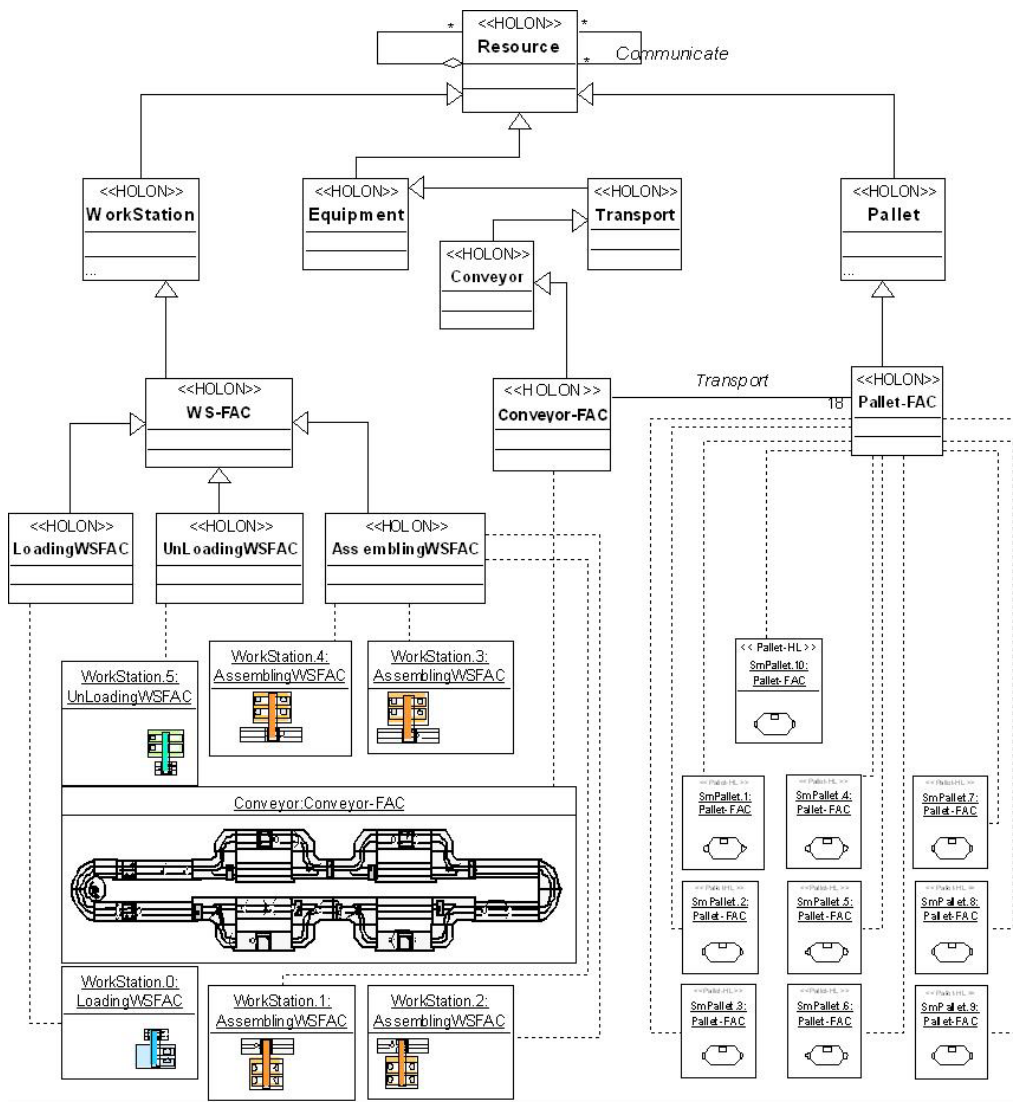


Figure 72: HFAC Resource-HLs

A *Holonic-Class Pallet* is also proposed. It is derived from *Holonic-Class Resource* once their instances are a kind of Resource-HLs. The *Holonic-Class Pallet* is specialized in the *Holonic-Class Pallet-FAC*. The instances of *Pallet-FAC* are Pallet-FAC-HLs.

Specifically, the Pallet-FAC-HLs are the holons SmPallet.1, SmPallet.2, SmPallet.3, SmPallet.4, SmPallet.5, SmPallet.6, SmPallet.7, SmPallet.8, SmPallet.9, and SmPallet.10. Each Pallet-FAC-HL comprises a pallet responsible for transporting products concerned to Smart-Product-HLs.

It is also proposed a *Holonic-Class Conveyor-FAC* whose instance is Conveyor.1. The Conveyor.1 comprises the conveyor that is responsible for the transport and buffering of pallets concerned to Pallet-FAC-HLs.

On the left side of the Figure 73 is presented the buffering regions (called segments) of the store and on the other side of the Figure 73 is presented the position where pallets being transported are identified and stopped. Both regions and position are treated by appropriate Attributes in the Conveyor.1.

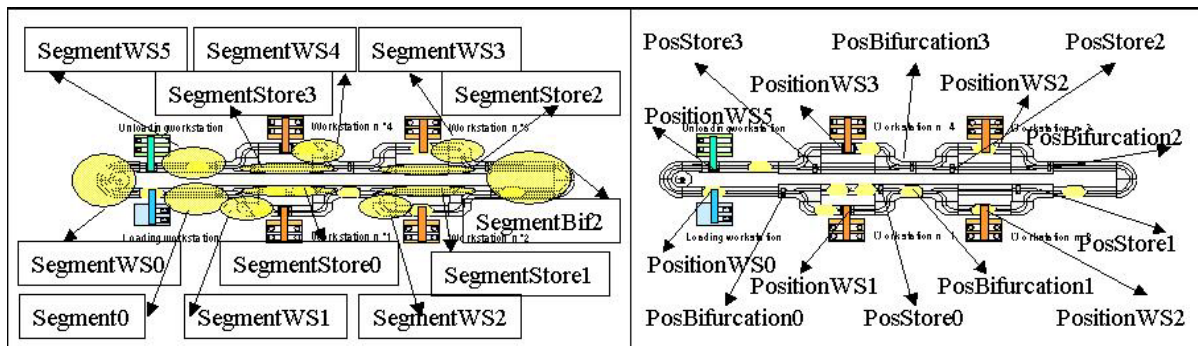


Figure 73: Segments and positions in the conveyor.

### 7.3.3. Resource Holons Development

The emulated equipment necessary to compose the Conveyor-FAC-HL and WS-FAC-HLs were not developed in the ANALYTICE II equipment library. Therefore, it was constructed an emulated conveyor to be the hard-part of the Conveyor.1 and it was constructed an emulated workstation to be the hard-part of each WS-FAC-HL.

The conveyor development gave a supplementary work, not only by the present difficulty to compose equipment in ANALYTICE II (e.g. dependency of graphical module), but also because basic elements (primitives) were not supported to equipment like conveyors, mainly in the graphical animator.

The conveyor was developed to emulate the functional aspect of the real FAC conveyor. It was also developed a graphical model for the conveyor, where graphical pallets (with graphical products) can be transported.

After that was developed the conveyor, it was also developed the Conveyor-FAC-HL (Conveyor.1). It was created Attributes for monitoring positions and segments. Still, Commands were created for requiring pallet liberation and switcher moving.

The conveyor has appropriately worked, emulating the physical comportment of the real FAC conveyor. The respective holon Conveyor.1 also worked as expected. Commands were appropriately instigated and Attributes have appropriately made their notification function.

Commands have transformed its high-level commands in low-level commands understandable by the emulated conveyor, while Attributes have used the information

available in the emulated conveyor to do their monitoring. All this feedback happened using the ANALYTICE II virtual-net.

After the Conveyor-FAC-HL has been developed, the subsequent concern was the development of the WS-FAC-HLs. The first step in the WS-FAC-HLs development was to create the emulated Workstations.

Each Workstation was construct in ANALYTICE II as *black box* equipment that receives a command and (after specific time) gives a feedback about the end of the command execution. At graphical level, it was used a simple table to represent each station, therefore the assemblage in graphical terms was not implemented.

Each WS-FAC-HL encapsulating a Workstation had a unique Attribute implemented. This Attribute is the Status whose state may be, for example, FREE or PALLET-OK<sup>121</sup>. Still, each WS-FAC-HL had also implemented Commands. Each Command demands for a kind of assemblage in the Workstation.

In both implementations of the Conveyor.1 and WS-FAC-HLs, the communication between the emulated-resource and respective virtual resource has been carried out by means of the virtual net.

### 7.3.4. Process Plan

In the HFAC, Smart-Product-HL should use the Resource-HLs for reaching their production objectives. Following the proposed meta-model, a set of Rules would organize the using of Resource-HLs by Smart-Product-HLs. Each Smart-Product-HL would allocate an appropriate Rule for each one of the desired Resource-HL cooperation and this Rule would properly control the desired cooperation.

For creating the Rules for Smart-Product-HL use, it is firstly necessary to define a process plan to each type of Smart-Product-HL. The process plan should indicate what processing resource should be visited to reach production objectives. After that, the Resource-HLs concerned to the carrying out of each production step are identified and then the appropriate Rules are created. The Smart-Product-HL would allocate one Rule associated to its process plan step for reaching its respective production desire.

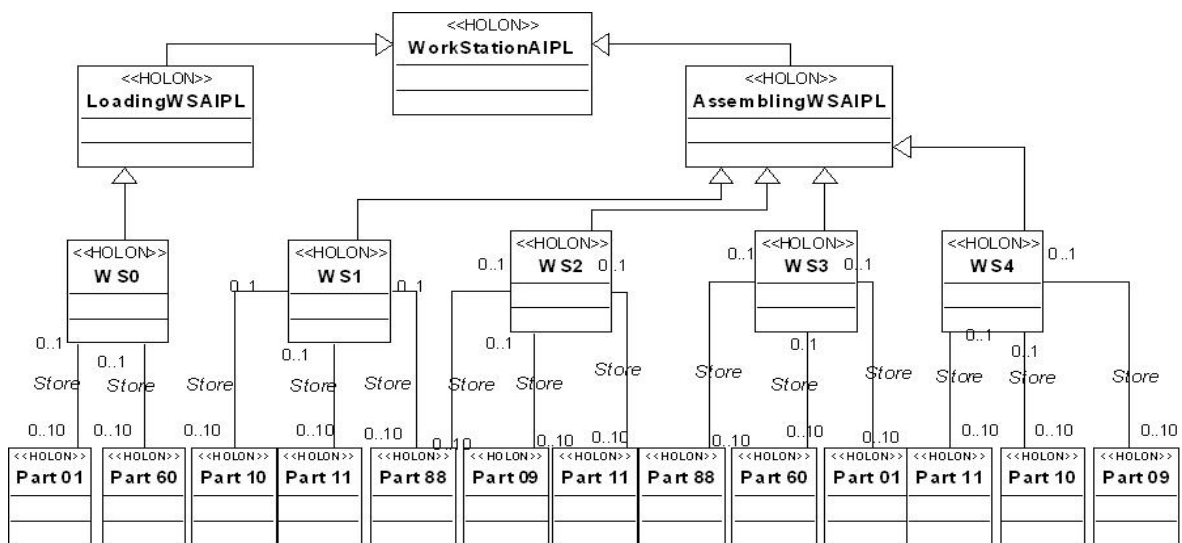
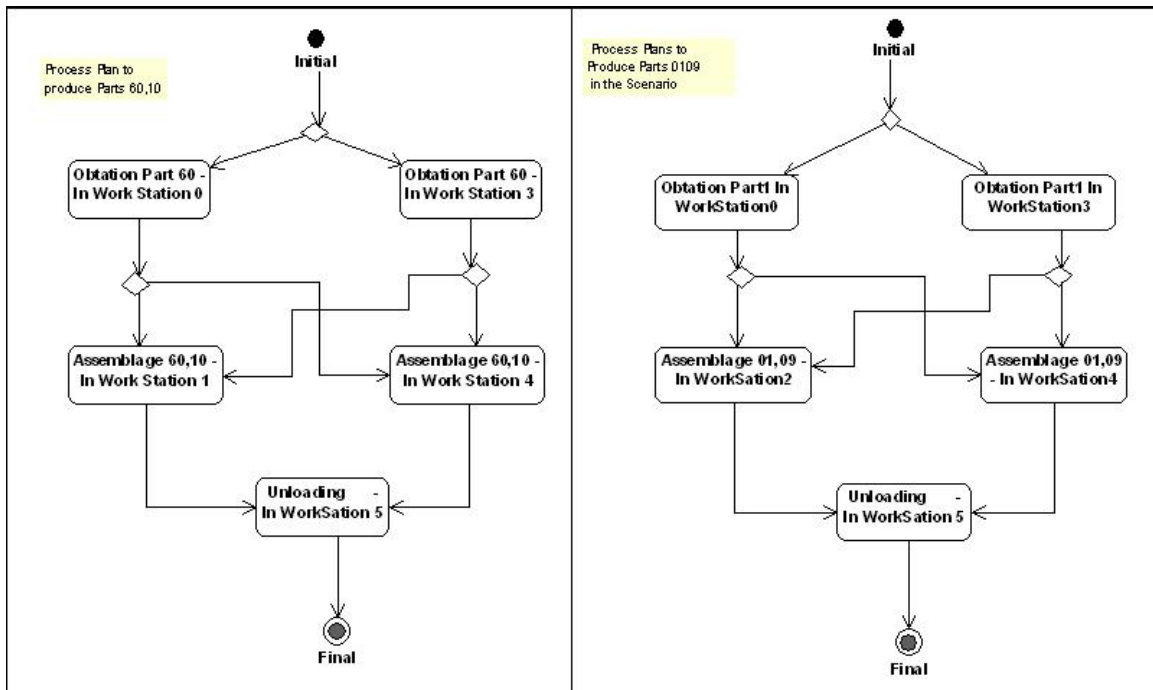


Figure 74: Distribution Scenario of Passive-Smart-Product-HLs.

<sup>121</sup> It was considered that a WS-FAC-HLs always had planned base parts in the simulation.

In this study case, the process plan would define what WS-FAC-HL that must be visited by the Smart-Product-HLs aiming at determined assembling operations. However, before generate process plans, it is necessary to know what kind of base parts are foreseen to each WS-FAC-HL to carry out assemblages. This means that there exists a dependency of scenario, which motivates the definition of the distribution scenario of base parts in the WSs. An example of scenario is presented in Figure 74.

The presented scenario has allowed defining process plans for Smart-Product-HLs of types A and D, respectively exposed on the left side and right side of Figure 75. The process plans for Smart-Product-HL types A and D have the alternatives about processing resources. However, these process plans does not specify the way of transport (the spatial transformation) neither does the buffering (the time transformation)<sup>122</sup>.



**Figure 75: Process Plan to Products of the types A and D.**

The first ramification of the process plan type A can be used as an example for a more detailed specification of the operation, i.e. considering space and time transformation. In this case, this ramification of the process plan A would comprise the follow operations:

- Obtain Part 60 in WS0, Transport from WS0 to Store.0, Wait for transport in Store.0, and Transport from Store.0 to WS2.
- Assemblage 01,09 in WS2, Transport from WS2 to Store.2, Wait for transport in Store.2, Transport from Store.2 to Store.3, Wait for transport in Store.3, and Transport from Store.3 to WS.5.
- Unloading in WS.5.

The more detailed process plans allow identifying what Resource-HLs or parts of the Resource-HLs (in the case of the Store.1) are involved in each operation. Once the set of

<sup>122</sup> In (Gouyon, 2004) it is defined a formal method, based on the theory of automatic synthesis, to define the trajectory (here called process plan) for the classes of Smart-Product-HLs being applicable over the instances. Gouyon has presented, for example, the automatic synthesis of trajectories for the product assemblages in the FAC, considering a specific scenario. In this sense, the process plans proposed in Figure 75 has been formally synthesized in (Gouyon et al., 2004).

Resource-HL cooperation are identified, Rules could be created for Smart-Product-HL utilization.

### 7.3.5. Rules

A product and rule-driven holonic control was instantiated over the Resource-HLs developed for composing the HFAC. The set of Rules was developed based on detailed process plan. The set of developed Rules are presented in Figure 76 and Figure 77. The implemented Rules had used the same C++ structure used in the other models. The difference was the knowledge of the Rules, specifically the Reference (Attribute referenced) in each connected Premise and the Method referenced in each connected Instigation<sup>123</sup>.

<pre> Rule As0 If WorkStation.0 Status      = Pallet-OK Conveyor.1  PositionWS0  = Pallet-UpOn Conveyor.1  Segment0     = Have-Space Then Conveyor.1  Liberate_Pallet(PositionWS0)                 </pre>	<pre> Rule As5 If WorkStation.5 Status      = Pallet-OK Conveyor.1  Segment0     = Have-Space Then Conveyor.1  Liberate_Pallet(PositionWS5)                 </pre>
---	--

**Figure 76: The first part of the set of Rules for the HDCS-HFAC.**

The Rule presented on the left side of Figure 76 deals with the suitable moment to liberate a pallet from the position WS0 while the Rule presented on the right side of Figure 76 deals with the suitable moment to liberate a pallet from the position WS5. A Smart-Product-HL on a pallet can use the Rule As0 for reaching an operation and a Pallet-FAC-HL can use the Rule As5 for reaching on Position WS5. The difference is that the Smart-Product-HL explicit allocates the Rule As0 while the Pallet-FAC-HL is passive in relation to the As5.

<pre> Rule As1.A0 If Conveyor.1 PosBifurcation0 = Pallet-UpOn Conveyor.1 SegmentStore0 = Have-Space Then Conveyor.1 Move_Switcher0(SegmentStore0) Conveyor.1 Liberate_Pallet(Bifurcation0)                 </pre>	<pre> Rule As1.B0 If WorkStation.1 Status      = Free Conveyor.1  PosBifurcation0 = Pallet-UpOn Conveyor.1  SegmentWS1     = Have-Space Then Conveyor.1  Move_Switcher0(SegmentWS1) WorkStation.1 Prepare()                 </pre>	<pre> Rule As2.A0 If Conveyor.1 PosBifurcation1 = Pallet-UpOn Conveyor.1 SegmentStore1 = Have-Space Then Conveyor.1 Move_Switcher1(SegmentStore1) Conveyor.1 Liberate_Pallet(Bifurcation1)                 </pre>	<pre> Rule As2.B0 If WorkStation.2 Status      = Free Conveyor.1  PosBifurcation1 = Pallet-UpOn Conveyor.1  SegmentWS2     = Have-Space Then Conveyor.1  Move_Switcher1(SegmentWS2) WorkStation.2 Prepare()                 </pre>
<pre> Rule As1.A1 If Conveyor.1 PosBifurcation1 = Free Conveyor.1 PosStore0       = Pallet-UpOn Then Conveyor.1 Liberate_Pallet(PosStore0)                 </pre>	<pre> Rule As1.B1 If WorkStation.1 Status      = Pallet-OK Conveyor.1  PosBifurcation1 = Free Then Conveyor.1  Liberate_Pallet(PositionWS1)                 </pre>	<pre> Rule As2.A1 If Conveyor.1 PosBifurcation2 = Free Conveyor.1 PosStore1       = Pallet-UpOn Then Conveyor.1 Liberate_Pallet(PosStore1)                 </pre>	<pre> Rule As2.B1 If WorkStation.2 Status      = Pallet-OK Conveyor.1  PosBifurcation2 = Free Then Conveyor.1  Liberate_Pallet(PositionWS2)                 </pre>
<pre> Rule As3.A0 If Conveyor.1 PosBifurcation2 = Pallet-UpOn Conveyor.1 SegmentStore2 = Have-Space Then Conveyor.1 Move_Switcher2(SegmentStore2) Conveyor.1 Liberate_Pallet(Bifurcation2)                 </pre>	<pre> Rule As3.B0 If WorkStation.3 Status      = Free Conveyor.1  PosBifurcation2 = Pallet-UpOn Conveyor.1  SegmentWS3     = Have-Space Then Conveyor.1  Move_Switcher2(SegmentWS3) WorkStation.3 Prepare()                 </pre>	<pre> Rule As4.A0 If Conveyor.1 PosBifurcation3 = Pallet-UpOn Conveyor.1 SegmentStore3 = Have-Space Then Conveyor.1 Move_Switcher3(SegmentStore3) Conveyor.1 Liberate_Pallet(Bifurcation3)                 </pre>	<pre> Rule As4.B0 If WorkStation.4 Status      = Free Conveyor.1  PosBifurcation3 = Pallet-UpOn Conveyor.1  SegmentWS4     = Have-Space Then Conveyor.1  Move_Switcher3(SegmentWS4) WorkStation.4 Prepare()                 </pre>
<pre> Rule As3.A1 If Conveyor.1 PosBifurcation3 = Free Conveyor.1 PosStore2       = Pallet-UpOn Then Conveyor.1 Liberate_Pallet(PosStore2)                 </pre>	<pre> Rule As3.B1 If WorkStation.3 Status      = Pallet-OK Conveyor.1  PosBifurcation3 = Free Then Conveyor.1  Liberate_Pallet(PositionWS3)                 </pre>	<pre> Rule As4.A1 If Conveyor.1 SegmentWS4     = Free Conveyor.1 PosStore3       = Pallet-UpOn Then Conveyor.1 Liberate_Pallet(PosStore3)                 </pre>	<pre> Rule As4.B1 If WorkStation.4 Status      = Pallet-OK Conveyor.1  SegmentWS4     = Have-Space Then Conveyor.1  Liberate_Pallet(PositionWS4)                 </pre>

**Figure 77: The second part of the set of Rules for HDCS-HFAC.**

<sup>123</sup> The notification mechanism had worked as before, once the same software components were used. The mechanism was also traced via Microsoft Visual C++ debugger. Still, all sharing of co-operations (e.g. notification from Attributes between Premises) have worked as expected.

The Rules presented in Figure 77 are divided in four groups (As1, As2, As3, and As4). Each group firstly allows a Smart-Product-HL chooses between a Rule of type AsX.A0 or AsX.B0. The first and second Rule type permits the Smart-Product-HL to respectively reach a Workstation or a Store. If the AsX.A0 is chosen then the next Rule to be allocated is the As1X.A1. Similarly, if the AsX.B0 is chosen then the next Rule to be allocated is the As1X.B1.

### 7.3.6. Smart-Product Holons.

In the simulation experiment carried out, the simulated Smart-Product-HLs were related to products type A and D. Each ProductA-HLs or ProductD-HLs was graphically represented by a cube. Each type implemented has a cube with different color that has permitted to observe in the graphical animator if the Smart-Product-HL has gone to the good direction.

The base parts have been not simulated, it was considered that they were always available and that the assemblages, loading, and unloading operation had been always well made. The aspects of historic (in traceability issues) were not implemented. Still, it was used eight Smart-Pallet-HL, each one transporting only one Smart-Product-HL. Therefore problems about merge of Smart-Product-HL desires did not happen.

An interesting phenomenon in this assembling case is that when a Smart-Product-HL is created, it does not have a physical part. Only after a first operation that the Smart-Product-HL acquires a physical part once at least a base part is associated. In the simulation, after assemblages over a first associated part only the respective holon is updated about, the product is not graphically changed.

The updating or synchronization of the soft part of a Smart-Product-HL about its hard part was made in the following way:

- Each time that a pallet arrives in a conveyor position, the conveyor detects what pallet was there and sends this information to the virtual-net.
- That information serves to, firstly, update the position of Smart-Pallet-HL hard-part in its soft-part and, secondly, this information serves to update about the associated Smart-Product-HL hard-part in its soft-part.

### 7.3.7. Production Flexibility

Each Smart-Product-HL, based on predetermined knowledge, knows what are the Rules to be selected for a possible allocation. In this sense, to test the capacity of taking alternative Rules, the segments relative to WorkStation.1 and WorkStation.2 (on the Conveyor.1) were loaded.

Those segments in loaded state had forced Smart-Pallet-HLs take a production alternative, based on the desires of related Smart-Product-HL. For example, a Product.D-HL (on a Smart-Pallet-HL) at the PosBifurcation0 tries to select/allocate the Rule As1.B0 (the preferred Rule) for reaching the WorkStation.1, but the Rule is false (because the segment is full), then it automatically searches a second determined option, that is the Rule As1.A0.

Once the Rule is effectively allocated, this Rule allows it going ahead, selecting/allocating (in appropriate moment) the subsequent Rules (namely As1.A1, As2.A0, As2.A1, As3.A0, and As4.B0) that allow it arriving in the alternative means of production (i.e. the holon WorkStation.4).

This simulation is not quantitative presenting some number or statistic. In fact, it is a qualitative simulation where is observable if the holons can carry out some agility properties, such as to explore the routing flexibility.

Additionally, other Rules could be created to evaluated if the segment for reaching a WS is semi-loaded and if another segment for reaching an alternative WS is non-loaded, thereby establishing a coordination way to reach the alternative WS. After that, concerned Smart-Product-HL could consider this Rule as preferential to a determined production step once it could generate a more equilibrated use of WSs.

### 7.3.8. Selection/Allocation Process

In this study case it was implemented an alternative way to the selection/allocation mechanism. Firstly, it was developed an Attribute relative to each position of Conveyor.1, generically called ProdRenter. An Attribute ProdRenter (e.g. ProdInPosBif0) knows what kind of product is in a specific position.

Secondly, it was developed a Premise to each possible combination of comparison between a ProdRenter and the kinds of Smart-Product-HLs possible, e.g. **Holon** Conveyor.1 **Attribute** ProdInPosBif0 = "Product.A-HL".

Once that a Premise like that was connected to a Rule (e.g. As1.B0), this automatically invalidate the approbation of this Rule to other kinds of Smart-Product-HL, but permitted to the Smart-Product-HL in concerned position *to allocate* the Rule with exclusivity.

As consequence, other similar Rules were created to deal with the allocation of other type of Smart-Product-HLs. In fact, the allocation of a Rule was handled by an artifice with a dedicated Rule to each kind of Smart-Product-HL. As result, this has augmented the number of Rules.

The performance of decision is not so affected by this solution, due to the notification mechanism, but the solution was so complicated. This implemented solution is not considered a good tactic of design that had took certain time of development.

The mechanism must be implemented in the other way, as previously defined. In short, the Smart-Product-HL should only monopolies the Rule for the time that it is using the Rule, in a more simplified way.

### 7.3.9. Information for Agility

The architecture or model proposed and simulated for this study case is another application of the meta-model defined during the development of this thesis. A feature to be highlighted about this solution proposed is that it does not change the structural bases of the elements of the considered system (e.g. the FAC), it only augments the level of information, integration, and control about these elements, for aiming to obtain a better resource utilization.

Actually, in some MS such as in the FAC of the study case, the specific coordination of resources may be made by means of an ordinary interlocked physical system. However, the real-time information extracted from each resource in the respective Resource-HL, which is notified to appropriate Rules, actually allows Smart-Product-HLs infer the appropriate production means from present and enabled ones.

Each Smart-Product-HL can choose a production way, among the effectively enabled ones, also based on some fixed parameters or variant parameters from the holarchy. In fact, these parameters are used by Smart-Product-HLs to orient their decision for reaching a production

performance goal. An example of production goal is the equilibrated production in the workstations. In this type of case, the main advantage of the proposed control solution, in short, is the informational support for more decisional capacity intending holonic issues.

The point is that the flow can be optimized if the Rules knowledge is appropriately profited by the Smart-Product-HL. Smart-Product-HLs can, for example, compete by a Rule based on their priorities. The left side of Figure 78 exemplifies two Smart-Product-HLs where the Smart-Product-HL with higher priority would win the competition for a Rule, forcing the loser to search for another Rule.

The priorities of Smart-Product-HLs or their preference degree by Rules can be changed by holarchy entities. For instance, the HFM could change this parameter aiming, for example, to optimize the Resource-HLs use, reduce the times of production or/and benefit the production of more lucrative product set.

An additional way to complementarily improve the Smart-Product-HL deliberation would be the approbation prevision. This mechanism would permit each Rule to know the probably state of a resource in certain time with certain probability.

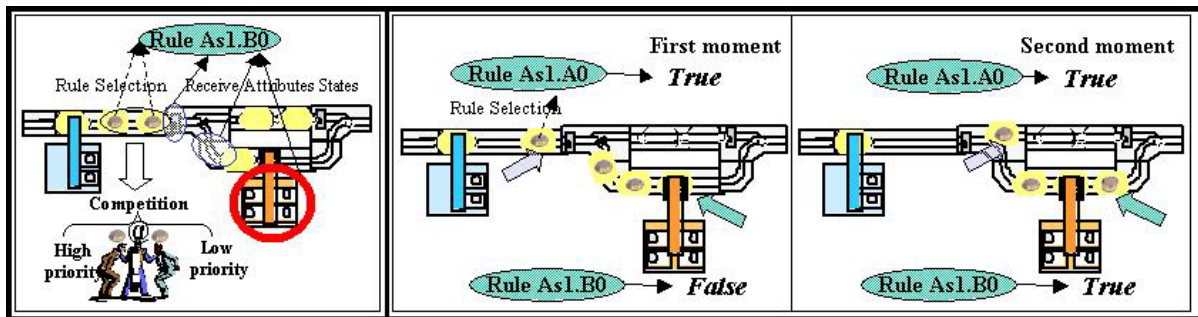


Figure 78: Flow issues.

In the right side of Figure 78, it is presented an example where a Smart-Product-HL chooses a less interesting Rule because the more interesting Rule was in a *false state*. It is also presented that just few moments after the execution of that Rule, the more interesting Rule reaches a *true state*. If the approbation prevision was enabled, the Smart-Product-HL could deliberate for waiting a little space-time, aiming to use the preferential Rule.

For carry out the approbation prevision mechanism, each Attribute could preview a state changing in certain time and in certain probability following internal models. After that, it could notify concerned Premises about the foreseen state. An example is the Attribute Status in WorkStation.1, which knows that the assembling process being executed will be finished in tree seconds and notifies interested ones that its possible future state will be Free with a certainty degree, e.g. 97%.

After that a Premise is notified about a future state, it calculates if its state will be changed. In affirmative case, the Premise would notify related Rules with the future state with the *same* parameters of time and probability. Each Rule that is notified could calculate a prevision of approbation when all connected Premise, in false state, had notified a future state as true. The prevision of time could be the larger received and the prevision of probability could be the worst that was notified.

### 7.3.10. Summary

This case study allowed observing more an application, within ANALYTICE II, of the proposed control solution. This experiment carried out is a way to confirm the property of



generality of the control solution. In fact, the generality has been already stated due to the (self) similarity of entities in the previous presented control instances.

However, this case study is an application over an assembling case, which is different from the machining cases previously considered. In the assembly case, new details appear, as Smart-Product-HLs initially without a hard part and Smart-Product-HLs being composed of passive Smart-Product-HLs.

The implementation carried out has presented similar Rules and (self) similar Resource-HLs working also by means notification from Resource-HL to the Rules and Rules being used by Smart-Product-HLs. Additionally, a differentiated way to the use of Rules by Smart-Product-HLs has been defined, however it has been considered not appropriate. Also, some flow coherence between pallets and virtual pallets has been implemented for reaching the flow coherence between product and virtual-products.

The experiment and models have confirmed that the flexibility exploitation for agility depends upon the information enabled. Additionally, some mechanisms to the control instance have been presented for reaching a better availability and better utilization of the information about the assembling cell. For example, the mechanism of approbation prevision has been described. Summarily, some new *horizons* in the use of solution have been defined.

The experiment carried out also serves to present the potential of the ANALYTICE II to carry out experiments, as well as to identify its present drawbacks. In fact, it was harder to construct the emulated assembling system within ANALYTICE II than instantiate the Resource-HLs and the related product and rule-driven control.

## **7.4. AIPL Case Study**

### **7.4.1. Introduction**

At this moment, the meta-motel for HDCS has been applied and considered in the scope of process-cells. It was elaborated a HDCS for a holonified-machining-cell and another HDCS for a holonified-assembling-cell within ANALYTICE II. Each one of these HDCSs is a type of Holonic Supervisory Control (HSC) for the holonified-process-cells.

A holonified-process-cell and its HSC can compose a high-level Resource-HL, i.e. a Process-Cell-HL. A Process-Cell-HL would have Attributes and Methods, like any Resource-HL. The Attributes of Process-Cell-HLs could be based on the Attributes of the *encapsulated* Resource-HLs. The Methods of Process-Cell-HLs could be similarly based on the Methods of the *encapsulated* Resource-HLs.

In the plant context, the Resource-HLs, e.g. Process-Cell-HLs and Equipment-HLs, would cooperate by means of a HDCS. The HDCS at plant level would be a Holonic Shop Floor Control (HSFC) or Holonic Manufacturing Execution System (HMES) working likewise a Holonic Supervisory Control (HSC).

A HSFC or HMES would be composed of a set of Rules, which would control the cooperation of Resource-HLs based on states that are notified from their Attributes and on instigation of their Methods. Furthermore, the Rules would be used by Smart-Product-HLs for reaching production desires, in the case of a product-driven HSFC or HMES.

The application of the meta-model at higher level in the factory hierarchy is clear due too the (self) similarity of entities of control. Even the Resource-HLs (whose concerned resources are from different nature) are (self) similar at external scope. Furthermore, above the Equipment-

HL level, the internal compositions of Resource-HLs are similar like in fractals, i.e. at least a HDCS working over Resource-HLs.

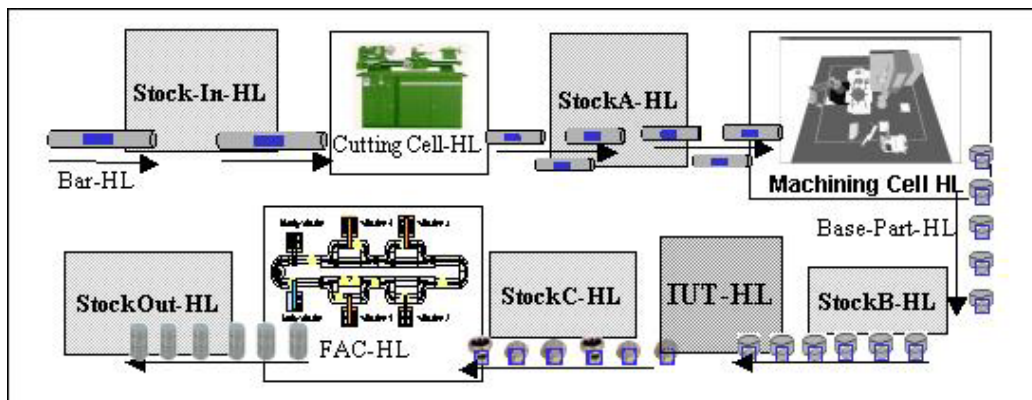
Concordantly, during the development of the thesis, it was suggested the use of the meta-model in a higher level than process-cell level. In this sense, an experiment has been developed in ANALYTICE II to simulate a factory-like, aiming to demonstrate the meta-model applicability in a more *concrete* way. A HSFC has been effectively instantiated based on the meta-model proposed for this simulated factory-like. The simulation was concerned to the production scenario inspired by AIPL and described at the beginning of this chapter.

#### 7.4.2. Simulation Context

The simulation of a factory-like comprises part of proposed production scenario based on AIPL context. The entities and dynamic in the simulation experiment are depicted in Figure 79. Each simulated resource is encapsulated in a respective Resource-HL. The main automatic Resource-HLs are Process-Cell-HLs and Sock-HLs. The transport between Resource-HL would be carried out by means of operators.

The simulated factory is composed of a cutting-cell, a machining-cell (like that from the first case study), an IUT (like a *finishing-cell*), and an assembling-cell (that would be the FAC from the second case study). For each one of the process-cells, there exist a store for buffering rough products and another store for buffering processed products.

In this designed production environment, metal bars are an input to the system, which arrive in an input store (Sock In). In the cutting-cell, the metal bars from the input store are cut generating shorter metal bars. The short bars are transported and buffered in the appropriate store (Sock A). After that, the short bars are transported to the machining-cell that processes them, generating base-parts.



**Figure 79: The sketch of the simulated plant in ANALYTICE II.**

The processed base parts are transported to a subsequent store (Stock B). After that, they are sent to the IUT for finalizations. After the IUT finalization, the parts are transported and buffered in another store (Stock C). The finalized parts are transported to WSs in the FAC where they serve as base part to assembling products. The assembled products are finally transported to a final store (Stock Out).

For the simulation experiment, it was chose a context of just-in-time production, inspired by the concept of customized production, for aiming to test some agile features. The idea is to produce 6 types of products within the FAC-HL without foresight and consequently to process the base parts and bars without foresight too.

The FAC-HL should start the assembling of a product due to an order that arrives in the holarchy. After that, the holarchy, by means of a HSFC, should perceive the need for base part replacement in the FAC (i.e. in its WSs). This perception could be based on the assemblages being carried out.

After that the HSFC determines the replacement needed, the HSFC should manage the chain of production determining when finalize base-parts, produce base-parts, and produce short bars. Also, the HSFC should control the cooperation of Resource-HLs for each Process-Cell-HL to receive the appropriate inputs and appropriately liberate its outputs.

### 7.4.3. Simulation Issues

Actually, in this simulation experiment is aimed to exemplify the applicability of the meta-model for HDCS in a higher level than a process-cell. However, the entire development of production environment, in a fine scale of detail, would be impracticable in little time and even out of the experiment goal. Therefore, in the simulation is made some abstractions focusing the main points related to the experiment goal.

In ANALYTICE II, each processing cell has been simulated as *black box*. The processing-cell receives a part to be processed and after some time the part is considered as processed. Each processing-cell was simulated in the emulation part of the ANALYTICE II having a respective virtual-processing-cell in the control side of the ANALYTICE II.

Due to the ANALYTICE II dependency on graphical model for simulated entities, an artifice used for enabling the experiment was to associate the graphical model of a piece of equipment to each simulated cell. This means that even if the experiment graphically presents “equipment”, the behavior model of the simulated entities are concerned to the process-cells<sup>124</sup>.

Also, for simulating each stock, one piece of equipment Store 3X3 was used. Each Store 3X3 had an equivalent virtual Store, composing Stock-HL in the simulation environment. All Equipment-HLs and Processing-Cell-HL created in this experimental effort have been its hard and soft parts synchronized using the virtual-network.

In this case study, the simulated parts are buffered in the Stores and processed in the Processing-Cells. The part that arrives in the processing cell (whose geometric model is a piece of equipment) is graphically hidden inside of cell, which reappears at the end of the related processing. Also, each type of part (bars, cut bars, processed part, finalized parts, and assembled parts) has been graphically represented by simple cube, which were differentiated by means of colors.

The differentiation of parts by means of colors permitted to graphically observe what type of product were produced. This allowed observing that the production of each part was appropriately managed. The production of each part was managed by means of a related Smart-Product-HL that suitably allocated enabled Rules for reaching their production steps.

Another feature of the simulation experiment is the abstraction about the transport system. In this experiment, it is considered that the transport is always enabled, not being it represented in the simulation. For graphically simulate the transport of parts, each part is appropriately hidden in the place of origin and it is appropriately showed in the final place.

---

<sup>124</sup> Graphically, for the cutting-cell, it was used a lathe graphic model, for the machining-cell it was used a machine-tool graphic model, and for the IUT was used a Table. Still for the assembling cell was used a table representing each WS and another table to represent the conveyor.

### 7.4.4. Holonic Control

In this case study, it was created three types of Rules. The Rules of the first type determine the moment to transfer parts from one resource to another resource. The Rules of the second type determines the appropriate situation to start the assemblages in the FAC.

Rules of the first and second types are allocated by Smart-Product-HLs aiming to reach production steps. However, the Rules of the third type are not allocated by Smart-Product-HLs. These Rules identify when Smart-Product-HLs related to base-part (i.e. Base-Part-HLs) should initiate the driven of their production.

In Figure 80, it is presented a set of Rules allocable for Smart-Product-HLs related to assemblages (i.e. Assembling-Product-HLs). In this set, the Rule *Product-Finished* is of the first type while the other Rules are of the second type.

An Assembling-Product-HL is created due to an order of production<sup>125</sup>. An Assembling-Product-HL created would use the suitable Rule for launching itself in the FAC-HL, e.g. an Assembling-Product-A-HL would allocate the Rule *Produce-Product-Type-A*. The allocated Rule appropriately instigates the starting of the self-driven production of the Assembling-Product-HL in the scope of FAC-HL. The Rule also instigates the update about the base-part number already implied in the production.

Rule Produce-Product-Type-B If FAC Part_Type_01_Inside = True FAC Part_Type_11_Inside = True FAC Part_Type_10_Inside = True Then FAC Start_Production_Product-Type-B FAC Decrement_Number_Part-Type-01 FAC Decrement_Number_Part-Type-11 FAC Decrement_Number_Part-Type-10	Rule Produce-Product-Type-E If FAC Part_Type_60_Inside = True FAC Part_Type_08_Inside = True FAC Part_Type_09_Inside = True Then FAC Start_Production_Product-Type-E FAC Decrement_Number_Part-Type-60 FAC Decrement_Number_Part-Type-08 FAC Decrement_Number_Part-Type-09	Rule Produce-Product-Type-A If FAC Part_Type_01_Inside = True FAC Part_Type_09_Inside = True Then FAC Start_Production_Product-Type-A FAC Decrement_Number_Part-Type-01 FAC Decrement_Number_Part-Type-09 Rule Produce-Product-Type-D If FAC Part_type_01_Inside = True FAC Part_type_09_Inside = True Then FAC Start_Production_Product-Type-D FAC Decrement_Number_Part-Type-01 FAC Decrement_Number_Part-Type-09 Rule Product-Finished If FAC Product_Finished = True Store-Out Position_Free = True Then FAC Liberate_Product_Finished. Store-Out Receive_Product_(Type?).
Rule Produce-Product-Type-C If FAC Part_Type_09_Inside = True FAC Part_Type_08_Inside = True FAC Part_Type_11_Inside = True FAC Part_Type_01_Inside = True Then FAC Start_Production_Product-Type-C FAC Decrement_Number_Part-Type-09 FAC Decrement_Number_Part-Type-08 FAC Decrement_Number_Part-Type-11 FAC Decrement_Number_Part-Type-01	Rule Produce-Product-Type-F If FAC Part_Type_60_Inside = True FAC Part_Type_08_Inside = True FAC Part_Type_11_Inside = True FAC Part_Type_10_Inside = True Then FAC Start_Production_Product-Type-F FAC Decrement_Number_Part-Type-60 FAC Decrement_Number_Part-Type-08 FAC Decrement_Number_Part-Type-11 FAC Decrement_Number_Part-Type-10	

**Figure 80: Rules for determining the assemblages in the FAC.**

Once a Smart-Product-HL would be inside of the FAC-HL, it would interact with the Rules and Resource-HLs in its scope. After the Smart-Product-HL has reached its production desire in the FAC-HL, it allocates the Rule *Product-Finished* for reaching the Store-Out. In the allocation of this Rule, the Smart-Product-HL informs its type. This information could be used by the Store-Out for updating its Attributes about products inside. Potentially, these Attributes could be useful to Rules outside of the AIPL scope.

<sup>125</sup> For this simulation experiment it was developed an interface to the users demand for products. After that a product is demanded, a respective Smart-Product-HL is created. In the creation of the Smart-Product-HL, the appropriate Rules to be allocated are informed.

In Figure 81, it is presented a set of Rules of the third type related to Base-Part-HL and Assembling-Product-HL productions. These Rules calculate the appropriate moment to start selves-driven production of Base-Part-HLs and to inform the FAC-HL that the needed base-parts are being provided for it<sup>126</sup>. The calculus is made based on information that is notified from the FAC-HL. This information is about the number of not-reserved base-parts for productions driven by Assembling-Product-HLs<sup>127</sup>.

Rule Start_Smart_Product_For_Part-Type-09 If FAC    Low_Number_Part_Type_09 = True Then AIPL   Start_Smart-Product-HL(Part-Type-09) FAC    Wait-For(Part-Type-09)	Rule Start_Smart_Product_For_Part-Type-88 If FAC    Low_Number_Part_Type_88 = True Then AIPL   Start_Smart-Product-HL(Part-Type-88) FAC    Wait-For(Part-Type-88)
Rule Start_Smart_Product_For_Part-Type-01 If FAC    Low_Number_Part_Type_01 = True Then AIPL   Start_Smart-Product-HL(Part-Type-01) FAC    Wait-For(Part-Type-01)	Rule Start_Smart_Product_For_Part-Type-11 If FAC    Low_Number_Part_Type_11 = True Then AIPL   Start_Smart-Product-HL(Part-Type-11) FAC    Wait-For(Part-Type-11)

**Figure 81: Rules for the starting of the Smart-Product-HLs concerned to base-parts.**

The Base-Part-HLs have some production steps to be reached after the starting of their selves-driven production. The production steps are listed before Rules be presented for reaching these steps:

- Machining related rough bar in the Machining-Cell<sup>128</sup>.
- Buffering of related processed base-part in the StockB.
- Finalization of related processed base-part in the IUT.
- Buffering of related finalized base-part in the Stock C.
- Buffering of related finalized base-part in an appropriate WS of the FAC.

Rule Production_Part-Type-09 If StockB    Low_Number_Parts_09 = True StockA    Cut_Bar_Inside        = True Mach-Cell    Receive_Part_Possible   = True Then StockA    Liberate-Product Mach-Cell    Receive_Product(Part-Type-09)	Rule Production_Part-Type-01 If StockB    Low_Number_Parts_01 = True StockA    Cut_Bar_Inside        = True Mach-Cell    Receive_Part_Possible   = True Then StockA    Liberate-Product Mach-Cell    Receive_Product(Part-Type-01)	Rule Cut_Bar If StockIn    Rough_Bar_Inside       = True Mach-Cell    Receive_Bar_Possible   = True StockA    Position_Free            = True Then StockIn    Liberate_Bar Mach-Cell    Receive_Bar
Rule Production_Part-Type-88 If StockB    Low_Number_Part_Type_88 = True StockA    Cut_Bar_Inside        = True Mach-Cell    Receive_Part_Possible       = True Then StockA    Liberate-Product Mach-Cell    Receive_Product(Part-Type-88)	Rule Production_Part-Type-11 If StockB    Low_Number_Parts_11 = True StockA    Cut_Bar_Inside        = True Mach-Cell    Receive_Part_Possible       = True Then StockA    Liberate-Product Mach-Cell    Receive_Product(Part-Type-11)	Rule Store_Cut_Bar If Mach-Cell    Cut_Bar_Inside       = True StockA    Position_Free            = True Then Mach-Cell    Liberate_Cut_Bar StockA    Receive_Cut_Bar

**Figure 82: Rules for enabling the rough material and production of base-parts.**

<sup>126</sup> The “conscience” about base-part production avoids incorrect calculus in the scope of FAC about the needs of base parts.

<sup>127</sup> In fact, in the experiment, always that a base-part were implied in the production of an assemble product, the respective Attribute *Low\_Number\_Part\_Type\_X* was considered as true. Therefore concerned Rule could be triggered.

<sup>128</sup> For the simulation experiment, it was considered one little bar for the machining of a base-part. Nevertheless, the experiment could generate, for example, two parts for each little bar.

In the Figure 82, it is presented the Rules *Production\_Part-Type-09*, *Production\_Part-Type-01*, *Production\_Part-Type-88*, and *Production\_Part-Type-11*. These Rules allow Base-Part-HLs reaching their first production step. The Base-Part-HL allocates the suitable Rule based on its type.

In the Figure 82, it is also presented the Rules *Cut\_Bar* and *Store\_Cut\_Bar* that enable the rough material for production of base-part. These two Rules control the transferring of bar and short bar respectively before and after their processing in the cutting-cell. These Rules works in a process-driven way, because the invariance of production does not demands Smart-Product-HL in the bar cutting.

Rule <i>Store_Machined_Part</i> If StockB Position_Free = True Mach-Cell Part_Finished = True Then Mach-Cell Liberate-Part-Finished StockB Receive_Product(Part-Type?)	Rule <i>Finalisation_Part-Type-09</i> If StockC Low_Number_Parts_09 = True StockB Machined_Parts_09_Inside = True IUT Receive_Part_Possible = True Then StockB Liberate- Part(Part-Type-09) IUT Receive_Part(Part-Type-09)	Rule <i>Finalisation_Part-Type-01</i> If StockC Low_Number_Parts_01 = True StockB Machined_Parts_01_Inside = True IUT Receive_Part_Possible = True Then StockB Liberate- Part(Part-Type-01) IUT Receive_Part(Part-Type-01)
Rule <i>Store_Finalised_Part</i> If StockC Position_Free = True IUT Part_Finished = True Then IUT Liberate-Part-Finished StockC Receive_Product(Part-Type?)	Rule <i>Finalisation_Part-Type-88</i> If StockC Low_Number_Parts_88 = True StockB Machined_Parts_88_Inside = True IUT Receive_Part_Possible = True Then StockB Liberate- Part (Part-Type-88) IUT Receive_Part(Part-Type-88)	Rule <i>Finalisation_Part-Type-11</i> If StockC Low_Number_Parts_11 = True StockB Machined_Parts_11_Inside = True IUT Receive_Part_Possible = True Then StockB Liberate- Part(Part-Type-11) IUT Receive_Part(Part-Type-11)

**Figure 83: Rules for reaching the finalization of base-parts.**

Subsequently, in Figure 83 it is presented the set of Rules that allows Base-Part-HLs reaching the second, third, and fourth production steps. The Rule *Store\_Machined\_Part* is related to the second step, the Rule *Store\_Finalised\_Part* is related to the fourth step, and the other Rules are related to the third step. Finally, in Figure 84, it is presented some Rules that allow Base-Part-HLs reaching the last production steps<sup>129</sup>.

Rule <i>Store_Part_09_FAC_WS2</i> If StockC Parts_09_Inside = True FAC Parts_09_Wanted_WS2 = True Then StockC Liberate_Part(Part-Type-09) FAC Receive_Part (WS2)	Rule <i>Store_Part_60_FAC_WS0</i> If StockC Parts_60_Inside = True FAC Parts_60_Wanted_WS0 = True Then StockC Liberate_Part(Part-Type-60) FAC Receive_Part (WS0)	Rule <i>Store_Part_88_FAC_WS3</i> If StockC Parts_88_Inside = True FAC Parts_88_Wanted_WS3 = True Then StockC Liberate_Part(Part-Type-88) FAC Receive_Part (WS3)
Rule <i>Store_Part_01_FAC_WS0</i> If StockC Parts_01_Inside = True FAC Parts_01_Wanted_WS0 = True Then StockC Liberate_Part(Part-Type-01) FAC Receive_Part (WS0)	Rule <i>Store_Part_10_FAC_WS1</i> If StockC Parts_10_Inside = True FAC Parts_10_Wanted_WS1 = True Then StockC Liberate_Part(Part-Type-10) FAC Receive_Part (WS1)	Rule <i>Store_Part_11_FAC_WS4</i> If StockC Parts_11_Inside = True FAC Parts_11_Wanted_WS4 = True Then StockC Liberate_Part(Part-Type-11) FAC Receive_Part (WS4)

**Figure 84: Rules for enabling base-part in the WSs of FAC.**

<sup>129</sup> There also exist complementary Rules for transferring base-parts to alternatives WS when they need parts and the main WS are with the appropriate number of base-parts.

In the carried out simulation experiment, for each product of assemblage, the respective number of base-parts was generated for reloading the WSs in the FAC. Each time that a base-part was reserved in the FAC-HL to an assemblage, a respective Base-Part-HL was started. Also, short bars were generated in a chained way to enable the production of base-parts.

The production chaining enabling just-in-time production was observable in the graphical module of ANALYTICE II. In the carried out experiment, it was possible simulating the assemblages of the product without demand previsions, avoiding large buffering.

In this simulation effort, the system considered has been effectively simplified. For a more refined design, involving production analysis, it would be necessary to develop a more detailed experiment. Nevertheless, the present experiment has served to its objective that is to exemplify the potential utilization of the meta-model for HDCS in a higher level than process-cell scope.

#### 7.4.5. Summary

In this case study, it was presented more an application of the meta-model for HDCS within ANALYTICE II, which preserves the Rule inference by means of Resource-HL notification, as well as the use of Rules by Smart-Product-HLs for reaching their production desires. In spite of ANALYTICE II drawbacks, this case study has additionally demonstrated a way to use ANALYTICE II in a more high-level way, by abstracting simulation details for reaching experiments in an easier manner.

Anyway, the main goal of the case study was reached by presenting an instance of the meta-model in higher level than process-cell scope. In this experiment, it is observed that the Rules are similar, Smart-Product-HLs are similar, and Resource-HLs are similar at interface level too. Furthermore, as expected, these entities in higher-level scope are (self) similar to those in the lower-level scope previously presented.

This experiment has allowed presenting a type of HSFC or HMES constituted by some Rules working in a process-driven way and other Rules working in a product-driven way. Additionally, the experiment has allowed presenting the coexistence of Resource-HLs comprising a resource (i.e. Stock-HL) and Resource-HLs comprising subholarchies (i.e. Process-Cell-HLs). In short, the case study exemplifies the complexity for composing a holarchy and simultaneously exemplifies the usefulness of homogenized elements to compose holarchies.

### 7.5. Considerations

A set of case studies was proposed, in this chapter, as additional works for sensitizing about the generality, reusability, and potentiality of the meta-model proposed for HDCS. The case studies were related to a same production context, each one considering a context aspect, in certain scope and level of the production execution. In the two first case studies the meta-model was understood as solution to HSC while in the last case study as a solution to HSFC or HMES.

The HSC and HSFC, due to the presented case studies, have been effectively understood as similar types of HDCS applied to different production levels. In addition, other models, using the previously proposed UML derivation, could be presented for better detailing the generated control instances and their similarities. Likewise, the control instances could be also expressed in terms of Petri Nets (PNs). However, the detailed and additional modeling would be so large to be expressed in the pages of the thesis.

In this work, it was presented that the control execution in an architecture instances is like a PN playing. Therefore, the PN model could be used to simulate, at high-level, those holonic control instances for observing some properties. The PN model could also serve as design step in the conception of each control instance over ANALYTICE II and even the results of both types of simulation could be compared.

The ANALYTICE II prototype has been used to test the control architecture. Concomitantly, ANALYTICE II has been improved by the application of the architecture to be an appropriate simulation tool for HMS issues. In fact, the drawbacks and advantages of this prototype have been clarified and related during the development of the experiments presented for testing the control architecture.

In spite of the ANALYTICE II advantages, it is necessary, for example, a set of improvements in its design features, aiming at an easier way to compose experiments such as those developed in this thesis. These improvements and other ones, related during the development of this thesis, could enable the large utilization of ANALYTICE II.

A differential feature of ANALYTICE II is the explicit separation between plant emulation and informational entities by means of the virtual network. This separation has been called a *realism feature*. Some considerations about this feature were made during the development of the experiments, namely:

- On the one hand, the virtual network is a certain difficulty to compose base holons that are the base elements on which the control instances work. This difficulty exists because the hard and soft-part holons must be synchronized, making harder the experiments to test the holonic control instances.
- On the other hand, the virtual network allows feeling the difficult degree to create base holons and run the control instances over them. Actually, some realism in the simulation environment allows taking a better feedback about HMS compositions.

In point of fact, the nature and actual evolution state of the ANALYTICE II, as well as the all complexity of a holarchy involving many decisional systems (as HDCS and HFM) make hard the development of a complete holarchy by means of one researcher. In this sense, a complete analysis of a holarchy agility would involve an implementation of many elements in ANALYTICE II, demanding a staff of development and/or many time of implementation.

The simulations carried out could be improved, by means of the appropriate human resources, with more types of Smart-Product-HLs, different scenarios, and son forth. The achieved agility by means of the solution use could be also observed in the ANALYTICE II graphical animation or data could be gotten and analyzed.

Anyway, the considered case studies enable some *artifacts* presenting the feasibility of the proposed meta-model for the construction of appropriate models and instances of agile controls. Also, detailed performance evaluations in the instances of holonic control are important to design of specific instances. However, the objective of this work is effectively to propose an open meta-model allowing the construction of agile HDCS.

Actually, suitable control system can be reached with appropriate project and knowledge of the controlled system in anyway. The differential is the quality of engineering-tools (e.g. meta-models) facilitating the construction of each suitable control instance. This thesis has presented a meta-model for the composition of HDCS, previously enabling a set of predefined and tested entities and comprising openness of the solution to correctness features and systemic integration. This features giving a certain guaranty and easiness for creating suitable control instances.



## **Conclusion and Perspectives**

---



# Chapter 8 : Conclusion and Perspectives

## 8.1. Conclusion

The initial objective within this doctoral thesis was the holonification of ANALYTICE II, a prototype for the design and simulation of Manufacturing Systems (MS), in order to obtain a tool for Holonic MS (HMS) issues. The holonification of ANALYTICE II was carried out comprising the development of a solution to compose base holons and a holonic control solution related to their cooperation. In fact, due to the ANALYTICE II nature, its holonification also comprised potential solutions to real production cases.

The structure of ANALYTICE II, including a virtual network, generates certain *realism* in its holonification. The virtual network, even if simplified in detail, simulates a network between plant and high-level control entities. In ANALYTICE II simulations, the environment is composed of heterogeneous and separate elements, in many aspects similar to the real production environments comprising *automatics* and *computational* subsystems. Actually, this feature allows feeling the complexity to compose integrated MS, such as HMS.

In order to overcome the complexity, caused by the heterogeneity in the environment, the holon (HL) concept was interpreted as the integration of each heterogeneous entity in the computational (or virtual) scope via an *expertise layer*. In ANALYTICE II, virtual entities have integrated respective emulated entities on the control-side, composing the Base-HLs (i.e. Resource-HLs and Smart-Product-HLs). Metaphorically, each virtual entity encapsulates an emulated entity because the internal specificities of Base-HLs are hidden at virtual level.

Any Resource-HL, for example, hides synchronizations between its virtual resource and emulated resource. In truth, Resource-HLs (e.g. Robot-HLs, Machine-HLs, and Process-Cell-HLs) present accurate data and receive services requests at virtual level for collaborations with other holons. Each Resource-HL also has its *enabled data* and *services* respectively patterned in Attribute and Method subholons, at virtual level, aiming to facilitate collaborations with the holarchy entities and reach more modularity in its structure.

The use of Attributes and Methods let Resource-HLs as (self) similar ones, at interface level, allowing standardized collaborations and reaching generality in their external model. In this sense, the homogenized Resource-HLs have facilitated the proposition of a solution to holonic control because its interactions with resources are at high-level and always in the same manner. The holonic control knows monitored data by means of Attributes, *calculates* the suitable actions by using control knowledge, and requests services by means of Methods.

In fact, this environment of ANALYTICE II, with homogenized Resource-HLs, enabled a technical platform to demonstrate the feasibility of a special holonic control solution that has been the main objective of the thesis. Effectively, a holonic control architecture was proposed having essential features tested within ANALYTICE II. Additionally, this architecture has concomitantly served to ANALYTICE II holonification once simulated holarchies can be created by the instantiation of holonic controls over simulated Resource-HLs.

The special solution to carry out control over Resource-HLs, such as those tested using ANALYTICE II, was proposed comprising the openness of the solution to a set of suitable features for holonic control, such as correctness, agility, and integration features. Actually, it was searched a proper control architecture for Holonic Discrete Control Systems (HDCCS) as a

contribution to the present efforts related to HMS and to the correlated efforts about execution and composition of *modern control system* such as SC and SFC/MES.

Summarily, it was developed a particular HDCS architecture, applicable to the cooperation of proposed Resource-HLs, which is considered feasible with respect to real systems due to some factors such as the nature of the testing environment. In fact, the testing environment comprising the implementation of the Base-HL solution and a holonic control solution constitutes a prototype to HMS simulation tool.

The prototype and the control solution by itself are contributions in agreement with the needs in holonic control. The proposed prototype enables a tool for issues of HMS design and test, which is considered promising due to features such as its economical and technical feasibility, its certain realism and scalability, and the integrated holonic control solution. Actually, the control solution is an innovative and self-contribution due to its set of particularities.

A special feature of the proposed control solution is its own nature. It consists of a generic control architecture that allows composing self-contained instances, comprising control knowledge for exploring flexibilities of the MS, aiming at agility. Moreover, the architecture is open enough to allow other decisional systems systemically interacting with instantiated controls, e.g. for improving the flexibility utilization. The architecture also has a set of other particularities, such as a *trade-off* between generality and applicability.

The generic architecture presents its modeling entities and their instantiations, as well as presents its boundaries and connection possibility with other HMS elements. Also, the generic architecture contains a set of features related to holonic and correct control. Examples are the particular inference process and the related openness to extra control mechanisms and distributed control. Furthermore, the generic architecture has two different types of orientations to holonic control, i.e. process or product-driven control.

In the process-driven case, the control solution enables an alternative approach within the HMS scope. This approach allows reaching certain agility by exploring *only* process information using an agent organization as technological mean. In the product-driven case, the control solution enables an alternative approach for the product-driven control because, beyond the Smart-Product-HL benefits, there exist the organization of Smart-Product-HL collaborations with Resource-HLs via control entities previously defined in the process-driven approach.

Essentially, the two defined orientations to compose holonic controls are complementary ones. In fact, each approach presents certain drawbacks. A drawback example in the process-driven case is some fixing about the buffering places for product-types while in the product-driven case is the complexity to effectively reach Smart-Product-HLs. Anyway, this double orientations property for holonic control architecture is desirable because the approaches can be used based on the benefits and drawbacks to each foreseen control instance.

The control solution proposed in this thesis was firstly thought in a process-driven way. The proposition of the process-driven architecture has been inspired by concepts and techniques found in the Artificial Intelligence (AI) and Software Engineering (SE) domains. Particularly, the Rule Based Systems (RBS), an AI technique, was a highlighted inspiration source. It was considered the use of causal rules, as in RBS, for the causal reasoning (e.g. in the control of Resource-HL cooperation) by means of an agent or object oriented (OO) *point of view*.

Within the AI domain, RBS has been considered cognitively appropriate to express causal relations, such as those that guide the control execution. The causal relations are expressed by means of *rules* in RBS that are a natural manner for human thinking. In a concordant manner, in this thesis, the rules have been considered cognitively proper to express the causal relations

of control. However, the present RBS technology has not been considered quite suitable to the execution of the holonic control where a set of rules determine the decision-making.

Classically, RBS execution uses a centralized inference engine, based on search methods, to infer about the rule applications with respect to a fact base. This contrasts with the potential distribution of holonic control. Furthermore, the principle of non-active entities depending of a unique inference element is incompatible with the principles of holonic systems, which are ideally composed of decoupled and proactive elements. In this thesis, it was decided to evolve RBS principles with respect to its own composition and to the inference process execution.

In fact, the RBS evolution aimed at a more suitable composition and inference process to holonic control. In the enhanced RBS, each abstract entity of an ordinary RBS (as rules and their conditions and actions) was *personified* as an actual computational entity, namely objects of the OO approach. These entities are also considered software agents or soft-holons in the holonic thinking. For instance, a rule is a soft-holon called Rule with Condition and Action subholons.

In the proposed RBS-like, the inference process was also changed. Actually, it was improved by the notification concept within a proper agent organization. In the proposed solution, the agents cooperate via notifications of facts to carry out the inference process. There not exist a monolithic inference engine using search methods. The notification about new facts comes from an active base of facts, i.e. the Resource-HL Attributes, reaching Rules that are deliberative soft-holons about the control moments.

The Rules, after notifications, deliberate and conclude about the appropriate moments to control actions (e.g. Resource-HL coordination) that suitably instigate Resource-HL Methods, implying in fact changes. This deliberation and coordination occurs, in fact, via a notification chain allowed by the Rule composition. The Rule *personification* reaches their structures with Premises linked to Conditions and Instigations linked to Actions. Attributes notify Rule Conditions via Premises potentially allowing Rule Actions activate Methods via Instigations.

Actually, the notification principle within this agent organization for RBS is an improvement to the execution of inference process, allowing direct propagation and sharing of inferred information. Moreover and mainly, this notification and agent-oriented RBS solution brings special advantages to control applications. In this doctoral thesis, it was presented actual and potential benefits of this solution to control. It has been presented, for example, the openness of this control solution based on notifications to correctness and holonic properties.

In the cooperation established via notification, entities notify only other concerned entities due to a policy defined in the Rule instantiations, from which the notification chain emerges. Summarily, this policy brings advantages because it implicitly optimizes the interaction number in the control system and creates certain independence from notification receivers. These properties facilitate the proposition of additional mechanisms for handling some control issues, such as conflict and determinism issues, both in distributed and non-distributed cases.

In reality, in this rule and agent oriented control, the defined notification policy generates the decoupling of the control entities, allowing from distributed to non-distributed implementations with optimized communication and very low computational complexity. For that reason, the solution is appropriate to a great amount of knowledge in its implementations, e.g. aiming at agile control. Briefly, the inference process based on notification is suitable in a RBS for control and enables its utilization for holonic control issues.

Moreover, the nature of the similar Rules respects a *trade-off* between generality and applicability, as well as the issue about proper holonic control execution and easy expression mainly if a suitable environment to rule definition is enabled. Actually, the expert that

composes a holonic control system is exclusively concerned with the knowledge that guides the control execution. The expert is not concerned with the creation of the control mechanisms that are coherently emerged, in background, based on the expressed control knowledge.

It is expected that the knowledge from experts to the Rules, about the process, permit the rule-oriented control to properly act for exploring system flexibilities. The experiment, models, and properties of the proposed architecture indicated this way. However, the process of the knowledge creation is not a central issue in this thesis, even if it had been indicated that the creation of coordination Rules should be based on well-elaborated product process plans, therefore potentially reaching a set of Rules that foresees the possible routing flexibility.

The control using causal rules is cognitively suitable to human experts, in both control creation and understanding. However, rules can be seen as a *soft* formalism to express the control content. Opportunely, the proposed solution has been considered as open to Petri net (PN) formalisms due to existent mapping of PN to RBS. Additionally, the execution of the proposed inference process is like the actual PN playing. Therefore, the architecture instances can be designed by means of PNs, which are widely used in Discrete Event Control (DEC).

The openness of the solution to PN formalism brings some advantages. The PN can be used as a design step for the control synthesis, whereby properties can be evaluated by the PN playing and even PN analysis, depending upon its class. Moreover, it is expected that the *proximity* of the control solution and PN natures tends to a control implementation reflecting the properties observed in its design. Also, this openness of the solution to PN formalism can complementarily open this solution of control execution to related synthesis methods.

The proposed control architecture presents certain openness to some control issues. It has been presented, for example, the openness of the solution to synthesis formalism, additional control mechanism, and systemic integration for agility concerns. An instance of systemic integration has been holons for production-flow management acting on the Rule priorities or validities for reaching some production adaptability. Indeed, this type of openness within the proposed process-driven control solution allows its adaptation to a product-driven control one.

Basically, in the product-driven case, coordination Rules serve to Smart-Product-HLs reach production steps by means of the concerned Resource-HLs. As a particularity, the Rules have its execution normally forbidden in this context. Each Rule ready to execute is effectively allowed to execute only when it can serve a Smart-Product-HL in a specific moment. It is the concerned Smart-Product-HL that notifies the Rule about its utility. Each Smart-Product-HL is informed about suitable Rules to each production step, in its creation, based on its type.

The adaptation of the proposed control solution for a product-driven control solution presents a beneficial double implication. The Rules organize and regulate the use of Resource-HLs by means of Smart-Product-HLs while Smart-Product-HLs potentially bring the benefits of the smart-product concept, such as additional control information even to the Rules. In general, Smart-Product-HLs also allow certain decoupling between the production launching and execution by means of the suitable utilization of system flexibility.

In short, in this doctoral thesis, it was proposed a control solution being an open and generic architecture whose entity models foresee (self) similar holons as much as possible. The (self) similarity of holons in the solution contributes to its certain instantiation easiness. In fact, the solution is a meta-model with potentiality for the concomitant and synergic treatment of a set of control issues. The control meta-model is a tool of system engineering whose essence is based on concepts of the computer science.

The meta-model was implemented as a framework over ANALYTICE II, which has been an engineering tool to aid in the composition of experiments. The proposed experiments allowed demonstrating the meta-model applied as Holonic Supervisory Control (HSC) and as Holonic Manufacturing Execution System (HMES) in the simulation scope. Effectively, this HDCS meta-model is an effort in integrative tool to add computational smartness necessary to agile production.

The proposed meta-model is an engineering tool to collaborate in the handling of the complexity found in HMS. Actually, it represents a viable research path to be followed due to its properties and potentialities presented within this thesis' development. The meta-model is an alternative solution oriented to AI and SE aspects potentially applicable to real case because its main features were tested in a proper simulator, i.e. ANALYTICE II.

The contributions of these thesis' efforts are engineering tools consisting of a particular prototype and control meta-model achieved in the context of the ANALYTICE II holonification. These tools are contributions to HMS issues considered by the research community, both in testing tools and control solutions. Specifically, these tools are also contributions for the CPGEI/CEFET-PR and CRAN-UHP research efforts to the development of engineering tools in the scope of modern manufacturing systems.

In fact, the prototype and control meta-model are involved in a proper approach to the use and development of engineering tools. Equally, they are involved in a research approach to investigate alternative, innovative, and/or better engineering solutions. This research approach actually allows presenting subjects to be followed by the industry branches in technological research and development.

## **8.2. Perspectives**

The engineering tools proposed in this doctoral thesis are related to a multidisciplinary context whose elaborations were established upon diversified viewpoints. The proposed solution clearly presents a diversity of branches to be investigated with respect to its scope. These branches are related to the improvements and applications directly concerned to ANALYTICE II and the control meta-model in its scope, as well as to the independent improvement and applicability of the control meta-model.

The application of the meta-model to real cases is an example of future works, which would allow a deep investigation of the solution feasibility and advantages. The institutions to which this thesis is associated have access to pedagogic production-cells, whereby the holonic-control meta-model could be applied and evaluated. These holonified systems should also serve to attract industrial investments. Moreover, these systems could be designed and simulated in ANALYTICE II, with certain refinement, aiming at comparisons and marketing.

Another set of future works should be the ANALYTICE II improvements, such as those described in the presentation of the case studies. In short, the improvements would be to facilitate the design, simulation, and evaluation of MS and HMS. Implicitly, this concerns to the development and integration of a set of modules to make easy the composition and comparison of control instances.

In fact, a relevant work could be to define and make comparisons between control systems implemented in different approaches (as hierarchic, heterarchic, and holonic) over the same MS simulated in ANALYTICE II. The comparison could even be between different control architectures or firstly between different instances of the proposed meta-model once different control types could be generate with appropriate knowledge in the Rules and Resource-HLs.

Effectively, a set of modules (or *wizard*) to control issues is essential to the activities foreseen to ANALYTICE II. With respect to the proposed solution, this *wizard* could integrate, for example, modules for the Rule or PN composition, Rule distribution, and Rule manipulation. Also, some modules in the *wizard* could additionally include methods for translating from a type of control formalism into another and methods for previously making the possible analysis of the control instances, at model level.

The *wizard* should also have certain independence from ANALYTICE II as much as the control solution is independent of the simulator. In this *wizard*, the control should be created without concerns about the base holons nature, i.e. real or emulated ones, once the meta-model presents this functional independence.

Moreover, the *wizard* could have a library of interfaces to hardware controllers aiming to facilitate the composition of real Resource-HLs. The *wizard* could even have another module for technologies related to the composition of Smart-Product-HLs. However, the composition of real Base-HLs is by itself another set of futures works even taking into account the efforts and results presented in the literature.

The foreseen *wizard* and an improved ANALYTICE II would constitute a robust tool for composing control system as HSC and HMES. In fact, an instance of the control meta-model in real application, developed using this tool, would be an additional integrative software layer. It would be an additional integrative software layer between specific controllers encapsulated in Base-HLs and high-level decisional information system, allowing the management and control of the production in a more agile way.

The integration of the proposed solution with present solutions to other decisional systems is also a subject to be explored. Also, some studies could be made about the fitting of the proposed solution and even integrated ones to the present normalization. Furthermore, the solutions could be related to more generic architectures for system integration that have been proposed in the scope of CIM and also in the scope of HMS.

The proposition of other decisional systems nearer of the holonic thinking and integrated to the proposed meta-model is another study subject. In the scope of these holonic decisional systems, it could be defined automatic manners to change Rules priorities and validities for optimizing or adapting the production. Moreover, studies about the automatic changing and even creation of Rules could be carried out.

Besides the amount of technical and systemic works around the proposed meta-model, other pertinent work would be its better formalization. The control meta-model has had some formalization by the use of the derivation of the UML's meta-model in its proposition. Some formalism has also been reached applying the Fusaoka's equation and some formalism has been searched in the *approximation* to PN. Nevertheless, the meta-model could be strongly formalized, for instance by means of formal languages, as B or Z languages.

Finally, another interesting works would be to investigate about the potential contributions that the meta-model could provide to other system engineering areas, such as intelligence artificial and discrete control in general. Due to (self) similarity between entities and the PN compatibility, the application of the architecture to DEC in a more general way is evident. Moreover, the RBS structure has been improved, being this present research a certain innovation in this subject whose consequences could be studied.



## **Bibliography**

---



# Bibliography

- (Bajic et Chaxel, 1997) Bajic E. and Chaxel F.: *Towards a holon-product oriented management*. 4th IFAC Workshop on Intelligent Manufacturing Systems, IMS'97. Seoul (Korea) 1997.
- (Bako et al., 1990) Bako B., Valette R., and Courvoisier M.: *A controlled rule-based system interpreter: an application to FMS simulation*. AI, Simulation and Planning in High Autonomy Systems. Tucson (USA) 1990.
- (Bako et Valette, 1990) Bako V. and Valette R.: *Towards a decentralization of rule-based systems controlled by Petri Nets: an application to FMS*. Fourth International Symposium on Knowledge Engineering. Barcelona (Spain) 1990.
- (Banks, 1996) Banks J.: *Software for Simulation*. Proceedings of the 1996 Winter Simulation Conference (ed.: J. M. Charnes, D. J. Morrice, D. T. Brunner, J. J. Swain). Coronado (USA) 1996.
- (Bongaerts, 1998) Bongaerts L.: *Integration Of Scheduling And Control In Holonic Manufacturing Systems*. Ph. D. Thesis. PMA / Katholieke Universiteit Leuven. Leuven (Belgium) 1998.
- (Brennan, 2000) Brennan R. W.: *Performance comparison and analysis of reactive and planning-based control architectures for manufacturing*. Robotic and Computer Integrated Manufacturing, Vol. 16 N. 2-3 (pp. 191-200) Pergamon, 2000.
- (Brennan et Norrie, 2001) Brennan R.W. and Norrie D. H.: *Evaluating the performance of reactive control architectures for manufacturing production control*. Department of Mechanical and Manufacturing Engineering, University of Calgary, 2500 University Drive NW, Calgary, Alta., Canada T2N 1N4. Received 24 march; accepted 13 march 2001. In: (Valckenaers, 2001)
- (Breugnot et al., 1990) Breugnot D, Gourgand M., and Kellert P.: *SIGMA: An Intelligent and Graphical Environment for the Modelling of Flexible Assembly-Systems*. Proceedings of the European Simulation Symposium. Ghent (Belgium) 1990.
- (Breugnot et al., 1991) Breugnot D., Gourgand M., Hill D., and Kellert P.: *GAME: An Object-Oriented Approach to Computer Animation in Flexible Manufacturing System Modelling*. Proceedings of the 24th Annual Simulation Symposium (pp 217-227) - joint publication - ACM SIG SIM The Special Interest Group on Simulation v. 21 n. 3, IEEE CS TCSIM The Simulation Technical Committee v. 35. April 1991.
- (Brugali et al., 1998) Brugali D., Menga G., and Aarsten A.: *The Framework Life Span: A Case Study for Flexible Manufacturing Systems*. In Object-Oriented Application Frameworks. Wiley, 1998.
- (Cardoso et Valette, 1997) Cardoso J. and Valette R.: *Redes de Petri*. Editora da UFSC. Florianópolis (Brazil) 1997.
- (Carvalho, 2003) Carvalho L.G.: *Concepção de um Módulo de Monitoração para ANALYTICE II*. Master in Science Thesis, CPGEI-CEFET\_PR. Curitiba (Brazil) 2003.
- (Cassandras et Lafortune, 1999) Cassandras C.G. and Lafortune S.: *Introduction to discrete event systems*. Kluwer Academic Publishers, 1999. ISBN 0-7923-8609-4.
- (Chaar et al., 1993) Chaar J.K., Teichroew D., and Volz R.A.: *Developing Manufacturing Control Software: A Survey and Critique*. The International Journal of Flexible Manufacturing Systems. Vol. 5. N. 1 (pp. 53-88). Kluwer Academic Publishers. Netherlands, 1993.

- (Chao et Wang, 1995) Chao D. Y. and Wang D. T.: *XPN-FMS: A CAD Tool for FMS Modelling, Analysis, Animation, and Simulation Using Petri Nets and X Window*. The International Journal of Flexible Manufacturing Systems vol. 7 (pp 339-360), Kluwer Academic Publishers, 1995.
- (Cheng et al., 2004) Cheng F., Chang C., and Wu S.: *Development of Holonic Manufacturing Execution Systems*. International Journal of Intelligent Manufacturing, 15, (2) (pp. 253-267), Kluwer Academic Publishers. Manufactured in The Netherlands, 2004.
- (Chézaviel et al, 1999) Chézaviel B.P., Künzle LA., Girault F., and Valette R.: *Calculation duration of concurrent scenarios in time Petri nets*. Rairo - Journal Européen des Systèmes Automatisés (JESA), v.33, n.8-9 (pp.943 – 958). Paris (France) 1999.
- (Coulouris et al., 2001) Coulouris G., Dollimore J., and Kindberg T.: *Distributed Systems – Concepts and Designs*. Pearson – Addison Wesley, 2001. ISBN 0-201-61918-0.
- (Cury, 2000) Cury, J. E. R.: *Modular Supervisory Control of Large Scale Discrete Event Systems*. In: Discrete Event Systems - Analysis and Control.1 ed. Boston: Kluwer Academic Publishers (pp. 103-110), 2000.
- (Cury et al., 2004) Cury J.E.R., Torrico C.R.C., and Cunha A.E.C.: *Supervisory Control of Discrete Event Systems with Flexible Marking*. European Journal Of Control, v.10, n.1 (pp.47 – 60), 2004
- (Cury et Queiroz, 2000) Cury J. E. R. and Queiroz M. H. de: *Controle Modular De Sistemas De Manufatura Discretos*. LCMi. Anais do XIII Congresso Brasileiro de Automática. Florianópolis (Brazil) 2000.
- (Cury et Queiroz, 2001) Cury J. E. R., Queiroz, M. H. de, and Santos, E. A. P.: *Síntese Modular do Controle Supervisório em Diagrama Escada para uma Célula de Manufatura*. V Simpósio Brasileiro de Automação Inteligente. Canela (Brazil) 2001.
- (Da Silveira et al., 2001) Da Silveira G., Borenstein D., and Fogliatto F.S.: *Mass customization: literature review and research directions*. International Journal of Production Economics, 72 (pp 1-13), 2001.
- (Davis et Williams, 1992) Davis L. and Williams G.: *Evaluating and Selecting Simulation Software using The Analytic Hierarchy Process*. Integrated Manufacturing Systems, vol. 5, n.1, MCB University Press Limited, 1992.
- (Deen, 2003) Deen S.M.: *Agent-Based Manufacturing: Advances in the Holonic Approach*, 2003. Springer. ISBN 3-540-44069-0.
- (Eteessami et Hura, 1991) Eteessami F. S. and Hura G. S.: *Knowledge Net Shell (Kns): Petri Net Based Development Tool for Expert System*. Microelectron. Reliab. Vol31, N. 4 (pp. 793-812), 1991.
- (Eyzell et Cury, 2001) Eyzell J.M. and Cury J.E.R: *Exploiting Symmetry in the Synthesis of Supervisors for Discrete Event Systems*. IEEE Transactions on Automatic Control, v.46, n.9, p.1500 - 1505, EUA, 2001.
- (Fletcher et al., 2003) Fletcher M., Brennan R. W. and Norrie D. H.: *Modeling and reconfiguring intelligent holonic manufacturing systems with Internet-based mobile agents*. Journal of Intelligent Manufacturing, 2003. Kluwer Academic Publishers in The Netherlands.
- (Forgy, 1982) Forgy C. L.: *RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem*. Artificial Intelligence, 1982.
- (Franklin et Graesser, 1996) Franklin S. and Graesser A.: *Is It an Agent, or Just a Program? A Taxonomy for Autonomous Agents*, Institute for Intelligent Systems. Proceedings of the Third

- International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag, 1996.
- (Frey et al., 2003) Frey D., Nimis J., Wörn H., and Lockemann P.: *Benchmarking and Robust Multi-agent-based Production Planning and Control*. In: (Morel et Grabot, 2003).
- (Fusaoka et al., 1983) Fusaoka A., Seki H. and Takahashi K.: *A description and reasoning of plant controllers in temporal logic*. Proceedings of the 8<sup>th</sup> International Joint Conference on Artificial Intelligence. Karlsruhe (Germany) 1983.
- (García et al., 2003) García A., McFarlane D., Thorne A. and Fletcher M.: *The Impact Of Auto-Id Technology In Material Handling Systems*. 7<sup>th</sup> IFAC Workshop on Intelligent Manufacturing Systems, IMS 2003 PREPRINTS (pp. 227 – 232). Budapest (Hungary) 2003.
- (Germain et al., 2003) Germain B. S., Valckenaers P., Brussel H. V., Hadeli, Bochmann O., Zamfirescu, C. and Verstraete P.: *Multi-Agent Manufacturing Control/ an Industrial Case Study*. 7<sup>th</sup> IFAC Workshop on Intelligent Manufacturing Systems, IMS 2003 PREPRINTS (pp. 227 – 232). Budapest (Hungary) 2003.
- (Gouyon, 2004) Gouyon D. *Contrôle par le Produit des Systèmes d'Éxecution de la Production: Apport des Techniques de Synthèse*. Ph.D. Thesis. Centre de Recherche en Automatique de Nancy - Université Henri Poincaré, 2004.
- (Gouyon et al., 2004) Gouyon D., Simão J. M., Khaled A, and Morel G.: *Product-Driven Issues for B2M-Control Systems Integration*. INCOM 2004 – 11th IFAC Symposium on Information Control Problems in Manufacturing. Salvador (Brazil) 2004.
- (Graham, 1988) Graham G.A.: *Encyclopedia of Industrial Automation*. Ed. Longman Scientific & Technical, 1988.
- (Gullander et al., 1998) Gullander P, Bongaerts L., and Wyns J.: *Comparison of Reference Architectures: Chalmers Modified Hierarchical Architecture vs. K.U.Leuven Holonic Architecture*. Proc. of European Conference on Integration in Manufacturing, Goteborg (pp. 710-719), 1998.
- (Harhalakis et al., 1995) Harhalakis G., Lin C. P., Mark L., and Muro-Medrano P. R.: *Structured Representation of Rule-Based Specifications*. In CIM Using Updated Petri Nets. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 25, No.1, January 1995.
- (Hartley, 1984) Hartley, J.: *FMS at work*. IFS (Publications) Ltd., UK, 1984. ISBN 0-903608-62-6.
- (He et al., 1999) He X., Chu W. C., Yang H., and Yang S. J. H.: *A New Approach To Verify Rule-Based Systems Using Petri Nets*. Twenty-Third Annual International Computer Software and Applications Conference (IEEE). Phoenix -Arizona, 1999.
- (Heikkila et al., 2001) Heikkila T., Kollingbaum M., Valckenaers P., and Bluemink G.: *An Agent Architecture for Manufacturing Control: manAge*. In: (Valckenaers, 2001) pp. 315-331.
- (Hollocks et al., 1997) Hollocks B.W., Gorason H.T., Shorter F.N. and Vernadat F.B.: *Assessing Enterprise Integration for Competitive Advantage*. Workshop 2, Working Group 1. Enterprise Engineering and Integration: Building International Consensus (K. Kosanke and J. G. Nell eds) Springer-Verlag (pp. 96-107). Berlin (Germany) 1997.
- (IEC 61499) International Electrotechnical Commission: *Functions Blocks, Part1 - Architecture*. IEC PAS 61499-1, Geneva, Switzerland, 2000.
- (ISA 95 - Part 1) ISA-dS95.01-1999 Draft Standard. Enterprise – Control System Integration Part 1: Models and Terminology. Draft 14, November 1999.

- (Iung et al., 2001) Iung B., Neunreuther E., Morel B.: *Engineering Process of Integrated-Distributed Shop Floor Architecture Based on Interoperable Field Components*. International Journal of Computer Integrated Manufacturing, 2001 Taylor & Francis Ltd. ISSN 0951-192X.
- (Jackson, 1990) Jackson P.: *Introduction to Expert Systems*. Addison-Wesley, 1990.
- (Jeng, 1995) Jeng M. D.: *Modular Synthesis of Petri Nets for Modelling Flexible Manufacturing Systems*. The International Journal of Flexible Manufacturing Systems, 7 (pp 287-310), 1995.
- (Kärkkäinen et al., 2003) Kärkkäinen M., Holmström J., Främling K., and Artto K.: *Intelligent products – a step towards a more effective project delivery chain*. Computers in Industry 50, (pp. 141-151) 2003. Elsevier.
- (Koscianski, 2000) Koscianski A.: *Projeto e Implementação de um Simulador com Animação Gráfica para FMS*. Master in Science Thesis. CPGEI-CEFET-PR. Curitiba (Brazil) 2000.
- (Koscianski et al., 1999) Koscianski A., Rosinha L. F., Stadzisz P. C., and Künzle L. A.: *FMS Design and Analysis: Developing a Simulation Environment*. In Proceedings of the 15th International Conference on CAD/CAM, Robotics and Factories of the Future, vol.2. (p.25 - 210). Águas de Lindóia (Brazil) 1999.
- (Kotak et al., 2003) Kotak D., Wu S., Fleetwood M., and Tamoto H.: *Agent-base holonic design and operations environment for distributed manufacturing*. Computers in Industry 52 (pp. 95-108), Elsevier, 2003.
- (Künzle, 1990) Künzle L. A.: *Controle de Sistemas Flexíveis de Manufatura – Especificação dos Níveis Equipamento e Estação de Trabalho*. Master in Science Thesis, CPGII-CEFET-PR. Curitiba (Brazil) 1990.
- (Künzle et al., 1999) Künzle L.A., Valette R., and Chézalviel B.P.: *Temporal reasoning in fuzzy time Petri nets*. Fuzziness in Petri Nets, vol.22 (pp. 146-173) Physical Verlag, 1999.
- (Langer et al., 2000) Langer G., Sorensen C., Schnell J., and Alting L.: *Design of a Holonic Shop Floor Control System for a Steel Plate Milling-Cell*. 2000 International CIRP Design Seminar on Design with Manufacturing: Intelligent Design Concepts Methods and Algorithms. Haifa (Israel) 2000.
- (Larsen et al., 2001) Larsen M. H., Sorensen C., and Langer G.: *Development of a Production Meta Product State Model*, 2001. In (Valckenaers, 2001).
- (Lavigne et al., 2003) Lavigne J., Mayer F., and Lhoste P.: *Category Theory Based Approach for IMS Modelling*, IMS 2003 Budapest.
- (Law, 1986) Law A.M.: *Introduction to Simulation: A Powerful Tool for Analysing Complex Manufacturing System*. Industrial Engineering, 1986.
- (Law et Kelton, 1991) Law A.V. and Kelton W.D.: *Simulation Modelling & Analysis*. McGraw-Hill, Inc., USA, 1991.
- (Lima et Dórea, 2002) Lima E.A. and Dórea C.E.T.: *An Algorithm for Supervisory Control of Discrete-Event Systems via Place Invariants*. Proceedings of the 15th IFAC World Congress. Barcelona (Spain) 2002.
- (Lima II et Dórea, 2004) Lima II E.J. and Dórea C.E.T.: *Synthesis and PLC Implementation of Supervisory Control via Place Invariants for a Manufacturing Cell*. In: 11th IFAC Symposium on Information Control Problems in Manufacturing. Salvador (Brazil) 2004.

- (Lira et al., 2000) Lira A.F., Santos A.M., and Stadzisz P.C.: *An Assembly Planning Method for Families of Products*. 2000 International CIRP Design Seminar. Haifa (Israel) 2000.
- (Lhoste et Pétin, 2001) Lhoste P. and Pétin J.: *Réactivité et Déterminisme ds Modèles de Systèmes à Événements Discrets. Pedagogical Material*. Centre de Recherche en Automatique de Nancy. UPRES-A 7039, CNRS – Université Henri Poincaré. Campus Scientifique, BP 239, F-54506 Vandœuvre-lès-Nancy Cedex.
- (Lüders, 2001) Lüders R.: *Controle Multivariável de Sistemas a Eventos Discretos em Dióides*. Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Departamento de Engenharia de Computação e Automação Industrial. Doctor Thesis, 2001.
- (Macchi et al., 2003) Machi M., Cavalieri M., Garetti M.: *The complexity of design and test of multi-agent control systems: lessons learned in a decade of research experience*. IFAC 2003.
- (Mařík, 2004) Mařík, V.: *Industrial Application of the Agent-based Technology*. Copyright 2004 IFAC.
- (McFarlane, 2003) McFarlane D.: *Product Identity and Its Impact on Discrete Event Observability*. European Control Conference 2003, Cambridge, UK, 2003.
- (McFarlane et al., 2002) McFarlane D., Sarma S., Chirn J. L., Wong C. Y., and Ashton K.: *The Intelligent Product in Manufacturing Control and Management*. IFAC 2002, 15th Triennial World Congress, Barcelona, Spain, 2002.
- (McFarlane et al. 2003) McFarlane D., Sarma S., Chirn J. L., and Ashton K. *Auto ID Systems and Intelligent Manufacturing Control*. In (Morel et Grabot, 2003), pp. 365-376.
- (McFarlane et Bussmann, 2003) McFarlane D. C. and Bussmann S.. *Holonic Manufacturing Control: Rationales, Developments and Open Issues*, 2003 In: (Deen, 2003), pp. 303-326.
- (Mendes 1995) Mendes R. S.: *Modelagem e Controle de Sistemas A Eventos Discretos*. Dca/Fee/Unicamp, Campinas, SP. In: *Manufatura Integrada por Computador: Contexto, Tendências, Técnicas*. Fundação CEFETMINAS – Minas Gerais, pp. 159 – 180, 1995.
- (MESA, 2000) Manufacturing Execution Systems Association (MESA) International. White Papers, 2000.
- (Minski, 1985) Minski M.: *The Society of Mind*, Heinemann, 1985. Cited in (Wyns, 1999).
- (Miyagi, 1996) Miyagi P.E.: *Controle Programável – Fundamentos do Controle de Sistemas a Eventos Discretos*. Editora Edgard Blücher Ltda, 1996. ISBN 85-212-0079-X.
- (Morel et al., 2003) Morel G., Panetto H., Zaremba M. B., and Mayer F.: *Manufacturing Enterprise Control and Management System Engineering: paradigms and open issues*. IFAC Annual Reviews in Control. 27/2, 199-209, Elsevier, 2003, ISSN:1367-5788.
- (Morel et Grabot, 2003) Morel G., and Grabot B., (Eds.): *Intelligent Manufacturing*. Special issue of Engineering Applications of Artificial Intelligence, vol. 16 (4), 2003.
- (Muhl, 2002) Muhl E.: *Contribution a la Vision Globale de l'Ordonnancement du Flux Vehicule*. Ph.D. Thesis, Centre de Recherche en Automatique de Nancy - Université Henri Poincaré, 2002.
- (Muhl et al., 2003) Muhl E., Charpentier P., Chaxel F.: *Optimization of Physical Flows in an Automotive Manufacturing Plant: some experiments and issues*. In: (Morel et Grabot, 2003), pp. 293-305.

- (Müller, 1998) Müller J. P.: *Architectures and Applications of Intelligent Agents: A Survey*. International House. Ealing London W5 5DB, Knowledge Engineering Review, 1998.
- (Nazareth, 1993) Nazareth D. L.: *Investigating the Applicability of Petri Nets for Rule-Based System Verification*. IEEE Transactions on Knowledge and Data Engineering, vol. 4, no. 3. June 1993.
- (Neunreuther, 1998) Neunreuther E.: *Contribution à la modélisation des Systèmes Intégrés de Production à Intelligence Distribuée: application à la distribution du contrôle-commande et de la gestion technique sur les équipements de terrain*. Ph.D. Thesis, Centre de Recherche en Automatique de Nancy - Université Henri Poincaré, 2002.
- (Ollero et al., 2003) Ollero A., Morel G., Bernus P., Nof S.Y., Sasiadek J., Boverie S., Erbe H., and Goodall R.: *Milestone report of the Manufacturing and Instrumentation coordinating Committee: From MEMS to Enterprise Systems*. IFAC Annual Reviews in Control, 26, 151-162, 2003. ISSN 1367-5788.
- (Pan et al., 1998) Pan J., DeSouza G. N., and Kak A. C.: *FuzzyShell: A Large-Scale Expert System Shell Using Fuzzy Logic for Uncertainty Reasoning*. IEEE Transactions on Fuzzy Systems, Vol. 6, No 4., November 1998.
- (Patrioti, 1998) Patrioti, V.: *Systèmes de Pilotage Auto-Organisés et Gammes Distribuées: Méthode de Conception et Application à Une Machine-Outil*. Ph.D. Thesis. Centre de Recherche en Automatique de Nancy - Université Henri Poincaré, 1998.
- (Prado et al., 1998) Prado J. P. de A., Abe J. M., and Ávila B. C.: *Inteligência Artificial Distribuída; Aspectos. Série Lógica e Teoria da Ciência*, Instituto de Estudos Avançados – Universidade de São Paulo. São Paulo (Brazil) 1998.
- (Qiu et al., 2003) Qiu R., Wysk R., and Xu Q.: *Extended structured adaptive supervisory control model of shop floor controls for an e-manufacturing system*. International Journal of Production Research, vol. 41 (8) (pp. 1605-1620), 2003.
- (Ramadge et Wonham, 1987) Ramadge P. J. and Wonham W.M.: *Supervisory control of a class of discrete event processes*. SIAM Journal of Control and Optimization, vol. 25, n°1, 1987.
- (Rumbaugh et al., 1999) Rumbaugh, J., Jacobson, I. and Booch, G.: *The Unified Modeling Language Reference Manual*. Ed. Addison Wesley Longman, 1999.
- (Rich et Knight, 1991) Rich E. and Knight K.: *Artificial Intelligence*. McGraw-Hill, 1991.
- (Rosinha, 2000) Rosinha L. F. F.: *Proposta de uma Arquitetura de Simulação para Sistemas Flexíveis de Manufatura e de Modelagem de Equipamentos Industriais*. Master in Science Thesis, CPGEI-CEFET-PR. Curitiba (Brazil) 2000.
- (Rosinha et al., 2000) Rosinha L. F., Koscianski A., Stadzisz P. C., and Künzle, L. A.: *Arquitetura Aplicada ao Projeto de FMS*. XII Congresso Brasileiro de Automática (p.1012-1017). Florianópolis, 2000.
- (Rosenschein and Zlotkin, 1994) Rosenschein J. S. and Zlotkin G.: *Rules of encounter: design conventions for automated negotiation among computers*. MIT Press Cambridge (Mass.), 1994.
- (Russell et Norvig, 1995) Russell S. and Norvig P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
- (Russell et Norvig, 2003) Russell S. and Norvig P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd ed., 2003.
- (Sautter, 1991) Sautter F. T.: *Escalonamento da Produção em Sistemas Flexíveis de Manufatura*. Master in Science Thesis, CPGII/CEFET-PR. Curitiba (Brazil) 1991.



- (Schastai et al., 2004) Schastai V., Lima E.A., Künzle L.A.: *Sequence Analysis for Time Petri Nets*. Workshop on Discrete Event Systems, 7th edition, 2004, Reims, 2004.
- (Schmeil, 1999) Schmeil M. A. H.: *Sistemas Multiagente na Modelação da Estrutura e Relações de Contratação de Organizações*. Ph.D. Thesis, Faculdade de Engenharia Universidade do Porto, 1999.
- (Schmid, 1995) Schmid H. A.: *Creating the Architecture of a Manufacturing Framework by Design Patterns*. Fachbereich Informatik, Fachhochschule konstanz. OOPSLA'95, 1995.
- (Schriber et Brunner, 1996) Schriber T. J. and Brunner D. T.: *Inside Simulation Software: how it works and why it matters*. Proceedings of the 1996 Winter Simulation Conference (ed.: J. M. Charnes, D. J. Morrice, D. T. Brunner, J. J. Swain), 1996.
- (Setti, 1998) Setti J.A.P.: *Simulação da Síntese de Controle e Arquitetura de Robôs Manipuladores no Sistema ANALYTICE*. Master in Science Thesis, CPGEI/CEFET-PR. Curitiba (Brazil) 1998.
- (Silva, 2001) Silva P.R.O.: *Especificação de uma Arquitetura de Supervisão Industrial para o Simulador de FMS ANALYTICE II*. Master in Science Thesis, CPGEI/CEFET-PR. Curitiba (Brazil) 2001
- (Silva et al., 1998) Silva M., Teruel E., Valette R., and Pingaud H.: *Petri nets and production systems*. In *Lecture on Petri nets II: applications*. Lectures notes in Computer Science 1492 (pp. 85-124). Springer Verlag, 1998.
- (Silva et al., 2000) Silva F., Castilho M., and Künzle L.A.: *Petriplan: a new algorithm for plan generation*. The International Joint Conference IBERAMIA'2000 - SBIA'2000, 2000, Atibaia.
- (Simão, 2001) Simão J. M.: *Proposta de uma Arquitetura de Controle para Sistemas Flexíveis de Manufatura Baseada em Regras e Agentes*. Master in Science Thesis, CPGEI/CEFET-PR. Curitiba-PR – Brazil. March 2001.
- (Simão et al. I, 2001) Simão J. M., Silva P. R. O. da, Stadzisz P. C., and Künzle L. A.: *Atividades de Controle e Supervisão Assistidas por um Simulador de FMS*. I Congresso Brasileiro de Engenharia da Fabricação (COBEF). Curitiba (Brazil) 2001.
- (Simão et at. II, 2001) Simão J.M., Silva P. R. O. da, Stadzisz P.C., and Künzle L.A.: *Arquitetura de Software de Controle Orientada a Regras e Agentes para Sistemas Automatizados de Manufatura*. Simpósio Brasileiro de Automação Inteligente (SBAI), Canela-RS Brazil, 2001.
- (Simão et al. III, 2001) Simão J. M., Silva P.R.O. da, Stadzisz P.C., and Künzle L.A.: *Rule and Agent oriented Software Architecture for Controlling Automated Manufacturing Systems*. Frontiers in Artificial Intelligence and Applications ("Logic, Artificial Intelligence and Robotics" LAPTEC 2001 Edited by J. M. Abe e J. I. da Silva Filho). Vol.71, IOS Press, Amsterdam - The Netherlands. ISBN: 1 58603 206 2.
- (Simão et al., 2002) Simão J. M., Quinaia M. A. and Stadzisz P. C.: *Um Padrão Arquitetural para Sistemas Computacionais de Controle Supervisório*. Second Latin American Conference on Pattern Languages of Programing (SugarLoafPLoP'03), Itaipava-RJ, Brazil, 2002.
- (Simão et al. I, 2003) Simão J. M., Fabro J. A., Stadzisz P. C., Arruda L. V. R. and Ishimatsu S.: *An Agent-Oriented Fuzzy Inference Engine*. 6º Simpósio Brasileiro de Automação Inteligente (SBAI). Bauru (Brazil) 2003.
- (Simão et al. II, 2003) Simão J. M., Stadzisz P. C. and Künzle L. A.: *Rule and Agent-Oriented Architecture to Discrete Control Applied as Petri Net Players*. Frontiers in Artificial Intelligence and Applications ("Advances in Logic, Artificial Intelligence and Robotics" LAPTEC

- 2003). Vol. 101 (pp. 121-129) IOS Press, Amsterdam - The Netherlands. ISSN: 0922-6389.
- (Simão et al. 2005) Simão J.M., Stadzisz P.C, Morel G.: *Manufacturing Execution System For Customised Production*. III Congresso Brasileiro de Engenharia da Fabricação (COBEF). Joinville (Brazil) 2005. Accpet in January 2005.
- (Simão et Stadzisz, 2002) Simão J. M. and Stadzisz P.C.: *An Agent-Oriented Inference Engine applied for Supervisory Control of Automated Manufacturing Systems*. Frontiers in Artificial Intelligence and Applications ("Advances in Logic, Artificial Intelligence and Robotics" LAPTEC 2002 Edited by J. M. Abe e J. I. da Silva Filho). Vol. 85 (pp 234-241), IOS Press, Amsterdam - The Netherlands. ISSN: 0922-6389.
- (Solot et Van Vliet, 1994) Solot P. and Van Vliet M.: *Analytical Models for FMS Design Optimization: A Survey*. The International Journal of Flexible Manufacturing Systems, Kluwer Academic Publishers, 6 (1994), pp 209-233.
- (Souza, 1991) Souza J.U.F.: *Tecnologia de Grupo: Algoritmos e Ferramenta Gráfica*. Master In Science Thesis, CPGEI/CEFET-PR. Curitiba (Brazil) 1991.
- (Stadzisz, 1990) Stadzisz P.C.: *Especificação de um Ambiente Computacional para Auxílio ao Projeto do Arranjo Físico de Sistemas Flexíveis de Manufatura*. Master in Science Thesis, CPGII/CEFET-PR. Curitiba (Brazil) 1990.
- (Stadzisz et al, 1991) Stadzisz P.C., Künzle L.A., and Tazza M.: *A Simulation Architecture for Flexible Manufacturing Systems*. Proceedings of the International Conference on Information and System. (p.258 – 261). HangZhou 1991.
- (Stadzisz et al., 2003) Stadzisz P. C., Simão J. M. and Quinaia M. A.: *A Pattern System to Supervisory Control of Automated Manufacturing System*. Third Latin American Conference on Pattern Languages of Programing (SugarLoafPLoP'03), Porto de Galinhas-PE Brazil, (August) 2003.
- (Stadzisz et Doll, 2002) Stadzisz P.C., Doll L.M.: *Utilizando Redes de Petri na Modelagem da Dinâmica de Sistemas Orientados a Objetos*. In: 2a. Jornada Ibero-Americana de Engenharia de Software e Engenharia do Conhecimento. Salvador (Brazil) 2002.
- (Stadzisz et Henrioud, 1998) Stadzisz P.C. and Henrioud J.M. *An Integrated Approach for the Design of Multi-Product Assembly Systems*. Computers in Industry. v.36 (pp.21 – 29), 1998.
- (Stadzisz et Tazza, 1992) Stadzisz P.C. and Tazza M.: *Um Método para o Projeto do Arranjo Físico de Sistemas Flexíveis de Manufatura*. 9º Congresso Brasileiro de Automática. Vitória (Brazil) 1992.
- (Tacla, 1993) Tacla C. A.: *Controle de Sistemas Flexíveis de Manufatura – Especificação do Nível Estação*. Master in Science Thesis, CPGEI/CEFET-PR. Curitiba (Brazil) 1993.
- (Tacla et al., 1994) Tacla C.A., Tazza M., and Künzle L.A.: *Control System For FMS Workstations*. IFIP WG5.7 Working Conference on Evaluation of Production Management Methods (pp.117-124). Gramado (Brazil) 1994.
- (Tacla et al., 1997) Tacla C.A., Achraf O., and Tazza M.: *Software Tool for Manufacturing Systems Simulation and Performance Evaluation*. Proceedings of IEEE International Conference on System, Man and Cybernetics, v.2 (pp.1285-1290).Orlando (USA) 1997.
- (Tacla et Barthes, 2003) Tacla C.A. and Barthes J-P.: *A Multi-Agent System for Acquiring and Sharing Lessons Learned*. Computers in Industry, v.52 (pp.5- 6), 2003.

- (Tacla et Tazza, 1995) Tacla C. and Tazza M.: *Ferramenta de Projeto e Análise de Sistemas Flexíveis de Manufatura*. In: Manufatura Integrado por Computador: Contexto, Tendências, Técnicas. Fundação CEFETMINAS pp.: 227 - 244, 1995.. Minas Gerais, MG, Brazil
- (Tacla et Tazza, 1996) Tacla C.A. and Tazza M.: *Petri Net Based Tool For Shop-Floor Control Design And Simulation*. 1st Advances in Industrial Engineering Applications and Practice. Texas (USA) 1996.
- (Tazza et al., 1990) Tazza M., Stadzisz P.C., Souza J.H.F.: *CELULA-ED : Software para o Projeto de Células Flexíveis de Manufatura*. Anais do IV Congresso Latino-Americano de Controle Automático (pp.35-40). Puebla 1990.
- (Tazza et al. 1992) Tazza M., Stadzisz P.C., Künzle L.A., and Tacla C.A.: *ANALYTICE : Ferramenta para o Projeto e Análise de Sistemas Flexíveis de Manufatura*. 9º Congresso Brasileiro de Automática (CBA) (pp. 1078-1083). Vitória (Brazil) 1992.
- (Tharumarajah et al., 1996) Tharumarajah A., Wells A. J., and Nemes L.: *Comparison of Bionic, Fractal and Holonic Manufacturing System Concepts*. International Journal of Computer Integrated Manufacturing, Vol. 9, No.3 (pp. 217-226), 1996.
- (Tharumarajah et al., 1998) Tharumarajah A., Wells A. J., and Nemes L.: *Comparison Of Emerging Manufacturing Concepts*. IEEE, CSIRO Manufacturing Science & Technology, Preston. Victoria (Australia) 1998.
- (Terzi et al., 2003) Terzi S., Cavalieri S., Garetti M., Macchi M., and Panetto H.: *Development of a Reference Architecture for Benchmarking Service of Manufacturing Systems*. 7<sup>th</sup> IFAC Workshop on Intelligent Manufacturing Systems, IMS 2003 PREPRINTS (pp. 227 – 232). Budapest (Hungary) 2003.
- (Valckenaers, 2001) Valckenaers, P.: *Special issue: Holonic Manufacturing Systems*. Computers in Industry. Volume 46, Issue 3, (pp. 233-331), October 2001.
- (Valckenaers et al., 1998) Valckenaers P., Van Brussel H., Bongaerts, L., Wyns J., and Peeters P.: *Holonic Manufacturing Systems: Architecture and Methodology*. International Federation of Automatic Control - Preprints of The 5th IFAC Workshop on Intelligent Manufacturing Systems, Gramado - RS, Brazil, November 1998.
- (Valckenaers et al., 2003) Valckenaers P., Van Brussel H., Hadeli, Bochmann O., Geramin B. S. and Zamfirescu C.: *On the Design of Emergent Systems: an Investigation of Integration and Interoperability Issue*. In (Morel et Grabot, 2003) pp. 377-393.
- (Van Brussel, 1994) Van Brussel H.: *Holonic Manufacturing Systems, The Vision Matching The Problem*. Proc. of the 1st European conference on Holonic Manufacturing Systems. Hannover (Germany) 1994.
- (Van Brussel et al., 1998) Van Brussel H., Wyns J., Valckenaers P., Bongaerts L., and Peeters P.: *Reference Architecture for Holonic Manufacturing Systems PROSA*. Computers in Industry, Vol. 37 (pp. 255-274), 1998.
- (Waldrop, 1992) Waldrop M. C.: *Complexity, The Emerging Science at the Edge of Order and Chaos*. VIKING, Penguin group, 1992. Cited in (Wyns, 1999).
- (Warnecke, 1992) Warnecke H. J.: *Die Fraktale Fabrik, Revolution der Unternehmen-Skulturr*. Springer, Berlin, German, 1992 (in German). Cited in (Wyns, 1999).
- (Viswanadham et al., 1995) Viswanadham N., Pattipati, K. R., and Gopalakrishna, V.: *Performability Studies of Automated Manufacturing Systems with Multiple Part Types*. IEEE Transactions on Robotics and Automation, Vol. 11, no. 5 (pp. 692-709), 1995.

- (Wyns, 1999) Wyns J.: *Reference Architecture for Holonic Manufacturing Systems – The Key to Support Evolution and Reconfiguration*. Ph.D. Thesis PMA/Katholieke Universiteit Leuven, 1999.
- (Yufeng et Shuzhen, 1999) Yufeng L. and Shuzhen Y.: *Research On The Multi-Agent Model Of Autonomous Distributed Control System*. In 31 International Conference Technology of Object-Oriented Language and Systems, IEEE Press. China 1999.
- (Yoshikawa, 1995) Yoshikawa H.: *Manufacturing and the 21st Century – Intelligent Manufacturing Systems and the Renaissance of the Manufacturing Industry*. Technological Forecasting and Social Change. 49 (pp.195-213). In: (Morel et al., 2003).
- (Zeigler et al, 1996) Zeigler B. P., Cho T. H., and Rozenblit J. W.: *A Knowledge-Based Simulation Environment for Hierarchical Flexible Manufacturing*. IEEE Transactions on Systems, Man and Cybernetics, vol. 26, n. 1 (pp. 81-90), January 1996.

# Abbreviations

AGV	–	Auto Guided Vehicles.
AI	–	Artificial Intelligence.
AIPL	–	<i>Ateliers Inter-établissements de Productique Lorrain.</i>
B2M	–	Business to Manufacturing.
BM	–	Behaviour Model.
CAD	–	Computer Aided Design.
CIM	–	Computer Integrated Manufacturing.
CNC	–	Computerized NC.
CORBA	–	Common Object Request Broker Architecture.
DAI	–	Distributed AI.
DCS	–	Distributed Control System.
DEC	–	Discrete Event Control.
ERP	–	Enterprise Requirements Planning.
ES	–	Expert System.
FAC	–	Flexible Assembly Cell.
FMS	–	Flexible Manufacturing System.
HC	–	Holonic Control.
HCS	–	Holonic Control System.
HDCS	–	Holonic Discrete Control System.
HFAC	–	Holonic FAC.
HFM	–	Holonic Flow Management.
HFS	–	Holonic Fault Supervision.
HL	–	Holon.
HMES	–	Holonic Manufacturing Execution System.
HMS	–	Holonic Manufacturing System.
HPla	–	Holonic Planning.
HSC	–	Holonic Supervisory Control.
HSch	–	Holonic Scheduling.
HSFC	–	Holonic Shop Floor Control.
IE	–	Inference Engine.
IMS	–	Intelligent Manufacturing System.
IiM	–	Integration in Manufacturing.
LHCS	–	Large Holonic Control System.

MAS	–	Multi-agent System.
MS	–	Manufacturing System
MES	–	Manufacturing Execution System.
NC	–	Numeric Control.
OO	–	Object Orientation or Object Oriented.
ORB	–	Object Request Broker.
PM	–	Production Memory.
PROSA	–	Product-Resource-Order-Staff Architecture.
PS	–	Production System.
RBS	–	Rule Based System.
RFID	–	Radio Frequency Identification.
SC	–	Supervisory Control.
SCADA	–	Supervisory Control and Data Acquisition.
SCM	–	Supply Chain Management.
SE	–	Software Engineering.
SHL	–	Subholon.
SFC	–	Shop Floor Control.
UML	–	Unified Modeling Language.
WM	–	Work Memory.
WS	–	Workstation.



## SHORT ABSTRACT

The tendencies about production demand agile Manufacturing System (MS). Thus, the holonic approach has been proposed. In Holonic MS (HMS), entities such as resources and products have certain smartness. These entities, called holons (HLs), have their collaborations organized by the Holonic Control (HC). Actually, HMS development requires tools for design and test. In this doctoral thesis, it is proposed a meta-model for process-driven HC, based on Rule Base System, which is applied and improves a simulation tool. The HC causal relations are treated by Rule agents that receive factual knowledge from Resource-HLs and deliberate about actions, such as Resource-HL coordination. The inference happens via a notification net. This approach allows high reactivity, entity decoupling, determinism, conflict treatment, and coherent control implementation and expression. The control mechanisms emerge from the knowledge of experts. Also, the solution is interpretable as product-driven control, a tendency in HMS, whereby entities like Smart-Product-HLs use Resource-HLs to reach customized productions. In the meta-model case, their interactions are organized by improved Rules.

**KEYWORDS:** Holonic Manufacturing System (HMS), Holonic Control Meta-Model, Rule and Agent Oriented Control, Product Driven Control, Simulation Tool.

## RESUME CONCIS

Les tendances autour de la production exigent des Systèmes Manufacturiers (MS) agiles. Ainsi, l'approche holonique a été proposée. Dans les MS Holoniques (HMS), les entités comme les ressources et les produits ont une certaine intelligence. Ces entités, appelées *holons* (HLs), ont leurs collaborations organisées par le Contrôle Holonique (HC). En effet, le développement des HMS demande des outils d'aide au projet et aux tests. Dans cette thèse, il est proposé un meta-modèle pour le HC dans une orientation « processus », inspiré des Systèmes à Base de Règles, qui est implanté sur un outil de simulation modifié spécifiquement. Les relations causales du HC sont traitées par des agents *Rules* qui reçoivent de la connaissance factuelle des *Resource-HLs* et délibèrent au sujet des actions, comme la coordination de *Resource-HLs*. L'inférence se passe dans une chaîne de notifications. Cette approche permet une haute réactivité, le découplage des éléments, le déterminisme, la résolution de conflit et l'implémentation et l'expression cohérente du contrôle. Les mécanismes de contrôle sont émergés à partir de la connaissance des experts. De plus, la solution est interprétable comme orientée « produit », une tendance dans les HMS où des entités comme les *Smart-Product-HLs* utilisent des *Resource-HLs* pour obtenir des productions customisées. Dans le cas du meta-modèle, leurs interactions sont organisées en utilisant des *Rules* améliorées.

**MOTS-CLES :** Système Manufacturier Holonique, Meta-Modèle pour Contrôle Holonique, Contrôle Orienté à Règles et Agents, Contrôle Orienté Produit, Outil de Simulation.

## RESUMO CONCISO

As tendências relativas à produção exigem Sistemas Manufatureiros (MS) ágeis. Assim sendo, a abordagem holônica foi proposta. Nos MS Holônicos (HMS), as entidades como os recursos e os produtos têm uma certa inteligência. Estas entidades, chamadas de Holons (HLs), têm suas colaborações organizadas por um Controle Holônico (HC). De fato, o desenvolvimento de HMS requer ferramentas de ajuda para projeto e testes. Nesta tese, é proposto um meta-modelo para o HC em uma orientação ao “processo”, inspirado em Sistemas Baseados em Regras, que é implantado sobre uma ferramenta de simulação evoluindo-a. As relações causais do HC são tratadas por agentes *Rules* que recebem conhecimento factual dos *Resource-HLs* e deliberam sobre ações como a coordenação dos *Resource-HLs*. A inferência se passa numa cadeia de notificações. Esta abordagem permite uma alta reatividade, o desacoplamento dos elementos, o determinismo, a resolução de conflitos e a implementação e expressão coerente do controle. Os mecanismos de controle emergem a partir do conhecimento dos especialistas. Adicionalmente, a solução é interpretada como orientada ao “produto”, uma tendência nos HMS onde entidades como *Smart-Product-HLs* utilizam *Resource-HLs* para obter produções personalizadas. No meta-modelo, suas interações são organizadas utilizando Regras aprimoradas.

**PALAVRAS CHAVES:** Sistemas de Manufatura Holônicos, Meta-Modelo para Controle Holônico, Controle Orientado a Regras e a Agentes, Controle Orientado ao Produto, Ferramenta de Simulação.

### ÁREAS / SUB-ÁREAS DO CONHECIMENTO :

03.08.01.02-8 Planejamento / Projeto e Controle de Sistemas de Produção  
01.03.03.00-6 Metodologia e Técnicas de Computação  
01.03.04.00-2 Sistemas de Computação

ano  
2005

Nº: 12