FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

J. Manuel Feliz-Teixeira

Flexible Supply Chain Simulation

Thesis

01 MARCH 2006

Text submitted in partial fulfilment for the degree of Doctor in Sciences of Engineering

Supervisor António E. S. Carvalho Brito
Advisor Richard Saw

RESEARCH SPONSORED BY THE:

EUROPEAN UNION
European Social Fund (III framework)

THROUGH THE:

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR Portugal

PUBLISHER:

Publindústria®
Produção de Comunicação, Lda.

First Edition:
Porto, 1 March 2006

ISBN: 972-8953-04-6

Legal deposit: 239362/06

Original cover art:
Jorge Pereira

Publisher:
Publindústria, Produção de Comunicação
Pr. Da Corujeira, 38 – Apt.3825
4300-144 Porto
Portugal
Tel: +351.22.589.96.20
Fax: +351.22.589.96.29
Email: geral@publindustria.pt
URL: http://www.publindustria.pt

Flexible Supply Chain Simulation

Thesis

J. Manuel Feliz-Teixeira*

01 March 2006

Text submitted in partial fulfilment for the degree of
Doctor in Sciences of Engineering

Supervisor: António E. S. Carvalho Brito
Advisor: Richard Saw

Research sponsored by the:

European Social Fund
(III framework)

Through the:

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR Portugal

* Complete name: José Manuel Feliz Dias Teixeira

To my mother and my father,
and my old professors of Physics.

J. Manuel Feliz-Teixeira

English revision: Deidré Matthee

Abstract

We live in a time where many different approaches for simulating systems are already being used and explored worldwide, mainly due to the last decade's explosion in high level programming languages and the vertiginous level of information conceded by the Internet. The simulation of Supply Chain systems is no exception. However, it is still very difficult to obtain information about the details behind most of the Supply Chain simulators announced out there, for they easily turn into products of high value to those who develop them, as well as to the consultancy agencies that make them highly rentable. But the world of simulation has also changed, from the trustful and innocent academic old-fashioned reliability to the present suspicious and voracious days of modern commerce. In reality, there are no available books or articles or any other texts on how to project a Supply Chain simulator obviously, therefore my option has been to build up my own personal approach from the ground.

This thesis, based on some hypotheses of flexibility, intends to expose the ideas behind such an approach, as well as to document the relevant steps that have been taken to reach this challenge, from the bibliographic review and the study of conceptual issues on Supply Chain systems and trends of simulation, to the conception and development of a practical simulator written in C++ with which some results and conclusions have been obtained.

Sumário

Vivemos num tempo em que uma grande variedade de métodos de simulação são utilizados e explorados em todo o mundo, principalmente devido à explosão das linguagens de programação de alto nível verificada na última década, e à vertiginosa capacidade de comunicar introduzida pela generalização da Internet. A simulação de Cadeias de Distribuição não é excepção. Contudo, continua a ser francamente difícil obter-se informação sobre os detalhes de construção dos simuladores de Cadeias de Distribuição anunciados, uma vez que estas aplicações facilmente se convertem em produtos de elevado valor, para quem os desenvolve e para as agências de consultoria que os tornam altamente rentáveis. Mas o mundo da simulação também mudou, da confiança académica atribuída aos “velhos” tempos, ao actual ambiente de suspeita e voracidade de moderno comércio. De facto, não há livros ou artigos ou quaisquer textos onde se ensine a projectar um simulador de Cadeias de Distribuição, pelo que a minha opção foi desenvolver a minha própria versão desde raiz.

A presente tese, apoiada em hipóteses de flexibilidade, tenciona expor as ideias por detrás desse desafio, desde a revisão bibliográfica e o estudo das Cadeias de Distribuição e das tendências da simulação, até à concepção e desenvolvimento de um simulador prático escrito em C++ com o qual foram obtidos alguns resultados.

Résumé

Nous vivons dans un moment où plusieurs approches de simulation sont employées et déjà explorées dans le monde entier, surtout grâce à l'explosion des langages de programmation de haut niveau vérifiée à la dernière décennie et au niveau vertigineux de la communication assurée par l'Internet. La simulation des chaînes de distribution n'est aucune exception. Cependant, il est toujours très difficile d'obtenir des informations sur les détails derrière la plupart des simulateurs de chaînes de distribution annoncés, parce qu’ils se transforment facilement en produits de valeur élevée pour ceux qui les développent, aussi bien que pour les agences de consultation qui les rendent fortement rentables. Mais le monde de la simulation a également changé, dès la crédible et démodée fiabilité académique aux soupçonneux et voraces jours actuels du commerce moderne. En fait, il n'y a aucun livre ou article disponible ou aucun autre texte sur la façon de projeter un simulateur de chaînes de distribution, donc mon option a été celle de développer mon approche personnelle dès le début.

Cette thèse, basée sur quelques hypothèses de flexibilité, prévoit d’exposer les idées derrière une telle approche, dès la révision bibliographique, l'étude des issues à propos des chaînes de distribution et les tendances de la simulation jusqu’à la conception et au développement d'un simulateur pratique écrit dans C++ avec lequel quelques résultats et conclusions ont été obtenus.

Acknowledgements

António E. S. Carvalho Brito, *GEIN, FEUP, Universidade do Porto, Portugal*
J. Barros Basto, *GEIN, FEUP, Universidade do Porto, Portugal*
J. António Sarsfield Cabral, *GEIN, FEUP, Universidade do Porto, Portugal*
Alcibíades Paulo Guedes, *Escola de Gestão do Porto (EGP), Portugal*
Jorge Freire Sousa, *GEIN, FEUP, Universidade do Porto, Portugal*
Francisco Vasques and **others**, *GEIN, FEUP, Universidade do Porto, Portugal*
Richard Saw, *University of Cranfield, UK*
Christine Rutherford, *University of Cranfield, UK*
Peter Klaus, *Friederich-Alexander University, Erlangen-Nürnberg, Germany*
Gunter Prockl, *Friederich-Alexander University, Erlangen-Nürnberg, Germany*
Achmed Nasa, *Barkawi & Partner GmbH, München, Germany*
Andreas Baader, *Barkawi & Partner GmbH, München, Germany*
Joern W. Ewaldt, *Barkawi & Partner GmbH, München, Germany*
Lutz Arnold, *Barkawi & Partner GmbH, München, Germany*
Juan Florencio Martin Arnanz, *BMW headquarters, München, Germany*
Cláudio Fernandes, *INFENION, München, Germany*
Fátima Rateb, *Universidad de Granada, Spain*
José Carlos Fernandes, *GalpEnergia, Porto, Portugal*

And others:

Noreen, Hilary and **others**, *University of Cranfield, UK*
Ricardo, George and **others**, *University of Cranfield, UK*
Soledade, Isabel, Carmo and **others,** *GEIN, FEUP, Universidade do Porto, Portugal*
José Ramiro Teixeira Santos, *Frankfurt, Germany*
Franciska Nasa and **Cherima Nasa**, *Nürnberg, Germany*
Cláudio Fernandes & Paula, *München, Germany*
Fátima Rateb, Anne Rateb, *Paris, France*
J. Paulo Feliz Teixeira, *Viseu, Portugal*
Pedro & Deidré, *Porto, Portugal*

Specially, I would like to emphasize my gratitude to my supervisor **António E. S. Carvalho Brito**, for his patience, sympathy and excellence, which always pushed me to finish this project, and to **Richard Saw** who kindly guided me in the visits to the University of Cranfield, for his advice, pragmatism, sympathy and sense of humour.

Sources of copyright figures:

Fig. 1.1 - Courtesy of Antique Automobile Club of America, http://www.aaca.org/

Figs. 1.2; 1.3; 1.4; 1.5; 1.6; 1.7 - Courtesy of Computer History Museum, http://www.computerhistory.org/

Fig. 1.14; - Courtesy of TECNOMATIX, www.tecnomatix.de/

Fig. 1.17 – Courtesy of Fujitsu-Siemens, http://www.fujitsu-siemens.com/

Fig. 1.20 – Courtesy of eM-Plant, http://www.emplant.de/simulation.html

Fig. 1.21 – Courtesy of CPN Group, University of Aarhus, Denmark, http://www.daimi.au.dk/CPNtools/

Fig. 1.22 - Courtesy of Los Alamos National Laboratory, USA, http://cnls.lanl.gov/avalon/

Fig. 3.9; 3.28 – Based on a lost reference, thank you anyway

Fig. 5.4 – Courtesy of GalpEnergia, http://www.galpenergia.com/

Table of Contents

FLEXIBLE SUPPLY CHAIN SIMULATION 1
INTRODUCTION 1
RELEVANCE 2
HYPOTHESES 4
METHODOLOGY 4
THESIS STRUCTURE 6
REFERENCES: 8

CHAPTER 1 11
LOGISTICS, LIFE AND SIMULATION 11
1.1 INTRODUCTION 11
1.2 THE PAST 13
1.2.1 The days of 1950s 13
1.2.2 The decade of 1960 16
1.2.3 The 1970s 19
1.2.4 The 1980s 23
1.2.5 The 1990s 27
1.3 THE PRESENT 33
1.3.1 Life and SC management 34
1.3.2 IT and Simulation 38
1.4 THE FUTURE 44
REFERENCES: 46

CHAPTER 2 51
ON MODELLING THE SUPPLY CHAIN 51
2.1 SYSTEMS AND MODELS 51
2.2 SUPPLY CHAIN OVERVIEW 52
2.2.1 Elements of the chain 52
2.2.2 Flows and resources 56
2.2.3 Management levels and policies 58
2.2.3.1 Inventory policies 60
2.2.3.2 Production policies 62
2.2.3.3 Distribution policies 65
2.2.4 Traditional metrics and flexibility 67
2.2.4.1 Facility related metrics 68
2.2.4.2 Fleet related metrics 71
2.2.4.3 The emphasis on flexibility 71
2.2.5 A step up to holistic metrics? 72

2.3 SUPPLY CHAIN MODELLING 74
2.3.1 System Dynamics 75
2.3.2 Petri nets 75
2.3.3 Sequential objects 76
2.3.4 Distributed objects 78
2.3.5 Agents 80
2.3.6 Web-based 81
2.3.7 Parallel 83
2.4 WHICH APPROACH TO CHOOSE 84
2.4.1 Level of detail 84
2.4.2 Where to stop 85
2.4.3 Simulation intents 87
2.5 MODEL CONSISTENCY, ACCURACY 89
2.6 GETTING RESULTS WITH THE MODEL 92
2.7 FINAL COMMENTS 94
REFERENCES: 97

CHAPTER 3 101
THE FLEXIBLE SUPPLY CHAIN APPROACH 101
3.1 INTRODUCTION 101
3.2 THE BASIC IDEA 102
3.3 A GENERIC NETWORK CONCEPT 103
3.3.1 The customer-supplier-unit 105
3.3.2 Connecting the CSUs 106
3.3.3 More about delivery 107
3.4 MODELLING THE CSU COSTS 108
3.4.1 Prices and costs 109
3.4.2 Purchasing costs 109
3.4.3 Stocking costs 110
3.4.3.1 Deeper into the holding costs 111
3.4.4 Manufacturing costs 112
3.4.5 Delivery costs 113
3.4.5.1 Another method for transport costs 113
3.4.6 Management costs 114
3.4.7 Total and global costs 114
3.4.8 The base price of the products 115
3.5 INFORMATION AND MONEY FLOWS 115
3.6 PRODUCTS AND PRODUCT-BOXES 116
3.7 MODELLING THE CSU RESOURCES 117
3.7.1 The stock 117
3.7.1.1 General information 117
3.7.1.2 Ordering policy 118

3.7.1.3 Technology 120
3.7.2 The fleet 121
3.7.3 The production section 121
3.7.3.1 Manufacturing policy 122
3.7.4 The management section 123
3.8 MODELLING THE CSU ACTIVITY 124
3.8.1 Process of purchase (or ordering) 125
3.8.2 Process of stocking 125
3.8.3 Process of delivery 127
3.8.4 Process of manufacturing 128
3.9 PRODUCTS, ORDERS AND VEHICLES 128
3.9.1 About the system products 129
3.9.2 The material-order structure 129
3.9.3 Modelling the vehicles 130
3.10 ABOUT THE FLEXIBILITY (MATRIX) 131
3.10.1 The rigidity concept 131
3.10.2 Matrix representation 133
REFERENCES: 135

CHAPTER 4 137
IMPLEMENTATION AND OVERVIEW 137
4.1 INTRODUCTION 137
4.2 THE SIMULATOR STRUCTURE 137
4.2.1 The classic proposal 138
4.2.2 The OOP tendency 139
4.3 DISCRETE EVENT MODELLING 139
4.3.1 System states and time advance 140
4.3.2 Events versus activity cycles 140
4.3.3 The SimEvent object 141
4.4 THE BASIC ENTITY 141
4.4.1 The SimEntity object 142
4.5 LISTING THE EVENT IDS 143
4.6 CODING THE STATES (ACTIVITIES) 143
4.7 THE SIMULATOR OBJECT 144
4.7.1 The schedule method 145
4.7.2 Random Normal generator 146
4.7.3 The executive loop 146
4.7.4 The SimDlg loop controller 147
4.8 ENTITY MOTION (SIMMOVIE) 149
4.9 A DAILY MARK (SIMTIMEMARK) 150
4.10 SOME UTILITIES 151
4.10.1 The SPECTRUM class 151

4.10.2 The graph window (GraphDlg) 152
4.10.3 Visual modelling (Area) 153
4.11 SUPPLY CHAIN ENTITIES 154
4.11.1 Paths 155
4.11.2 Nodes 156
4.11.3 Vehicles 159
4.11.4 Material-order (Encomenda) 161
4.11.5 The product 162
4.11.6 Product-box 163
4.11.7 Product-plan 165
4.11.8 The CSU entity 166
4.11.8.1 Configuring the facility 167
4.11.8.2 Icon 167
4.11.8.3 States (activities) 167
4.11.8.4 Event handlers 168
4.11.8.5 Other methods 169
4.11.8.6 Main resources 169
4.11.8.7 Suppliers 169
4.11.8.8 Fleet 170
4.11.8.9 Graphs 170
4.11.8.10 Final metrics 170
4.12 THE SIMULATOR APPLICATION 171
REFERENCES: 175

CHAPTER 5 177
CASE STUDIES AND COMMENTS 177
5.1 INTRODUCTION 177
5.2 VERIFICATION STRATEGY 177
5.3 ABOUT VALIDATING THE SIMULATOR 179
5.4 CASE 1: PROCUREMENT OF ADDITIVES IN THE OIL REFINERY OF PORTO, PORTUGAL 179
5.4.1 About the system 179
5.4.2 Building the model 180
5.4.3 Validation, corrective factors 183
5.4.4 The simulation strategy 185
5.4.5 Final results and comments 187
5.4.6 Conclusions 189
5.5 CASE 2: COMPARING STANDARD AND NAIVE ORDERING POLICIES IN AN INLINE SUPPLY CHAIN 190
5.5.1 Introduction 191
5.5.2 Overview on the KANBAN 191
5.5.3 The in-line Supply Chain 192
5.5.4 Simulating the case 193
5.5.5 Results with the KANBAN 194

5.5.6 Results with the naive BanKan 195
5.5.7 KANBAN versus BanKan 198
5.5.8 Conclusions 200
5.6 CASE 3: DISTRIBUTED SIMULATION GAME FOR SUPPLY CHAIN MANAGEMENT TRAINING 201
5.6.1 Introduction 201
5.6.2 The "Cranfield Blocks Game" 202
5.6.3 The client-server application 203
5.6.4 Results and comments 206
5.6.5 Conclusion 208
5.7 CASE 4: COMPUTING THE RIGIDITY MATRIX OF A DIDACTIC SUPPLY CHAIN 209
5.7.1 Inventory-rigidity-to-demand 209
5.7.2 Interpreting the matrix 210
5.7.3 Conclusions 211
5.8 LAST COMMENTS 212
REFERENCES: 213

CHAPTER 6 215
CONCLUSIONS AND FUTURE WORK 215
6.1 THE FINAL CHAPTER 215
6.2 SPECIFIC ADVANCES 216
6.2.1 The CSU concept 216
6.2.2 New delivery elements 216
6.2.3 New reordering policies 217
6.2.4 The Gk geometric factor 217
6.2.5 The ultimate customer 217
6.2.6 Ordering by Internet 217
6.2.7 Three levels of results 217
6.2.8 Demand steps 218
6.2.9 The theory of flexibility 218
6.2.10 Events receiving pointers 218
6.3 VERIFICATION OF THE HYPOTHESES 219
6.4 GENERAL CONCLUSIONS 221
6.5 FUTURE WORK 221
6.5.1 Rigidity (flexibility) studies 222
6.5.2 Reverse logistics 222
6.5.3 Holistic metrics 222
6.6 CLOSING 224
REFERENCES: 225

BIBLIOGRAPHIC REFERENCES 227

APPENDIX 1 235

C++ SIMULATOR CLASSES 235
CLASS <AREA>: 235
CLASS <CSU>: 235
CLASS <MATERIAL>: 240
CLASS <ENCOMENDA>: 240
CLASS <GRAPHDLG>: 241
CLASS <NODO>: 242
CLASS <PRODUCT>: 243
CLASS <PRODUCTBOX>: 243
CLASS <PRODUCTPLAN>: 245
CLASS <SIMENTITY>: 246
CLASS <SIMEVENT>: 247
CLASS <SIMMOVIE>: 247
CLASS <SIMTIMEMARK>: 247
CLASS <SIMULATOR>: 247
CLASS <SPECTRUM>: 248
CLASS <TRAMO>: 249
CLASS <VEICDATA>: 249
CLASS <VEICULO>: 250
CLASS <VIA>: 251

APPENDIX 2 255
WEB LINKS 255
SOME WEB LINKS ABOUT SIMULATION: 255
USEFUL WEB LINKS FOR SUPPLY CHAIN MANAGEMENT: 256
CONFERENCES, MAGAZINES, JOURNALS, DATABASES: 257
INSTITUTIONS RELATED TO SUPPLY CHAIN MANAGEMENT: 258
OTHER WEB LINKS: 259

INDEX 261

FLEXIBLE SUPPLY CHAIN SIMULATION 1
INTRODUCTION 1
RELEVANCE 2
HYPOTHESES 4
METHODOLOGY 4
THESIS STRUCTURE 6
REFERENCES: 8

Flexible Supply Chain Simulation

Doctoral thesis

© J. Manuel Feliz-Teixeira

01 March 2006

Introduction

"*Pause on the esplanade of La Sabika and gaze upon your surroundings. The city is a lady whose husband is the hill. She is clasped by the belt of the river, And flowers smile like jewels at her throat… La Sabika is a crown upon the brow of Granada, In which the stars yearn to be studded. And Alhambra — God watch over it! — Is a ruby at the crest of that crown.*" wrote the poet Ibn Zamrak in the 14th century about the place where I decided to start this thesis. In reality, it seems that there could not be a better place to start it than in this town of Granada, where people live the modern times of distribution of products and services and the speed induced by such processes, and, at the same time, the inspiration and the tranquillity of the Arabic quarters, where palm trees and little vendors still coexist in an almost unreal harmony.

There is, in truth, some justification for this fact, for this work have been strongly motivated by the idea of conceiving and developing an approach to simulate Supply Chain systems in a way that it could be flexible enough to represent their variety and, in addition to that, analyse them in terms of what is currently referred to as "flexibility". In this place, one is definitely able to identify from the old times human paper chain linking warehouses, retailers and consumers, to the most modern *Information Technology* (IT) and advertising Supply Chains of nowadays. It sounds somehow ironic, however, to realise that in many cases consumers still prefer the former approach.

Concerning the simulation of the Supply Chain, it is important to notice that we live in a time where many different approaches are already being used and explored worldwide, mainly due to the last decade's explosion in the number of people using not only high level language programming but also the advantage of the information exchange made possible by the Internet. We live, in effect, in the times where people invade Internet cafés to constantly communicate with one another by *email* or by *chat*; in the last few years the universe of programmers have exploded, and the usage of simulation tools obviously started

to exhibit the same trend. Nowadays, if not yet simulating everything, humankind is at least already trying to simulate everything.

Nevertheless, and in a certain sense as a paradox, it is still very difficult to obtain detailed information about the concepts behind most of the Supply Chain simulators announced out there, for they easily turn into products of high value to those who develop them, as well as to the consultancy agencies who make them highly rentable. It is a fact that everybody wants to handle a model of his/her problem, from the *United States Department of Defence* (DoD), to the most recondite and anonymous flower producer of *Beira Alta*[1]. Such a wide spread interest explains the "boom" in advertising these precious tools. On the other hand, the world of simulation has also changed dramatically during the last years, from the trustful and somehow innocent academic old-fashioned reliability to the present suspicious and voracious days of modern commerce. The pressure of advertising is so high that it frequently hides the quality of the product, and this is a general phenomenon also observed in simulation tools and certain other complex software tools. In reality, there is not any available books or articles or any other texts on how to project a Supply Chain simulator obviously, and, apart from a few extremist researchers that decide to free the source of their simulators on the Internet, the most reliable information about the structure of the products is only achieved in their *Operation Manuals*, which means, after buying them, or, with some luck, borrowing them from a friend. Of course the *Operations Manual* is not the appropriate document in which one can expect to find much about the structure of the internal processes involved in such tools, or how they have been conceived, but at least some information can be captured from it indirectly. This, and the difficulty of finding new interesting aspects in the literature more than the usual discrete approaches based on *events*, *processes*, *Petri-nets*, etc., as well as on some new trends using "intelligent" entities named *Agents* (Nwana, 1996), which I do not consider an attractive approach to simulate the dynamics of the Supply Chain, in the end the choice have been to build up my own personal approach from the ground.

This text intends to expose the ideas behind the development of such an approach, as well as to document the relevant steps that have been taken to reach that challenge, from the bibliographic review and the study of conceptual issues about Supply Chain systems, their structure and their management, as well as the current trends of simulation, to the conception and the development of a practical simulator written in C++ with which some results and conclusions have been obtained.

Relevance

The simulation of Supply Chain Systems is presently widely recognized as of great value, as these kinds of systems frequently exhibit complex behaviours, too difficult to be represented by analytical equations or other continuous mathematical techniques. The modelling approaches in use at the moment are diverse, as diverse still are the schools of modelling and the objectives for which simulation is used. In general, as we

[1] A Province of Portugal.

will see soon in the chapter reserved for the literature review and the state of the art, *events*, *Petri-nets*, *Bond graphs*, *Agents*, etc., are concurrent paradigms with which modellers continue to try to represent their systems. There is not, however, any special reason for electing any of them as the *primus-inter-pares*, as their usage strongly depends on the school of modelling or on the sympathy the modeller has for a particular paradigm or tool. It was not so surprising to meet a senior researcher at a conference in Naples, three years ago, who kept on using an old version of SIMULA, since this language has continued to fulfil his needs and was already the "mother" of the *Object Oriented Programming* (OOP) paradigm.

In that sense, the approach presented in this work is still based on the classical *next-event* paradigm, although it will be seen that the conceptual model of the Supply Chain already includes some novel features, as it tries to follow issues like *just-in-time* (JIT), KANBAN, *flexibility* and even what some people call *demand pipelines*, for example. These are seen as recent tendencies for research in this field of knowledge, and practically ignored in the common Supply Chain simulators.

Another important note is the fact that most of the present simulators seem to persist in using the common methods to create the model, mainly based on some sort of *queue-activity diagrams* from which the dynamics naturally emerge. To the contrary, the present approach has been conceived in order to encapsulate such "low-level" diagrams and enable the user to faster and easily employ a logic similar to that of the real system, thus reducing the modelling process to the handling of "real" elements such as *transport paths* or even complete *facilities*. Each element is also stochastic, and a set of graphical output data attached to it will be available. Thus, in a certain sense, from the point of view of the modeller, the present approach can be considered not only a paradigm of "real objects" but also, and in a certain way, of "real reasoning".

A last issue that must be mentioned is the universe of applicability of this proposal. This simulator have been developed based on the *next-event* dynamics, and thus it is prepared to handle quite well not only events separated by some seconds, minutes or hours, but also those separated by weeks, months or years. That will depend, of course, on how the modeller configures each element in the simulator, on how he defines the appropriate transportation, or on how he considers the patterns of demand. Therefore, and also since every element is configured by "default", at least theoretically it will be possible to simulate the systems at the *operational* perspective, as well as for *tactical* or *strategic* purposes. In reality, it will not be difficult to obtain results with this approach from a simulation of "some days" to a simulation of "some years". The treatment of the output data and the process of its analyses will be the last term that will establish the point of view in fact considered. It is, however, important to keep in mind that usually the uncertainty of the results will grow as the period of simulation increases.

Hypotheses

"*Flexible Supply Chain Simulation*" has been chosen as the title of this research for two main reasons. Firstly, the adjective "flexible" must be applied to the *simulation* itself, since its structure has been conceived with the intent to allow a wide range of scenarios to be modelled without losing the simplicity of the modelling process, not only in terms of different Supply Chain network structures, but also regarding the usage of different stock and production policies, from the classical approaches of producing-to-stock, to the modern trends of JIT. In this first sense, the hypotheses can be described by the following two statements:

H1) the simulator will be able to represent diverse kinds of Supply Chains in terms of different network structures, and allow the analyst to draw conclusions about their performances at least based on some standard Supply Chain measures.

H2) the simulator will also be able to represent different stocking and producing policies at each facility, as well as to test different kinds of vehicles for delivery, in order for these to be compared by means of some standard measures.

Secondly, the adjective "flexibility" must be applied to the Supply Chain itself, and that means that the present approach is expected to give the user some quantitative indications about the degree of "flexibility" of a certain scenario. The third point of the Hypothesis is, therefore:

H3) by means of this simulator it will be possible to obtain a quantitative measure of the "flexibility" to demand variations, not only related with each facility but as well with the entire Supply Chain structure. Different scenarios are expected to be possibly compared based on these sorts of measures.

Methodology

To achieve the objectives expressed in these hypotheses, a first plan for the program of research was established, inspired by the believe that "*one must first understand the system so that later one is able to model it*". This implied the orientation of the research in two main directions:

a) Simulation: study of the literature and investigation on the "state of the art" by reading articles published in conferences, journals, magazines and on the Internet, as well as about its basis in relevant books, by means of which it was finally possible to classify the various recent tendencies of simulation applied (or applicable) to Supply Chain, and to recognize the merits of each of them.

b) Logistics and Supply Chain: study of the basis of Logistics and Supply Chain management in some relevant books on the field, followed by an extensive literature review focused on scientific articles obtained from magazines, conferences, journals, Internet, etc., with the intent of learning as much as possible (given the time constraints) about the kinds of systems that were to be simulated. This study has been extended to most matters concerning Supply Chain management, but in the end it has been mostly focused on *production* and *inventory* policies, on different delivering *vehicles* and on the understanding of the

various Supply Chain structures, as well as the actual trends and philosophies involving the field. A historical overview on the Supply Chain management has also been accomplished.

Once the knowledge concerning these matters had reached a satisfactory level, a new process was started in parallel, leading to real contacts with people, centres of knowledge, universities and enterprises, with the intent to definitely take a real measure of the problems and ideas occurring in the field, often quite distant from the usually optimistic literature.

These contacts included meetings and discussions with people strongly related to the management of Supply Chains, most notably with the *Management School of Porto - EGP* (Portugal), the *University of Erlangen Nuremberg* (Germany), the enterprises *BARKAWI & Partners* and *BMW* from Munich (Germany), the *Centre for Logistics and Supply Chain Management of the University of Cranfield* (UK), and, finally, some interviews with the *GalpEnergia* company (Portugal).

Parallel to this, the process of implementing and day-by-day improving the C++ Supply Chain simulation application started. This would later lead to the very first version of the Supply Chain simulator, which basis had then been presented at the "*European Simulation and Modelling Conference*", Naples (Italy), in 2003 (Feliz-Teixeira & Brito, 2003) .

This method of framing the research reveals a *top-down* strategy in the process of learning, followed by a *bottom-up* strategy in the development and the implementation. Although only after the learning process reaches the bottom level one can start to implement, this method is probably the most interesting and reliable method to use in projects of this kind, as, without doubt, it ensures that the results will be derived from a good background of knowledge. This scheme also provides the researcher with enough information about the relevant matters, giving him or her the ability to better handle the real problems and even to formulate new suggestions. That was the case of the theory of Supply Chain *flexibility* proposed in this work, and meanwhile published in Feliz-Teixeira & Brito (2004), even if this matter was not specifically related to simulation; and also the case of a small study comparing an inline classical Supply Chain with an imaginary *demand pipeline* chain (Feliz-Teixeira & Brito, 2005) since the last is indicated by some authors as the Supply Chain of the future (Hewitt, 2001).

In order to enable the exploration of any interesting concepts that could meanwhile emerge from the research, efforts have been made to ensure the methodology was kept flexible enough, instead of *a priori* reducing it to a predefined set of constraints. This fact has also made possible the development of a distributed Supply Chain Game (Feliz-Teixeira et al., 2004), for example, with which a class of students seated at different computers could exercise inventory management in a didactical Supply Chain.

Apart from this, several procedures of *verification* have been carried out in the proposed simulation approach, in which the

simulator internal processes were approved, and a first *validation* of its outputs have been achieved by means of simulating some practical cases, as far as possible. The most significant of these was a real case study based on data provided by the company *GalpEnergia*, which resulted in the confirmation of the practical value of the approach, and contributed to testing and confirming the hypotheses.

The hypotheses were finally tested and inferred as true by means of analysing the data obtained in the *GalpEnergia* case, as well as in the *demand pipeline* case and some other appropriate Supply Chain "small paradigms"; and, finally, by the success in the calculation of a quantitative measure of *flexibility* for a didactic Supply Chain, known as the "*Cranfield Blocks Game*" (Saw, 2002).

Thesis structure

The thesis begins with an introduction to Logistics and *Supply Chain Management* (SCM) as an overview of how the concepts related to these matters have evolved since the end of the *Second World War*, as well as of how the computer Simulation, scientific development and even certain aspects of common life have progressed concurrently. This opening results from a general literature review focused on those issues with the intent to establish a general view on the evolution of the times and the state of the arts (chapter1).

Since the present work involves a significant specific knowledge in two different fields, the field of Logistics and SCM and the field of Simulation, and since it is expected that people specifically from these fields do not necessary dominate both matters to an extent adequate for understanding the present work, it was considered justifiable and even convenient that each field would be firstly presented in some detail before moving to the presentation of the work. In a way, this text may, therefore, also be considered didactic. The thesis is mainly about Simulation, but it is applied to a specific type of system, which is known as Supply Chain. It is, therefore, essential to ensure the convergence of knowledge of the two fields to a point that Supply Chain managers and Simulation experts can both understand and use this text. Chapter 2 is, for this reason, dedicated to an overview of the relevant issues concerning both the Supply Chain and the various modelling and simulation technologies used to model these kinds of systems.

In chapter 3, the proposed approach for Supply Chain modelling and simulation is conceptually presented, with each of its elements explained and analysed in detail, with emphasis on its central element, the general Supply Chain facility that we have named *Customer Supplier Unit* (CSU). Modelling the fixed and the variable costs of the Supply Chain, the products, the facility resources, as well as the facility dynamics and the vehicles dynamics, and finally, presenting the theory of *flexibility*, are the type of subjects presented in this chapter.

Chapter 4 is dedicated to the C++ implementation of the current simulation approach, and includes an overview of the structure of the simulator, as well as the code and comments on the most important

object classes in the application. Although there is a good extent of C++ code in this chapter, each important object or structure has also been described by means of accessible diagrams, so that non-C++ users may also benefit from its information. A short overview of the final application for modelling and simulation concludes this chapter.

Following a brief discussion concerning the *verification* and the *validation* of the simulator application, the rest of chapter 5 is reserved for the presentation of some experimental results achieved by simulating a few practical cases. Relevant comments and discussions will be dispersed throughout this chapter.

Finally, chapter 6 is entirely dedicated to the final conclusions and the perspectives for future research. It is here that the verification of the hypotheses is also done, as well as some ideas related with *holistic* measures presented.

NOTE: Bibliographic references are presented at the end of each chapter and also at the end of the thesis.

References:

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2003). *An Approach for Dynamic Supply Chain Modelling.* Paper presented at the 2003 European Simulation and Modelling Conference, Naples, Italy.

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2004). *On Measuring the Supply Chain Flexibility.* Paper presented at the 2004 European Simulation and Modelling Conference, UNESCO, Paris, France.

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2005). *Comparing a Standard and a Naïve Stock Refill Policies by Means of Simulation.* Paper presented at the 2005 European Simulation and Modelling Conference, University of Porto, Porto, Portugal.

Feliz-Teixeira, J. M., Brito, A. E. S. C., et al. (2004). *Distributed Application for Supply Chain Management Training.* Paper presented at the Industrial Simulation Conference 2004, Malaga, Spain.

Hewitt, F. (2001, May 2001). After Supply Chains, Think Demand Pipelines. *Supply Chain Management Review, 5,* 28.

Nwana, H. S. (1996). Software Agents: An Overview. *Knowledge Engineering Review, 11,* 1-40.

Saw, R. (2002). *Cranfield Blocks Game*: Centre for Logistics and Supply Chain Management (CSCM), University of Cranfield, UK.

CHAPTER 1 11
LOGISTICS, LIFE AND SIMULATION 11
1.1 INTRODUCTION 11
1.2 THE PAST 13
1.2.1 The days of 1950s 13
1.2.2 The decade of 1960 16
1.2.3 The 1970s 19
1.2.4 The 1980s 23
1.2.5 The 1990s 27
1.3 THE PRESENT 33
1.3.1 Life and SC management 34
1.3.2 IT and Simulation 38
1.4 THE FUTURE 44
REFERENCES: 46

Chapter 1

Logistics, Life and Simulation

An overview on the times and on the state of the arts

1.1 Introduction

The expression "Logistics" has begun to be used far after mankind have begun to handle logistic processes, and it mainly results from the need man of 20th century felt to classify those processes in order to start treating them in a more scientific way. Taxonomy is many times the beginning of such courses.

Anyhow, as often happens with many other designations, the term "Logistics" is still on the mind of many authors as a kind of diffused idea about its meaning, even if some insist on searching for its ultimate definition, as it can be noticed on reading common literature on the subject. Also in the *Internet* it is easy to realise there is a wide range of such definitions, stated by a wide number of institutions, like the *Council of Logistics Management*[1], the *United States Department of Defence* (DoD), the *Logistix Partners Oy*[2]*, from Helsinki,* the *Websters Dictionary*, the *American Heritage Dictionary*, or the *Canadian Association of Logistics Management*, etc., only to name some of the dozens forming part of a glossary maintained by *LogisticsWorld.com*[3].

Although such diversity of opinions fails to free us from the initial confusion on the matter, it is somehow true that the term is in general accepted to reference an art, a process, a discipline or a science, other than a system or a network, what at least can be considered a positive aspect pointing to a certain agreement.

It is a fact that each one adopts the definition which seems more adapted to his/her purposes or convictions, similarly to what happens in matters of subjectivity, like love or friendship. Nevertheless, there was one of those definitions that touched me most due to an almost naïve simplicity, stated by a little Australian enterprise as follows: "*In an industrial context, logistics means the art and science of obtaining, producing, and distributing material and products in the proper place and in proper quantities. In a military sense, its meaning can also include the movement of personnel.*"[4]

Although this characterization already tries to declare a slight innocent distinction between civil and military "Logistics" (in reality most of the technology have been military driven, till now), the truth is that in

[1] Founded in 1963, is a preeminent association for individuals involved in Logistics and Supply Chain Management (http://www.clm1.org).

[2] A consultancy agency, based in Finland.

[3] http://www.logisticsworld.com/logistics.htm

[4] http://www.homercomputer.com.au

general most people still correlate "Logistics" to the movements of troops, facilities and materials in the war. That is the origin of the term and perhaps still is the best way to visualise the idea behind it. In that sense, one can easily imagine "Logistics" as the overall process which is responsible for fulfilling the needs of the troops on their movements and their stay far from "home", where goods and materials could be easily accessible. Thus, "Logistics" is the operations that "support" the cause.

Of course processes of this kind are already known at least since the time of the Romans, where huge amounts of materials and men had to be moved along huge distances, and for that reason thousands of kilometres of roads have been built across the Empire, to mention just a case, even if those processes could not yet be managed with the perfection of nowadays.

Logistics, therefore, deals with supplying the needs of people, and thus of producing, of storing and handling the products, of their transportation and delivery, as well as of all the sub-processes related with each of this processes, including information. Even if it has never been (and will never be) detached from the military, who still are the masters on this art, with the time Logistics have also naturally spread to civil organizations and systems, turning them more powerful, efficient and reliable.

More than a system, Logistics is thus a huge field of knowledge including different kinds of systems, different issues, and even diverse levels of reasoning, acting in a network of dependencies. That explains why it usually is studied in separated subjects, as procurement, production, storage, delivery, forecast, sales, etc., as well as in some other financial topics.

In a certain way, one could say it would be impossible to simulate Logistics for the same reasons it is impossible to simulate Optics, for example, since both are in fact fields of knowledge, disciplines or arts, and not real systems.

Anyhow, simulation techniques have already been applied with success to some areas of Logistics, for instance, to warehousing, or to production, where normally the systems are already too complex to predict and characterize. And lately, simulation has also been used on studying the "Physical Distribution" across the Logistic network, a complex process including the transport of materials that since the 1960s (Metz, 1998) has evolved to what is nowadays referred to as the "Supply Chain".

This chapter will introduce the fields of Logistics and Simulation in terms of their historical courses and relevancy, and at the same time in relation to the advances in computers, electronics, operations, political and social events, as well as to everyday life, in an interrelated holistic phenomena of development. This is expected to help the reader on better following the close relations existing between the two main fields, similar to what happens between science and engineering, for instance.

In between, also the "state of the art" in each matter will be presented, resulting from a literature review on Simulation and on Logistics and operations management, which slowly will be focused on the most relevant issues for the present work, that is, those related to the Supply Chain.

Finally, some considerations will be made about what probably will be the future tendencies in those fields.

1.2 The past

Half a century ago, in the decade of 1950, the big acceleration that would result in the current world of "all kind of achievable commodities" was beginning. Some years before, America and its allies had won *World War II* and that fact was reflecting on the people a wave of confidence and enthusiasm which clearly would lead to the prosperity of the following decades. The world was preparing itself for the craziest race of the 20th century, already in good use of the early *Scientific Management* concepts due to Frederick W. Taylor (1903), as well as dominating and practising notions like the *Statistical Sampling* for quality control (Shewhart, 1931), the *Economic Order Quantity* (EOQ) from Harris (1913), and many other tools from the field of *Operations Research*, as, for instance, the *Simplex Method for Linear Programming* developed some years earlier.

1.2.1 The days of 1950s

After the war, together with thousands of ordinary Jewish people, many engineers and scientists immigrated or were even invited to join the "*American Dream*" in order to develop and implement the new trends that would lead mankind to the big change. People moved in enormous ships from Europe to America, many of them carrying their entire families, and this mass flow of manpower made possible the fast progress of science, technology, medicine, politics and many other aspects that became marks of the modern American society.

Also in terms of social life, the country was living one of its most promising times, inspired by an "*optimistic vision of a semi-Utopian technological future, including such devices as the flying car*" (Wikipedia, 2004a). A clad black-and-white *SuperMan* was frequently seen on the TV, while the cars in the streets exhibited a certain coloured flying machine style lent by the "*Flash Gordon*" movies of 1930-40, almost as in a dream. Henry Ford, for instance, already with a long practice in mass production, was in 1955 producing models like the popular *Thunderbird* as well as the *Fairlane* (shown in figure 1.1), while for the first time young girls could play with *Barbie* dolls.

Fig. 1.1 *Ford Fairlane*, from 1955

In the streets, the ordinary family man was a relaxed man driving one of these nice and styled heavy cars, while the movement in the towns was growing and the new standards of living changing the day-by-day life. On the minds of America, and in the world in general, there was a unique fervour to believe in a glorious destiny for mankind, obviously inspired by science, while figures like Humphrey Bogart, Kirk Douglas, Lana Turner, Paul Newman and Elizabeth Taylor were breaking the hearts of lovers around

the planet, and Hollywood produced "*Cat on a Hot Tin Roof*", and "*The Bad and the Beautiful*".

At the same time, Wernher Von Braun, the former top expert in ballistics who had achieved the bombing of London from Germany with his *V2*, was now in charge of the ambitious American space program, as director of the *NASA's Marshall Space Flight Centre*. The aggressive German regime had been defeated. Now the objective was the Moon.

Driven by the prodigious mind of John von Neumann, a Hungarian-born scientist living in America who in 1945 outlined the basis of the architecture of modern computer, as well as by the invention of the *transistor* in 1947 by John Bardeen, Walter Brattain and William Shockley, the fifties have also been times of strong computer development and technological incubation as well as the beginning of the trend of reducing the size of electronic devices, till then dominated by the *vacuum tube* technology.

At the end of the 1940s, computers were incredible machines with which only some experts could communicate by means of writing special codes in a *paper tape* or using switches, codes that only later could be directly loaded from a magnetic tape. The output, in characters code, was first presented in a kind of printer and only many years after on a vacuum tube screen. A good computer memory size could be 2,000 binary words (˜0.002 MByte), and an expectable speed around 2,000 operations per second (˜0.002 MHz). The potent UNIVAC-I (in figure 1.2), for instance, which occupied an entire big room, could work with a speed of 1,905 operations per second, a memory size of 1,000 12-digit words and an input/output technology based on magnetic tapes, unityper and printer, and was costing the equivalent of around 1,000,000€, including the "high speed" printer (C.H.M., 2004).

Fig. 1.2 The UNIVAC-I computer, from 1951

But also the number of computers was very small, compared to nowadays, and some of them could need the space of a building floor. During 3 years, IBM would only sell 19 units of its first electronic computer (the 701) to some research laboratories, aircraft companies and federal governments, in the times of 1953 (C.H.M., 2004). But meanwhile transistorized programmable computers appeared in 1956, with the TX-0 built at *MIT's Lincoln Laboratory* (C.H.M., 2004), and from then a great reduction in size would start.

The top of Logistics in that time was the Korean War (1950-53), a particularly savage conflict that succeeded in repelling the North Korean communist invasion of the South Korea, and where America could already practise Logistics as a routine, as documents from the time confirm (N.H.C., 2000). Also during that time, French[5]

[5] The French Vietnam War, during the 1950s, was in reality the seed of a later American war that would stay to the history as the *Vietnam War* (1964-75).

military was already practicing heavy Logistics, while battling Ho Chi Min in Vietnam (TheHistoryPlace, 1999), precisely when *Inventory Management* was first introduced by Thomson Whitin (1953).

However, modern enterprises were still distant from integrated Logistics worries, since they were much more focused on scientific development and production, as was the case of NEC, *Texas Instruments* and *Bell Labs*, for example, all of them related with the digital computer. And even if it was true that the industry of transports was growing worldwide, the term "Logistics" was still a property of the military, and "Supply Chain" literally meant nothing yet.

Instead, the term in use was "Physical Distribution", concerning the distribution of materials between the elements of the supply network, which were still being managed on their own, without any special concepts of partnership, data interchange or network optimization. In a certain way, it is understandable that things were like that, since the most important focus in those times was not on delivery, but on new inventions as well as production lines. Only when the rate of products flow got higher more attention was turned to the distribution process. Nevertheless, "Physical Distribution" was already dealing with freight management, warehousing, material handling, inventory, plant and warehouse site selection, etc., matters which would stay in its domain till today.

About simulation, which actually has been used by scientists since the beginning of science, there is some evidence that ship models were already built for hydraulic experimenting in the 17th century, first in England, then in Holland, France and Russia (R. Müller, 2000), the models of the 1950s where largely dominated by scale sculpts of the systems, continuous mathematical models, as well as mechanical or electromechanical representations[6].

There were, anyhow, already some trials on the development of simulation languages and to introduce mathematical, statistical and stochastic concepts into the computer, what in 1958 lead to the first symposium on "*System Simulation*" held in Baltimore at the *American Institute of Industrial Engineering* (AIIE).

The field of *System Dynamics*[7] was also undergoing changes from the hand-simulation stage to the computer modelling stage (Forrester, 2004), and Richard Bennett was creating the first system dynamics computer modelling language, called SIMPLE (Simulation of Industrial Management Problems with Lots of Equations), while, a year later, in 1959, Phyllis Fox and Alexander Pugh wrote the first version of DYNAMO (DYNAmic MOdels), an improved version of SIMPLE that became the industry standard for over thirty years, as Forrester (2004) observes.

Also important contributions to the field of Simulation were coming from the *RAND Corporation*, with important advances on *"Systems Analysis"*, while general efforts were running to find *mechanical models* for economic, biological and psychological processes (N. F. Morehouse et al. 1950; O.

[6] In fact these models were a kind of analogical automation of a continuous mathematical model, where blocks of differentiation and integration were used to process the output of a differential equation system.

[7] *System Dynamics* is a continuous mathematical modeling methodology developed by Forrester during the 1950s, for studying and managing complex feedback systems. Only the study of the whole system as a feedback system will lead to correct results (Forrester, 2004).

J. Smith, J. M. Erdley 1951; Arnold Tustin 1953; Donald E. Broadbent 1957), as cited by Müller (2000).

One can think of the fifties as dominated by transistorized computers, more compact than those of the previous decade, but still accessible only by military and space research institutions, or by the most powerful civil companies, as it was the case of the telephone company (*Bell Labs*). On the other hand, programming languages were not yet available and during most of the decade the code introduced in those machines was linear, time consuming and dedicated to resolving specific problems. The introduction of FORTRAN was only in 1957[8]. With this language, for the fist time was possible to make use of "DO" programme loops, which extremely have improved computational processes.

Only in the end of the decade, in 1958, Jack Kilby from *Texas Instruments* was presenting the primitive prototype of the *integrated circuit*, the component which would evolve to the technology used in the computers of today. The digital computer, in use since 1944, would be extremely improved by means of this *integration* technique.

1.2.2 The decade of 1960

In one year the *Berlin Wall* would separate East and West Germanys. In two years the tensions with USSR would reach the red limit with the Cuban Missile Crises, bringing the world to the doors of a nuclear confrontation.

In those times, a powerful technology was already available, but still it was especially heavy, space and power demanding and very expensive for the civil society. And it was mainly concentrated in places where the ordinary man would never expect access. For the common man of that time, a computer was probably a kind of mystic machine of electronic intelligence with which some "crazy" scientists could project space ships and Atomic Bombs, and would never ever cross his mind that each one of us would nowadays carry its own. FORTRAN was already available, but there would be no general propose operating systems until the end of the decade, when in 1969 Kenneth Thompson from *Bell Labs* developed the first UNIX operating system (Bell-Labs, 2002). Anyhow, the computer languages flourish, as one of the most interesting challenges of the time was to transform the computer in a multi-propose machine. COBOL, LISP, BASIC and LOGO, were born in the sixties.

Following this tendency, the simulation programming was also passing from its *Period of Advent* to what some future researchers as Richard E. Nance (1993) would identify as the *Formative Period* (1966-1970), with the computer simulation mainly being driven by *simulation language* developments. SIMSCRIPT, the general-purpose simulation language created by Harry Markowitz et al. at *RAND Corporation* in 1963 could now be applied to large discrete simulations (Kinnersley, 1995). During the next year, SIMULA I (SIMUlation LAnguage) was developed in Norway by Kristen Nygaard & Ole-Johan Dahl, and launches the basis for the future *Object Oriented Paradigm*, while IBM produces the

[8] In this year the *European Economic Community* (EEC) was also created , by the Treaties of Rome.

first computer based on silicon chips (IBM360), and researchers recognize the need for a general theory of systems and models separated from the simulation languages (Lackner, 1964).

In the middle of the decade, some analysts could already make use of facilities like the *Computer Aided Design* (CAD), since in 1964 IBM have developed the first CAD Computer, the "System 2250". As Müller (2000) also notices, CAD was advanced from US-military research on space travel, and later made accessible for the public. It would then spread to the civil industries of ships, automobiles, electrical circuits and injection moulds. A higher optimization of industrial processes and product distribution was taking place, and the *National Council of Physical Distribution* was created in the United States. But such pressure imposed by the mass production race was also starting to be reflected on many aspects of social life.

With the industry openly running in the *Mass Production Era*, as Heizer & Render (2001) define the period of 1910-1980, the 1960s were marked by an extraordinary scientific and technological advance mainly imposed by the automotive industry, the armament race of "Cold War" and the "ideological" race of "Space Conquest" (the *regimens* had to sustain their ideologies by constantly exhibiting their powers and their abilities). In the centre of all this, anyhow, was the digital computer.

With the intents of preventing the spread of communist ideology, previous tensions between America and North Vietnam would in 1964 give rise to a new American war. Anyhow, opposing such a war mindset, an unexpected trend was also emerging from ordinary people, which in a certain way could be seen as the awakening of the masses. The memories of the horrors of the Atomic Bombs, which have astonished people worldwide with the events of Hiroshima and Nagasaki, slowly were transforming into a wave of movements for peace and search for different ways of living, mainly among the youth, which a decade later would lead to the affirmation of POP culture as an icon committed to destroy such concentration of powers and turn the political issues more public, more "popular", as well as the technologic advances meanwhile achieved by science. The *Beatles* were already playing since the beginning of the decade, they used electric guitars and electronic sound systems, the *beatnik generation* was also on the move, while common people in America were starting to blame the high rate *Mass Production* for everything from the crisis in higher education to the demise of the family (Stephens, 2003). Not even the launch of Von Braun rockets towards space, which could usually be seen by everyone on the TV, was able to calm down the tensions provoked in the civilian society by the Vietnam War. There was somehow a "revolution" approaching.

Although such "revolution" had first made itself noticed in the artistic field (groups like "*The Beatles*" or "*The Stones*" could easily show the tremendous power contained in the masses), the fact is those ideas slowly have expanded to other population concerns and started to pervade and sway the behaviours of the following decades. The conservative anti-communist society that ten years earlier had watched

"*The Day the Earth Stood Still*"[9] (1951) was now to face a force in another direction, whose tendency was to oppose the war and to suggest the release of individual tensions, that tensions had to be released, not allowed to rise constantly seemed to be the doctrine. Thus, the new proposal implicitly would include the usage of technology for purposes other than war, as entertainment, artistic expression, day-by-day life or even pleasure. The society of consumption was fast developing. May 1968 took place in Paris, once again confirming the anti-war trend and the potential hidden in the masses, which were not anymore the huge human crowd without any voice or aspirations as they seemed to be. And all this would slowly lead the future to the inevitable liberalization of technology[10] observed in the last decades of the 20th century. It seems masses were slowly thinking on products, and such tendency would later set in motion an even bigger acceleration in society change[11].

A compassionate description about the research during these times can be found in Zobel (2001), who in 1964 was working on the analogue computer at the *Sperry Gyroscope Company*, UK, and later on the digital computer. He had been chairman of the UKSIM, a society related to the American *Society for Computer Simulation* (SCS) and the EUROSIM.

In 1965 Man was giving the first steps in the era of minicomputers, with the *Digital Equipment Corp.* introducing the PDP-8 (Fig. 1.3), which was faster, much smaller and one-fifth the price of a small IBM mainframe of that time. The demand of such machines increased, as *Microelectronics* advance with chips technology, and the components get more compact, less power consuming and faster, leading to more memory and more complex operations.

Fig. 1.3 The DEC PDP-8, from 1965

Computer is slowly becoming affordable to society, mainly to people at the universities. Thus, more and more people feel attracted by software development, and general-purpose languages naturally evolve, as well as discrete simulation languages and other computer techniques. In 1967 the first "*Computer World*" magazine is published.

Simulation is obviously moving towards the discrete field, and the main paradigms

[9] Movie where an extraterrestrial comes to Earth to warn of the dangers of Nuclear Bombs, also related to the fears of Communism.

[10] This is the point of view of the author, probably shared by some and rejected by others.

[11] The *Information Society* of nowadays was in fact the result of the popularization of the electronic and digital technologies, including videos, TVs, computers, musical instruments, mobile-phones, networks, computer applications, games, etc., what in our times make the customer the most precious element in the economical system. In fact, due to this popularization of products, devices and services, nowadays customers are measured in millions, an extremely attractive number for those out there thinking of selling…

of discrete simulation take place, most of them resulting from different views on the *state transition diagram* used for system's representation. The most relevant of theses paradigms were the *process-interaction* approach used by Gordon (1961) in his GPSS[12] simulation language, focused on how entities flow through the system and providing a *process* for each of these entities; the *activity-scanning* approach proposed by Buxton & Laski (1962), which represent the model dynamics based on the execution of certain *activities* every time certain conditions are satisfied; the *event-scheduling* approach developed at *Rand Corporation*, describing the model as a sequence of *events* that cause the changes in the state of the system; and later the *three-phase* approach published by Tocher (1963), an interesting compromise between *events* and *activities*. Based on these approaches, languages like CSL, SIMSCRIPT and SIMULA, emerge and become standards.

Moreover, it is important to point out that also *Petri nets*, first introduced by Petri (1962), became since that time an important formalism directed at distributed systems. This approach was inspired on the activity-scanning of automata programming, and turned into a formal and graphical language already using the notions of concurrency, non-determinism and synchronization, and was considered the first general theory for discrete parallel systems to be formulated (P.N.W., 2004). Actually, in contemporary simulation conferences *Petri nets* appear to dominate the taste of a substantial number of researchers.

[12] General Purpose System Simulator

Obviously, all this improvement in the computer field was spreading progress into the different aspects of society, from the space industry and military to the ordinary devices for domestic use. In the end of the decade, the *Formative Period* of discrete simulation was over (Nance, 1993), but the field of *Operations Management* continues to evolve with the computer, already facing matters like *Decision Analysis*, *Theory of Scheduling*, and even *Advertising*. The first ARPANET/INTERNET site was also set in 1969 in the University of California, Los Angeles (UCLA). Nevertheless, the people's greatest fascination stayed the Moon, and in 1969 Von Braun could finally land the first human astronauts there. Figure 1.4 shows the computer which was responsible for steering *Apollo 11* to the lunar surface.

Fig. 1.4 The Apollo Guidance Computer (1968)

1.2.3 The 1970s

In the book "*Distribution Management: mathematical modelling and practical analysis*" from Eilon, Wantson-Gandy & Christofides (1971) it seems evident that the dominating issues in the beginning of 1970s on the "Physical Distribution" (which slowly was being called just "Distribution"), were still the problem of depot location, linked to strategic planning, which first contribution

have probably been from Weber (1909), as the authors notice, and the problem of fleet management, more focused on tactics and on the *travelling salesman problem*, a classic of *Operations Research* (Flood, 1956). Numerical analysis was normally chosen to board these matters with the help of the computer, and by considering various static cost models as the base to calculate the "best solution".

Simulation was entering the period that Nance (1993) classifies as the *Expansional Period* (1971-1978), and the basic paradigms meanwhile developed started to spread in several different ways, naturally leading to an explosion of simulation languages and simulation tools, like in a tree. From now on, it starts to be less easy to follow the timeline of all the descendents from the basic paradigms, as they fast reproduce to a wide range of simulation applications. In the middle of the decade, for instance, in his book "*Systems Simulation, the art and the science*", Shannon (1975) was already indicating as the most important simulation languages the following: CSL, ESP, FORSIM-IV, using activities; SIMSCRIPT, GASP II, SIMCOM, on the event approach; and SIMULA, OPS, SOL, GPSS and BOSS for those who preferred to work with processes. Of course, he forgot the SLAM, another event approach language of the time directed at analogue modelling, but from such a list of languages it is not difficult to induce that there were not any special novel trends in the field during those times. Instead, the tendency seemed to be the upgrading and refinement of the old concepts in order to adapt them to a wider range of subjects and situations, thus leading to more perfection and an increase in the potentials of each paradigm. *Visual and Interactive Simulation* (VIS) was also giving the first steps with Hurrion (1976), and with the first VIS simulator SEE-WHY in 1979 (Fiddy et al., 1981).

One could say that what was happening in simulation was, in a certain way, happening in other fields of knowledge. The pioneering spirit of the previous two golden decades was entering a kind of relaxation, and seemed to be replaced by the main idea of upgrading and refining the technology meanwhile achieved. Hopp & Spearman (2001) argue that this could also be related to the view of the *Professional Manager* in America, that since the sixties had engaged first in a *Marketing* outlook and then in a *Finance* outlook, while devoting less attention to the operational levels of industry. Unlike Henry Ford and his contemporaries, who knew everything about the processes running in their factories, modern managers started to get apart from those processes and have entered too deep the abstraction. One of the results would be a natural relaxation in the industry's ability to face competition, for example.

And at the same time, more and more people were rallying in the streets against the Vietnam War, confronting the politicians with the "*Peace & Love*" trend which was spreading and acquiring an unexpected authority as a new political intervenient. POP was giving cards. And two years after the beginning of this decade, the Vietnam War was finally finished, mainly due to the power of American and European public opinions.

People would now engage in a series of new expressions against war and mostly

against nuclear energy. They seemed much more interested in enjoying peace and comfort and diversion than ever. They would buy washing-machines, hair-dryers, TVs, and any kind of electronic devices that technology was already able to produce, as well as flowered dresses and psychedelic cloths, books and music. *Progressive Rock*, for instance, a new intellectual expression in electronic music aligned with these ideals, fast conquering numerous adepts worldwide, with PINK FLOYD, VANDERGRAAF and GENESIS, among others, was pushing millions of people to buy their works recorded in vinyl discs. But to listen to these discs an electronic sound system was needed, thus also home sound systems were selling at a rise. Computers were still in the domain of universities and some powerful institutions, but *Management* could now start to think on focusing better the attention on the "Physical Distribution" of these products.

In the year of 1973, Dennis M. Ritchie had written the *C* programming language for the UNIX operating system, and with the improvement of precision in manufacturing techniques *Microelectronics* was getting able to compact electronic components to the order of including in a single package (*chip*) all the fundamental elements of Von Neumann's architecture: the *Microprocessor* was born, and with it the microcomputer.

Actually, the young *Internet* (ARPANET) soon would be in expansion for connecting scientific groups and universities for free sharing of email and files, while *XEROX* was designing its "XEROX *Alto*"[13] (Fig. 1.5). This was the first workstation using a built-in mouse for input, and already storing several files in windows with menus and icons, and also being able to link to a local area network (C.H.M., 2004).

Fig. 1.5 XEROX Alto, the workstation with mouse (1974)

What in the minicomputer was occupying a normal plaque of electronic components could now be reduced to the dimensions of a single *chip*, and being faster and less power consuming.

But in 1974 the sudden Arab Oil Embargo started to shake the industry, and the economy of America entered in recession. The barrel of crud was jumping from $3 to $12 (Williams, 2003). On the other hand, Germany and Japan had already completely recovered from WORLD WAR II, and their industries gave signals of more competition than what was expected by analysts. In particular Japan, implementing the ideas of statistician William Edwards Deming (1900-

[13] Although XEROX never sold the Alto commercially, it gave a number of them to universities. Engineers later included its features into other computers (C.H.M., 2004).

1993), previously ignored in his own country, was showing an ability to manage production and distribution with more superior performances than ever. The ability of the Japanese to fast absorb concepts and quickly turn them into high-quality new products was shifting the production of stereo-radios, TVs, microwave ovens and other appliances to Japan, as Goldratt & Fox (1986) seriously remind in their book "*The Race*".

Maybe already previewing a decline in the American industry, and thus the need for better control manufacturing processes, Orlicky (1975) developed the first closed loop *Material Requirements Planning* (MRP)[14]. This is a technique for dependent demand which decomposes any multipart product in the tree of its elemental parts and, from that, schedules production and establishes the new orders to the suppliers (Heizer & Render, 2001). It is a *push* system, since elemental parts are produced based on their independent demand and *pushed* in the direction of the assembling line.

Energy got more expensive, processes started to get leaner, with quality being the major issue since "everything" could be compared with the Japanese products. The 10% level of defects accepted till the end of 1960s no longer would be a measure of quality. Now, the target had to be much less than that, as Goldratt & Fox (1986) observe.

Nevertheless, Mankind was at the doors of a new burst of growth on its way to maturity, and all these "problems" would transform into something positive. Since then, we would enter a world of diverse approaches, and knowledge and technology would start to spread to other parts of the globe, as it had to be. In the car industry, for instance, France, Britain, Germany and Japan were again important players, with Japan side by side with America (A.A.C.A., 2004). At the same time, China, in the end of 1975, was successfully launching its first recoverable satellite, becoming the third country with the ability to launch this kind of satellites, after America and the former USSR (C.I.I.C., 2004). Bill Gates & Paul Allen found Microsoft precisely in this year, and started to try spreading BASIC programming language. The magazine "*Computers and Operations Research*" was established. Researchers gave attention to "*Queueing Systems Theory*" (Kleinrock, 1975), and in the field of simulation Zeigler (1976) proposed a formalism based on the *event-scheduling* approach for the design of hierarchical and modular models, named *Discrete Event System Specification* (DEVS). Meanwhile, the flexible disk (diskette) appeared, and, in the year of 1977, starting realize the dreams of many researchers worldwide, several personal computers are released in the market. The COMMODORE PET, shown in figure 1.6, was the first of them.

Fig. 1.6 COMMODORE, first personal computer (1977)

[14] It seems a first MRP approach has been tested in the sixties, but with no success, and was abandoned (Goldratt & Fox, 1986).

This machine was already sold fully assembled and straightforward to operate, with either 4 or 8 KByte of memory, two built-in cassette drives and a membrane keyboard (C.H.M., 2004).

Apple Computer would also introduce the APPLE II, and *Tandy Radio Shack* its TRS-80 desktop computer. Very conveniently, David Bunnell publishes "*Personal Computing*", and a year later around half a million computers exist in the USA, even if there is not yet a unifying operating system, a problem that soon would be solved by Mr Bill Gates. *Microprocessors* gain power by increasing operating speed and memory available, while using minicomputers some devote to research on *distributed* simulation (Chandy & Misra, 1979). Minds seem to open worldwide, the last fascist regimens of Portugal and Spain disappear from Europe; yet, the *Cold War* continues, as well as the *Space Race*, in a permanent measure of forces between capitalist America and communist USSR.

1.2.4 The 1980s

If in 1974 the barrel of crud was jumping from $3 to $12, in the beginning of this decade an even more dramatic increase in prices was observed due to the *Iranian Revolution* followed by the *Iraq-Iran* war. World crud prices would reach more than $50 per barrel and would stay above $20 during all the period of 1979-1986 (Williams, 2003). Coincidentally, this is the phase described by Nance (1993) as the *Period of Consolidation and Regeneration* concerning simulation. Robinson (2004), on the other hand, prefers to call this decade the period of *Revolution* in simulation. Nevertheless, the fact is that the next advances in the field would be driven by three important factors: (1) the *graphics microprocessor* developed at the end of last decade, (2) the release of *IBM Personal Computer* (PC) in 1981, and (3) the development of *Visual and Interactive Software* (VIS). The first made available a powerful technology; the second would make the computer accessible to any company, since the price of such machines would get near the 4,000€ (of nowadays money); and the third would make much easier than before the human interface with this machine. The evolution to follow would naturally result from this. Programmers got focused on developing applications of every sort, mostly to run on the PC's operating system, the MS-DOS created by *Microsoft*. Spreadsheet LOTUS-123 emerges, as well as word processors like *WordStar*, in the attempt to turn the computer even more popular. Integrated office software would be soon available, somehow imitating what big companies were doing with their UNIX and VMS mainframe systems, like DIGITAL, for instance. And when in 1984 *Apple Computer* launches the "Macintosh", whose screen is represented in figure 1.7, people were definitely fascinated with its graphic user interface.

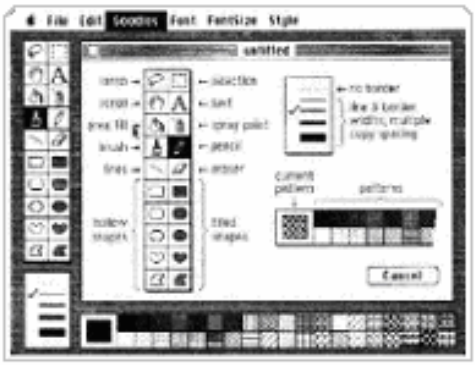

Fig. 1.7 The screen of first "Macintosh", from (1984)

It is important to notice that in this decade the microcomputer is passing from the level of expert to the level of ordinary user, hence the importance not only of developing programs of general interest but also of searching for an easier and more intuitive interface. The simulation tools were also following this trend.

Besides, in parallel with microcomputer popularization, hard-line computers running powerful operating systems like the UNIX or the VMS would stay, and a lot of work on simulation would be done on them, mainly when related to big systems from the fields of aeronautics, military or space research. But also parallel supercomputers[15] were being used. For instance, a CRAY[16] X-MP running CX-OS, the first UNIX-like operating system for supercomputers was delivered to NASA Ames in 1984 (Wilson, 1994). As we will see later on, the *United States Department of Defence* (DoD), where supercomputers run complex war games, is still nowadays one of the most involved institutions on *parallel* and *distributed* simulation. Most of the attention, however, was in that time turned to the popular microcomputer, which day-by-day conquers more fans.

With industry facing so many challenges, mainly demand increase and variability, quality (less than 1% of defects was now acceptable) and the Japanese competition (meanwhile Japan is number one in the automotive industry), managers realized that manufacturing processes had to be more efficient and leaner[17]. They answered with upgrading the old MRP system to the new *Manufacturing Resource Planning* (MRP II).

Unlike the first MRP, which basically ensured sufficient material was available when needed in the next workplace, the new MRP II would include, in the same computer system, information from other departments concerning demand, supply, statistical analysis, forecasting, and so on, what in fact was turning it more powerful than its predecessor.

The other answer was the change to the *Flexible Manufacturing Philosophy*, which in 1986 was defined by the *Institution of Production Engineers*, England, as: "*The proven response of a total facility which can serve a volatile market with minimum response time from order input to saleable product using the minimum of working capital*" (Carrie, 1988). In practical terms, however, as noticed by Carrie (1988), many experts believe that any system which seeks to achieve certain objectives by a variety of means is a *Flexible Manufacturing System* (FMS).

At the same time, managers also began talking of *Just-In-Time* (JIT), the Japanese *pull* philosophy practiced by *Toyota* at least since the previous decade (Ohno, 1978). Thus, *zero defects* started to be the new minimum acceptable level for quality, since the Japanese were meanwhile measuring the quality of their products in parts per million scrap, as Goldratt & Fox (1986) notice in "*The Race*", a book published two

[15] Supercomputers are usually based on multi-processor parallel digital machines, while common computers use the single processor architecture.

[16] The fastest computer of that time; its speed was in part due to its "C" shape, that was reducing the time of propagation of the signals.

[17] Actually, Heizer & Render (2001) define the period of 1980-1995 as the *Lean Production Era*.

years before Taiichi Ohno made available in English the first secrets of his "*Toyota Production System*" (Ohno, 1988), where the *Kanban*[18] plays the major role.

Computer-Aided Design (CAD), a kind of graphical interaction with the computer by which one could draw the parts on the computer's screen and then send that information to numerical machines, was also being used to try improving manufacturing processes; as well as other new concepts, like *Electronic Data Interchange* (EDI), an exchange of electronic documents among partners using message standards between computers to speed bureaucratic processes; or even *Total Quality Management* (TQM), a strategy of management seeking to achieve the quality in all organizational processes in order to improve production and reduce waste, supported by statistical methods.

Japanese JIT, however, in its innocence, was the main responsible for all this "occidental" panic, once it seemed in itself a miracle for achieving total quality. Thus, one could ask, why did the "occident" not apply these Japanese techniques? But apart from the fact this seem an intelligent question, an effective answer has never been given, probably because it lives in the domain of the proud and absurd. The most one can find in literature about this is the argument of *different cultures…*

Two good books from these times are "*The Race*" (Goldratt & Fox, 1986), focused on manufacturing, and where an interesting resentment seems directed to the Japanese success of that time, and "*Computer Simulation in Management Science*" (Pidd, 1984), a more elegant and neutral text about discrete simulation.

As other consequences from this decade, TCP/IP protocol for network communication was inducing countries worldwide to join the *Internet*, as well as some companies like DIGITAL and *Hewlett-Packard*. Object oriented C++ programming language was born at *Bell Labs* due to Bjarne Stroustrup, in his attempt to write discrete simulations like in SIMULA while taking advantage from the superior speed of C language. A formal method for representing the dynamics of discrete systems was introduced by Schruben (1983), based on event diagrams that became known as *event graphs*. The CDROM was born. The 32 Bit *microprocessor* appeared, at the end of the decade. Managers talk of *Empowerment*, an approach to stimulate the self-help efforts of the poor, instead of providing them with social benefits, or, from the point of view of management, to give workers greater ruling and resources for better serving customers and the interests of the business (Wikipedia, 2004b). America had lost the leadership of *microchips*. Civil logistics got practiced, mostly focused on *fleet management* and *warehousing*, as fleets get larger and their action spread, many times developing into international companies, like the *Haeger & Schmidt GmbH*[19], the *Mercantile*[20] and others. The speed of materials flow was increasing, while *lead-times* and product *life-cycles* were being dramatically reduced. The term "Logistics" in published literature have surpassed the term "Distribution", see list of references in (SOLE, 1999) and (Guedes, 1994), since the tendency was now the integration of the entire Logistics network (Gattorna, 1983), counting with

[18] The *Kanban* is a signal to authorize the production of next bin of material, and it is often associated with a card.

[19] Former ABX-LOGISTICS, (ABX-Logistics, 2004).

[20] Former *Maersk-Logistics*, (Maersk-Logistics, 2004).

supply, production, warehousing and delivery, as represented in figure 1.8.

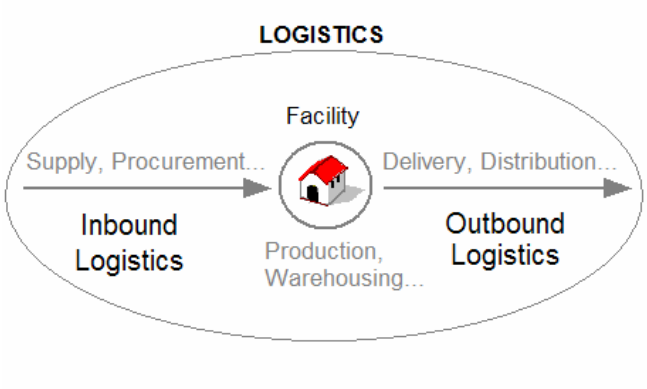

Fig. 1.8 The idea of "Logistics" as an integration function

In the field of simulation, packages like WITNESS, HOCUS, GENETIK, SIMAN/CINEMA and *ProModel* were already available by the end of the decade to those who wanted to work with *Visual and Interactive Software* (VIS), most of them *data-driven*[21]. Simulation was already perceived as an interesting decision-aiding tool, principally in manufacturing, even if its adoption was not yet widespread, as Robinson (2004) points out. But also the well known SIMFACTORY (CACI, 1986) was available, as well as other simulation tools like ECSL, for activities, EXPRESS, SIMON, GASP, SLAM II, SIMAN, SIMSCRIPT II.5, for events, and GPSS/H, SIMULA-P and again SIMAN, for processes, as referred by Carrie (1988). It was, in fact, an explosion of simulation applications, as it can be noticed by inspecting figure 1.9, from the pre-written simulation libraries like GASP or SIMON, to the most fascinating VIS packages of WITNESS or SIMAN/CINEMA.

As a curiosity, in figure 1.9 is represented the amount of different simulation tools directly referenced in the Ph.D. thesis of Guedes (1994), plus the book "*Simulation of Manufacturing Systems*" (Carrie, 1988), and the online bibliography of *The International Society of Logistics* (SOLE, 1999). Obviously, it shows an explosion of tools for simulation during the 1980s.

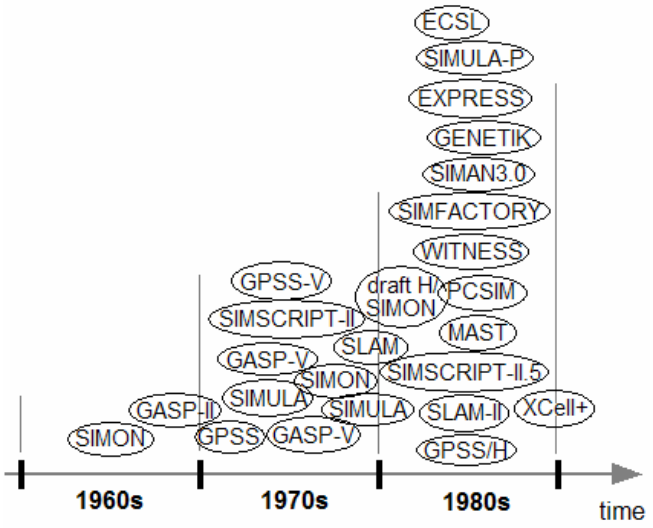

Fig. 1.9 Simulation tools cited in articles, 1960-90

Regarding the rest of the world, the price of crud have finally returned to near $20 the barrel, *Benetton* challenging marketing campaigns was becoming known worldwide, the first ZARA store outside Spain opens in Porto (Portugal) (Dutta, 2002), and the relations between Occidental and Eastern blocks were getting less tense, thanks to the fatigue of 45 years of *Cold War* and to the moderate approach followed by Mikhail Gorbachev in the USSR. People were bored of constantly living under the fear of a huge nuclear confrontation, with 20,000 nuclear missiles at each side pointing towards the other side, but soon two new events would change such an atmosphere: the fall of the

[21] *Data-driven* tools are those who freed the user from the need of programming to build the model.

Berlin Wall (8-Nov-1989), which finally signals the end of *Cold War*, and the invention of the *Worldwide Web* (WWW), in CERN[22], Switzerland, by the physicist Tim Berners-Lee (Berners-Lee, 1989), finally signalling the beginning of the *Information Society*.

1.2.5 The 1990s

The nineties begun with all the condiments pointing towards a volatile near future. On one hand, the tensions from *Cold War* suddenly vanished, leaving people with the feeling of confidence about the days to come again, but on the other a political instability in the East that would soon lead to the Soviet Union collapse and the emergence of several new countries. The social and economical recovery of reunited Germany began, while too many things were changing at the same time.

Although America was still leading the *ration of exports*[23] in the aircraft industry (3x), Japan was already leading in computer and business machines (7x), communications equipment (8x), scientific instruments (6x), and electricity transmission equipment (4x), while Germany was leading in the industry of drugs (2x), as noticed by (Chryssolouris, 1992). And people were now much more comfortable to sample the products and services that meanwhile invaded the markets, mainly electronic news. The pressure of a global business concurrence was increasing, and soon PC *compatible* computers were coming into sight from Asiatic competitors based in Taiwan and South Korea, demolishing the hegemony of IBM in the field.

With enterprises naturally enlarging their domains towards global levels, frequently with joint-ventures or acquisitions of other companies in order to remain competitive, the complexity of the network connecting all these organizations increased, and soon researchers understood it was necessary to focus the attention on those systems as a whole, thus introducing for the first time the subject of *Supply Chain Management* (SCM), foreseen as an integrated Logistics management, see Ellram (1991), Curran (1991), Towill (1992) and Christopher (1992). As Martin Christopher at that time stated, "*The Supply Chain is the network of organizations that are involved… in the different processes and activities that produce value in the form of products and services in the hands of ultimate customers…*", and adding, "*Supply Chain Management is not the same as vertical integration. Vertical integration normally implies ownership of upstream suppliers and downstream customers*" (Christopher, 1992). Already with a good usage of EDI, the recent developments in communications were in fact opening the doors of the organizations to the idea of SCM.

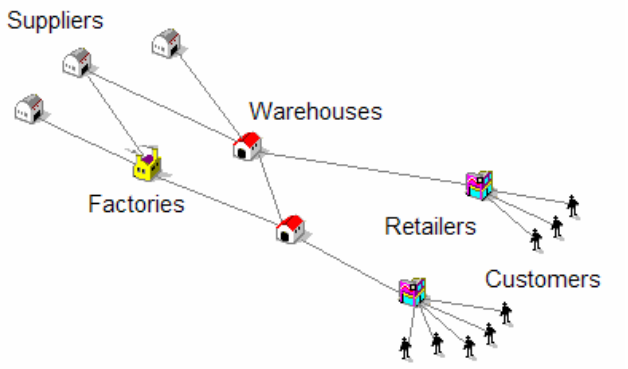

Fig. 1.10 An illustration of a Supply Chain (network)

[22] *European Laboratory for Particle Physics.*
[23] (Exports/Imports). Notice the values inside brackets are approximated.

Expert enterprises on Logistics, known as *Third Party Logistics* (3PL), would now have to face the evolving of SCM. Nevertheless, it seems the term SCM was not promptly adopted by managers and logisticians, and, for example, Sussams (1992), in his book "*Logistics Modelling*", does not yet mention such a term, even recognizing the need of integrated Logistics. On the other hand, Guedes (1994), in its Ph.D. thesis on Logistics simulation, at Cranfield University (U.K.), the single reference to SCM was linked to Christopher (1992). Anyhow, the tendency was about to invert.

On the other side, WINDOWS 3.0, the first graphical *Microsoft* multi-windows system, released in 1990, was turning the PC more user-friendly and attractive than before, pushing millions of minds to think on acquiring one. Prices of microcomputers were falling, and of course the number of programmers was increasing at the same rate. The *Internet* was also entering the domain of ordinary enterprises, and the *Worldwide Web* (WWW) was step by step connecting more institutions and computers worldwide, for the first time giving the chance of sharing information in a very attractive and graphical quality, which soon would be used for anything from advertising and email to file-transfer or online instant messaging (*chat*). The world was getting denser, and seriously keen on information. A short war (the *UN-Iraq* conflict) became an impressive televised show, not only of the military power of United States forces but also of its ability on Logistics, by which thousands of soldiers, guns, boats, airplanes and land vehicles could be send to the Persian Golf in a very short time, in front of millions seated at home in front of their TVs.

One year later, the *European Union* (EU) was birthed by the Treaty of Maastricht (1992), succeeding the *European Economical Community* (EEC), and people and goods started to move around in Europe without the need of passport or customs checks at most of the internal borders (E.U., 2004). This also accelerated the establishment of Pan-European enterprises, and again the need for improving the Logistics approach.

Simulation, once again, was following the trends of the systems, and taking the most possible advantage from the evolution of computers and networks. In these times of the beginning of the decade, Law & Kelton (1991) were presenting as basic models the single-server queuing system and also the inventory system, which they modelled with general-purpose languages like FORTRAN, PASCAL and C; but as well they were already mentioning the interesting advent of *distributed* simulation, an active area of research with pioneers like Chandrasekaran & Sheppard (1987) and Chandy & Misra (1979), in which several computers were linked in a network to run parallel individual tasks, therefore reducing the overall time of the simulation process. Actually, even *Artificial Intelligence* was slowly entering the field of Simulation, by introducing the concept of Distributed Simulation *Agents*[24] (Graham & Wavish, 1991).

Using a completely different approach, Towill (1992) is studying certain effects on

[24] An *Agent* is a component designed for distributed systems which include not only attributes and methods, as the classical *class*, but also a certain ability to learn and make decisions on its own. It is able to communicate with other *Agents* asynchronously. Nwana (1996) is considered the "Bible" for the *Agents*.

the Supply Chain behaviour using *System Dynamics Simulation*, an approach directed to continuous mathematical modelling based on difference equations and causal loop diagrams (Forrester, 1960). Brito (1992), on the other hand, using *event-scheduling*, explores CAD techniques for configuring VIS warehouse dynamic models, a new approach for warehouse design, while Sussams (1992) makes use of static cost functions for modelling processes associated with transportation, warehousing and other Logistics issues. Later, Guedes (1994) would also use static functions to model facilities and transport costs in the Supply Chain, developing a VIS simulator for *Logistics Strategy Planning* (LSP) centred in the facility location problem.

In the field of *Petri nets*, Jensen (1992) was publishing "*Coloured Petri nets. Basic Concepts, Analysis Methods and Practical Use*", and Silva (1993) his "*Practice of Petri nets in Manufacturing*", between others.

Nevertheless, it was with Michael Pidd that an attractive classification of tools for discrete simulation had been proposed, with the seven different classes presented in figure 1.11. In this figure, these classes have been organized in the form of a graph, trying ordering them by the *versatility* and the *simplicity of usage*.

It is important to notice that Pidd makes a clear distinction between VIS systems and *Visual Interactive Modelling* (VIM) systems, since the first run the simulation as an operational game, watching and interacting, while the second uses a *Graphics User Interface* to build the model as well as for visual interaction (Pidd, 1992).

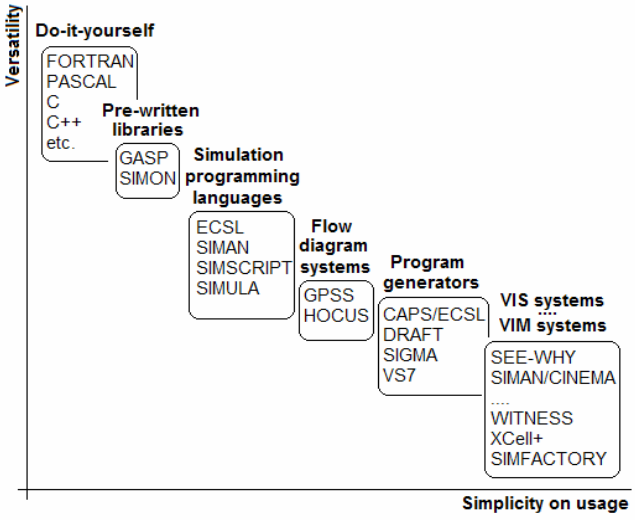

Fig. 1.11 Classification of simulation tools, based on Pidd's (1992) proposal, and irrespective *versatility* and *simplicity on usage*

Obviously, VIM is more recent than VIS, and naturally follows the trend of software applications designed to run on modern multi-windows operating systems. Anyhow, an example of a VIS system is shown in figure 1.12, from a simple model made with SIMPROCESS, the *process-oriented* support of SIMFACTORY.

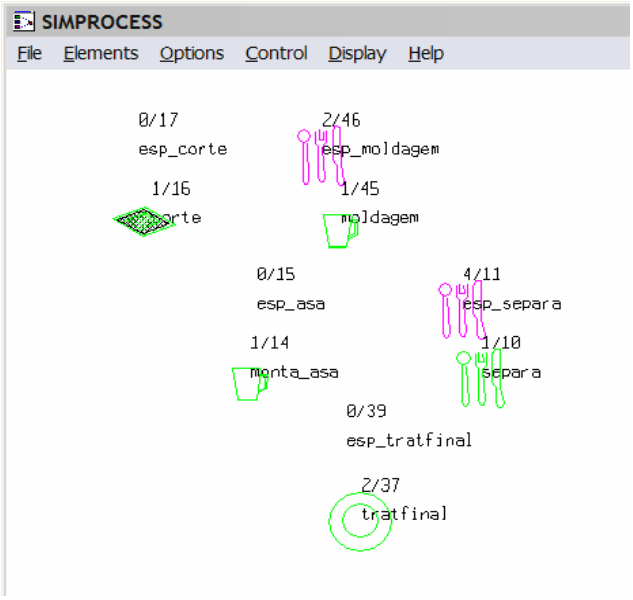

Fig. 1.12 Modelling a simple manufacturing process with SIMFACTORY II.5, a VIS system

In the middle of the decade, America had recovered from the crises and, although not dominant, it was again seen as the stronger and the most competitive economy of the "global" world. Nevertheless, the invasion of global markets by cheap PC *compatible* computers produced in Asia and the release of *Microsoft* WINDOWS 95 were the two main facts that would provoke another burst in all fields of knowledge. Everybody would benefit from this. In Science, Engineering, Medicine, Business, Sociology, Architecture, Arts, etc., the boom comes naturally from the dramatic fall of the computer into the hands of common institutions and citizens. Such a powerful machine could now be useful for almost anything, from the integrated-office software commodities to the design of new products using CAD, as well as to access, share and present high graphics information when connected to the "global world" of *Internet*[25]. For the first time, each user had the world at his fingertips. Future benefits or ruins would depend on how users would use their fingers. It was not as much a technologic advance than a kind of social distribution of technology. Coming from the top, the computer had touched the ground and reached the public domain.

The human power of computation is thus spreading to a larger extent of brains, precisely at the same time communications are getting global. The world turns more compact, and time compresses. *Internet* cafés open in Tokyo, London, Berlin, Paris, New York, etc., and online providers like AOL, *CompuServe* and MSN, give the user the possibility of accessing a wide range of "virtual" services, like news, forums, encyclopaedias, stock markets, flight and train timetables, and much more. Virtual realities like the *Worlds Away*, a graphics virtual world created by *Fujitsu* where people assume a figure built by their own and take for granted certain unrealistic powers, capture the interests of many in the community, as do virtual games. *Information Society* was expanding as a boom, due to the recent advances in the field of *Information Technology* (IT).

Regarding *Management*, the concept of *Enterprise Resource Planning* (ERP) is starting to replace MRP II, as the vision concerning manufacturing processes also gets wider and extends to the entire enterprise process. The new ERP will append the MRP II with a sophisticated set of software tools specially designed for each enterprise department, where simulation as well as optimization can be incorporated, and finally integrated by the main system's application. SAP[26] is one of the most popular distributed systems of this kind (see also EPICOR[27], *PeopleSoft*[28] and others[29]). The enterprise maintains the original *push* philosophy of MRP II, contrasting with JIT, but it is now supported by a sophisticated software environment for decision and planning.

But not only the manufactory face was to change thanks to the new advances, also the relations with *last customer*s, which could now be measured in millions, connected to each other by the *Internet* and asking more and more products with their own specific tastes. The *Mass*

[25] Apart from the fact *Internet* is just the support for the WWW, the two concepts have merged in usual language.

[26] http://www.sap.com:80/index.aspx
[27] http://www.epicor.com/www/
[28] http://www.peoplesoft.com
[29] http://www.2020software.com/

Production Era was giving the place to the *Mass Customization Era* (Heizer & Render, 2001), and *customers* become the "precious thing". At the same time, *e-Commerce* start emerging, with some of the important companies already presenting their products by *Internet* to the public, in a kind of a digital catalogue from which customers can directly order their products, and in certain cases even customize them in accordance to their tastes.

Besides, with the international tariffs reduction due to the endorsement of *World Trade Organization* (WTO), the signatory countries borders become more permeable to foreign companies, and once again the global expansion is stimulated. Managers talk of *Learning Organization* as the idea of easing and promoting the flow of relevant information between partners about their particular experiences. Enterprises try to meet the *International Quality Standard* ISO 9000[30], while the idea of agility reaches the domains of Manufacturing and the Supply Chain. *Agile-Manufacturing*, as explained by Hopp & Spearman (2001), is the practice of using equipment and layouts in a way to enable accommodation of new products, change in volume and shifts on products mix. *Agility* in the *Supply Chain* refers to the ability of the Supply Chain to thrive in turbulent market conditions, as stated by Harrison et al. (1999); see also Goldman et al. (1995), who give special attention to *customer value* and *cooperation* between partners in the Supply Chain[31].

By the end of the decade, and apart from some memories of a fast NATO intervention in the Balkans in 1995, with the United States leading the mission and bombing Belgrade without the permission of the *Security Council* of the United Nations, the world seemed relatively calm on its growing consumption of products. For some months, China had frozen WTO talks due to the mistakenly bombing of the Chinese embassy in Belgrade (CNN, 2001), but globalization was spreading and the global network of international enterprises increasing in complexity, and even China was giving obvious signs of the intention to open its gigantic market. Common people in many occidental countries could already afford not only colour TVs, high-fidelity sound systems, mobile phones, and all kinds of appliances, but also personal computers equipped with digital sound blasters and prepared to run high graphics video games, as well as with telephonic connections to the *Internet*, and many other "interesting" devices seen by everyone as authentic pleasures. Mainly due to the exponential increase of concurrence, new products, or at least new versions of the products, are released into the markets in cycles of just a few months, while demand figures get extremely volatile. Such a high level of consumption, on the other hand, seemed to make people more and more anxious for consuming, and if in the late 1960s a growing number of persons was complaining about the demise of family, now a growing number of persons started to complain about the general demise of "values", being

[30] A standard created in the later 1980s by the *International Standards Organization* (ISO), accepted by many countries and criticized by some managers for being little practical.

[31] As the Supply Chain is a network usually belonging to more than one company, ultimate customer's satisfaction naturally becomes a measure of performance easily accepted by partners. At the same time, the concept of *cooperation* results naturally from the composite "geometry" of the problem. *Cooperation* also brought to light the problem of *trust*, in terms of SCM.

personal or cultural, since everything seemed to start being reduced to products, economics, and ambiguous politics, as well as pointing out the dangers of "globalizing" such a simplistic theory of world[32].

In what concerns simulation, the usage of graphic compilers by most programming languages and the popularization of *Object Oriented Programming* (OOP), provoked a general trend of adapting simulation tools to these kinds of interface and paradigm, as it is the case of the VIM system shown in figure 1.13.

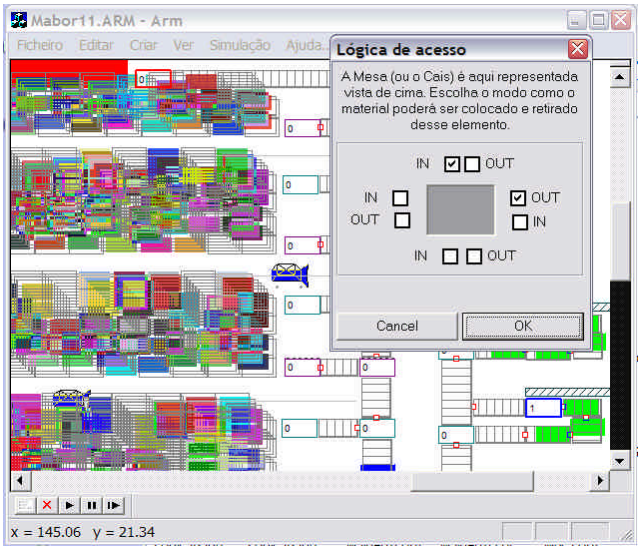

Fig. 1.13 VIM warehouse simulator, from Feliz-Teixeira & Brito (1999)

This trend led to the proliferation of a new brand of high graphics VIM systems, most of them offering a very comfortable menu-based user interface, and some even including a *three dimensional* (3D) view on the model and on the simulation. As mentioned by Robinson (2004), commercial VIM tools like *ProModel*, ARENA, QUEST, *Taylor II*, *AutoMod*, *AweSim*, *Micro Saint* and *Enterprise Dynamics* had been presented along this decade at the "*Winter Simulation Conference, USA*", one of the major international events on simulation advances.

Anyway, also simulation languages and other simulation related tools have been developed, like MODSIM, for example, an *Object Oriented* simulation language in use since the beginning of the decade (Herring, 1990), or C++SIM, a simulation library with SIMULA-like routines, including random number generators, queuing algorithms and thread package interfaces (Little & McCue, 1994); or even *SimJava*, a simulation library written in *Java*, allowing graphic animations (Howell & McNab, 1998), and much more. A very useful online source of simulation tools can be visited at Rizzoli (2004).

Of course most of these tools can be used to build models of Logistics systems and Supply Chains, as most of them are general propose simulation applications, unlike the one shown in figure 1.13, which is strictly directed to warehouse modelling. Analysts at IBM *Research*, for instance, were using MODSIM language to model and analyze the European PC business due to the decline of IBM sales in Europe (OR/MS, 1996). The Supply Chain was represented by blocks based on objects and later made to run in SIMPROCESS. Although the alleged saving of $40 million per year does not contribute very positively to the simulation believers, since later IBM lost the PC market, the fact

[32] The *Anti-Globalization Movement* has a wide support around the world, mainly in central Europe and Canada, as well as in South Korea, Japan, Brazil, and recently in Africa and even in America. It is more against what is being globalized than against the natural process of globalization, and defends the respect for diversity of geographic regions and cultures, and the planning of the development in terms of their own natural resources and believes, instead of simply imposing the *neo-liberal* paradigm by installing there companies from rich countries, see (Wikipedia, 2004c).

is that this was a very interesting approach to a practical problem, and deserves to be studied, see Feigin et al. (1996).

On the other hand, also the technology of *Agents* was being used by Fox et al. (1993) in the development of an *Integrated Supply Chain Management System*, an architecture based on interactions between *Agents* to reengineer operations in the Supply Chain in order to make it react quickly to changes. Much of this belongs to the trend of what was starting to be called the "Virtual Enterprise", or the "Virtual Supply Chain" (Chandrashekar & Schary, 1999). *Agents* were in this case used for simulating "cooperation" and "negotiation" between certain activities in the Supply Chain. Despite the reflections of Günter Müller et al. (1999) about the risks of these kind of *Multi-Agent Systems* for Supply Chain coordination, many researchers were at the time focused on this approach, as in the case of Eymann et al. (1998) or Chen et al. (1999) or Bruzzone & Mosca (1999), and others.

Using other kind of approaches, Zeigler et al. (1999) were at the time proposing to adapt the previous IBM *Supply Chain Analyzer* to the more recent DEVS/CORBA distributed simulation environment, while Kemers & Merkuryev (1999) were using ARENA to simulate a case of *Logistics Strategy Planning* (LSP) for some IT distributors in the Baltic countries. Besides, *ATS Automation-France* was choosing *Tecnomatix â eM-Workplace* software to help design its systems. In figure 1.14 is shown a typical 3D view by this tool, which provides detailed layout design capabilities and materials-flow simulation in a single environment, enabling users to evaluate performance and logistics and the effects of any induced changes on materials flow (Tecnomatix, 1999).

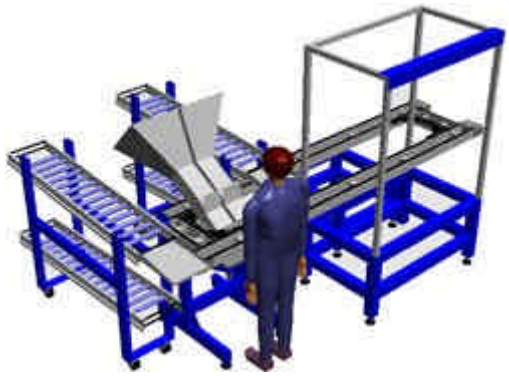

Fig. 1.14 *eM-Workplace* simulator, from Tecnomatix®

1.3 The present

The arrival at the 21st century is getting so complex in terms of diversity of trends, systems and believes that the use of classification becomes complicated. It is time, perhaps, to substitute the classical deterministic approach for a more holistic perspective on systems, as it happened in the 1930s in the field of Physics with the introduction of *Quantum Mechanics*[33].

What in terms of technology is driving this generalised complexity are the high-speed communications, using fast optical-fibres and satellites; the powerful digital microprocessors (in the early days of 2000 a microcomputer could already use a clock speed of around 500 MHz, 128 MByte of RAM and 2 GByte of hard-disk-storage); the globalization of information, with the massive spread of WWW and *Internet* (8.2 billion emails were sent worldwide in 2000,

[33] *Quantum Mechanics* is not interested in a deterministic knowledge of a certain process, but on the "modes" in which such process can be found, and in the probabilities of finding it in each of those "modes". The father of this approach is considered to be the German physicist Werner Karl Heisenberg, 1925.

and the WWW was estimated to surpass 1 billion indexable webpages); the advances in nanotechnology (IBM makes experiments with a microprocessor based on a molecule with five fluorine atoms); the advances in genetics (the human *genome* have been completely decoded); as well as the manned travel to Mars, somehow of controversial interest, at the moment. In addition to this, there is the general decrease of transport and communication costs.

1.3.1 Life and SC management

Defying such wish of expansion, however, resources of oil, steel, and some other crucial raw materials happen to be too few to satisfy the ever increasing demand, with the steel, for instance, not being enough in Earth to produce a car for us all and for each Chinese citizen. Besides, too much pollution in industrial districts and towns and an excessive level of carbon dioxide in the atmosphere are pointed out by scientists as responsible for the climate change and the rise of sea waters, as well as for certain diseases and the increasing power of storms, naturally leading to the spread of the idea of *Sustainable Development*, soon defined by the United Nations (UN) as the "*Development that meets the needs of the present without compromising the ability of future generations to meet their own needs*" (UN, 2004).

Unlike the 20th century, in which most processes were seen as being linear and without any substantial feedback on their surrounds, the 21st century promises to be subject to recognizing that processes are many times nonlinear and in certain cases with a subtle but strong feedback influence on the medium. And this seems to be a serious issue that affects most systems, mainly when the regimen of their utilization is made to approach limits.

Although for many people the need for facing this *feedback problem* represents a serious obstacle to development, already a substantial number of modern enterprises and institutions are treating the issue as part of their *Total Quality Management* (TQM) policy, moving towards the modern "green" state of mind.

Somehow following this same sort of logic, *Supply Chain Management* (SCM), for instance, struggling to handle so many kinds of products, resources, times and interests on the current turbulent markets, instead of the "old" *concurrence* between partners, is already recognizing the value of *cooperation* and *trust*, thus understanding that there will be a future reflex in its universe depending on the actions taken at the present. Concurrence, as it is argued, is now present between Supply Chains, instead of between partners. But the problem is not so simple anymore, since partners can also choose to belong to more than a single Supply Chain… Perhaps SCM is becoming more an art than a science, as Richard Saw[34] declared during one of my visits to Cranfield, obviously enlightening the high complexity of the processes involved in such discipline, but it also means we are reaching a wider view on systems behaviour.

A very simple strategy on facing such complex behaviour can be to decline understanding it and trust in systems to

[34] *Richard Saw* is at the moment the director of executive development of the *Centre for Logistics and Supply Chain Management (CLSCM)* at *University of Cranfield, UK. email:* r.j.saw@cranfield.ac.uk

sooner or later regulate themselves through each other, and so our job might only be to ensure the best flow of what flows between systems. In that sense, the present idea of *visibility* is seen as a means for achieving such transparency, thus leading to the best possible performances in practice. This, of course, is just an approach, a hypothesis which considers all systems intended for good, and therefore neglects the human tendency to pervert, as well as the amount of time wasted due to it, for example.

Nevertheless, in SCM this is believed to result also in a natural improvement on decisions and follows the trends of EDI, ERP, *cooperation*, *virtual enterprise*, etc., as well as the idea of pushing the demand information upstream on the supply network to allow reducing inventories and minimize the amplification known as *Bullwhip Effect* (Forrester, 1960). Of course, this is in the direction of JIT, and can well be seen as a good support for *e-Commerce*.

These trends are at the present defended at most schools of management, anyhow, in practice all these "theories" constantly face a serious challenge related precisely to perversion. Such challenge is called *trust*.

It seems the complexity of our days has brought us to a very interesting situation: either we become trustable to each other or we will have no chance to manage anything. In that sense, perhaps the answer must also pass by what Hopp & Spearman (2001, pp. 43) call the "*need of managers to embrace detailed technical knowledge and not only maintain a love affair with marketing and finance*".

Focusing a little more on manufacturing, today, even if the value of other approaches is still recognized, like the traditional *Make-To-Stock* (MTS), where parts are produced based on demand forecasts and the customer is served from the stock, or the expensive *Engineering-To-Order* (ETO), with emphasis on the design of a solution specifically tailored to a customer order, the preference is obviously directed to the approaches of *Make-To-Order* (MTO), which is very close to the JIT concept, as well as to the *Assemble-To-Order* (ATO), which is the most important representative of our *Mass Customization Era*. In reality, these approaches tend to locate the Supply Chain *decoupling-point*[35] at different places, as figure 1.15 suggests.

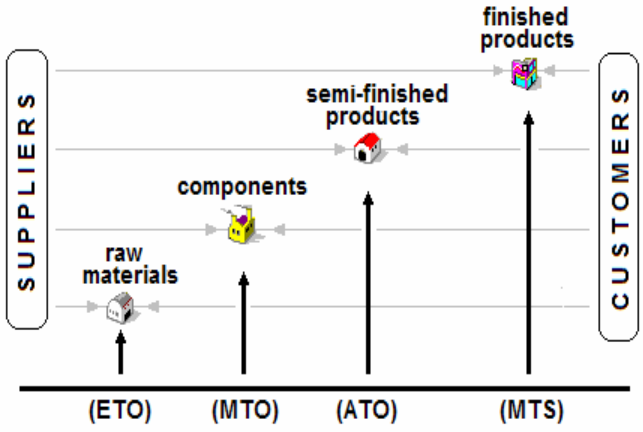

Fig. 1.15 Different positions of the demand decoupling-point, inspired on Dong's (2001) Ph.D. thesis

As Dong (2001) notices, the MTS locates this point at the finished-products seller (retailer), the ETO locates it at the raw-materials level, while both ATO and MTO locate it somewhere between those extremes. ATO is usually at the level of semi-finished products, while MTO is at the level of components. The same author

[35] "*The decoupling-point is a standard term given to the position in the materials pipeline where the product flow changes from PUSH to PULL*" (Mason-Jones & Towill, 1999).

associates with each of these approaches the following primary objectives: to MTS the *responsiveness*; to ATO the *customization*; and to MTO and ETO the *lean production*. An extensive work about the importance of the position of the *decoupling-point* in the Supply Chain has been made by Mason-Jones & Towill (1999), who also have studied the *Bullwhip Effect* on inventories by applying *System Dynamics* simulation to the *Beer Game*[36], see Mason-Jones et al. (1997).

But with the ever increasing demand of products in the global market, also other less common approaches are being used in the manufacturing/distribution processes, like the *merge-in-transit*, the *cross-docking*, or even other schemes nowadays practiced by DELL or UPS, for instance, in a kind of MTO in which the last manufacturing steps are completed while in movement to the customer. All this is in the trend of a flexible, synchronized and "demand driven" response to customer orders, and some researchers claim it as the seeds for the future "moving factories", of which the "factory ship" would be a particular case (Hewitt, 2001). The new concept of *Demand Pipeline Management* (DPM) is even being suggested by Hewitt (2001) to replace SCM, due to the continuous-like flow of products (as in a pipeline) observed in many distribution processes of our days, and more recently also defended by firms like the *MarketBridge Corp*. (Furey, 2003). A higher competence in manufacturing and in assembling techniques, the usage of digital mobile-phone communications to trace products on delivery, *code-bar* references, as well as the almost instantaneous online information shared by partners through WWW sites, place the present information potentials much above the simple EDI serial transmissions from the 1990s, thus allowing a better synchronization of all these new processes.

Cases deserving a special reference in the literature for adopting this JIT "style" are, among others, DELL *Computers*, *Harley-Davidson*, *Matsushita*, *Volkswagen*, and, as the *primus inter pares*, ZARA[37], which is almost considered in itself a paradigm. Assuming mostly a philosophy of *reacting* instead of *predicting*, these companies are made to operate with low inventories and high rotation of stocks, while generating high *Return On Investment*s (ROI)[38].

By studying the literature on the matter, one easily realizes that nowadays *operations* tend generally to be driven by the following management concerns:

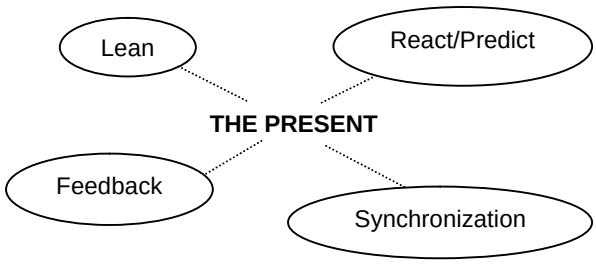

Fig. 1.16 Current concerns in *Operations Management*

Or, in summary, companies are thinking *lean* to minimize the waste on materials,

[36] The *Beer Game* was developed at the *Massachusetts Institute of Technology* (MIT) in the 1960s in the lessons of Sterman (1992), and it is a lineal chain including Retailer, Wholesaler, Distributor and Factory. The game is played by hand and between groups. For more information please see: http://web.mit.edu/jsterman/www/SDG/beergame.html

[37] Fashion clothes ZARA is a company from INDITEX Group, which has headquarters in the town of La Coruña, Spain. Apart from its success, ZARA is known for rejecting any kind of advertisement: http://www.zara.es

[38] ROI = (Revenue – Costs)/Assets

processes and organizational; choosing from *react* or *predict* to optimize inventories and operations costs, without affecting service levels; thinking synchronization to reduce the waste of time and resources while interfacing with each other and with the customer; and thinking feedback in order to minimize future impact on the medium and on the enterprise itself. This starts to seem more in harmony with Leonardo Da Vinci's thought "*everything connects to everything else*".

It was, however, in this kind of global business atmosphere that in 11SET2001 New York lost its most emblematic symbols of globalization, the "Twin Towers" of the *World Trade Centre*. The deadliest, the most implausible and destructive terror attack of history has brought down those giant skyscrapers by crashing into them two commercial flights. On the entire globe minds paused, confused, terrified and apprehensive while following the scenes on TV. Many analysts agreed that since that day the world would never be the same again.

At the present time, people and business around the world are still adapting to the generalized instability provoked by such event, which in the first approach have leaded to a significant disorder in many business areas, as well as to a global political insecurity due to war. First, it was the war of Afghanistan, to remove the Taliban regime from power, and about one year later the military invasion of Iraq by American and British forces, also to overthrow the defying regime of Saddam Hussein. This last action, however, was since the beginning classified as illegal by the *United Nations* (UN), and has been openly opposed by the world's public opinion as well as by states like France, Germany, Russia and China.

Thus, what in terms of economy, politics, social or general development will result from this in the years to come, one has to wait to see. It would be unwise to search for a trend while the system is still oscillating. Nevertheless, such instability was also a confirmation that the mechanics of a "globalized" world are much more complex than expected, since actions will more and more reflect on the entire globe. A problem in Asia can produce a storm in Beijing or New York, and the opposite is also true.

But life continued. Meanwhile, China was finally accepted as member of the *World Trade Organization* (WTO) at the end of 2001, and the new currency of *European Union* (EU), the EURO (€), was adopted in January 2002. Even under the uncertainties resulting from the Iraq war, climate change[39], as well as the new crisis of oil that brought the crud barrel higher than $65, global markets continue developing, mainly with the consolidation of the economies of *European Union* (EU), which meanwhile was enlarged to 25 members, and the fast economic growth of China (around 9% a year). At the same time, the mobility of people, merchandise and business inside the EU greatly increased, from the outset giving the effective idea of a "unified" and opened continent. In these times of the *Information Society*, easier contacts, more sources of

[39] In reality, some scientists use to claim nothing was happening deserving to be called "climate change", but the truth is that in many places the atmosphere have reached insupportable levels of degradation, and even the *Pentagon* is at the moment seriously concerned with the issue, see (Townsend & Harris, 2004) or (Stipp, 2005).

information, more variety of solutions and points of view, easier international travelling and less bureaucracy are leading the societies to a transparency which also enables students, teachers, managers and scientists to often take part in international projects for common interest.

But a worldwide sourcing of materials has also turned the procurement processes more complex, and managers many times face the need for simpler approaches to effectively act. Vossebeld (2002), for instance, notices that the worldwide concurrence of suppliers greatly increased the need for managing *inbound Logistics*, and, adding the difficulty of calculating total Supply Chain costs when transportation is frequently outsourced to 3PL groups, he argues it is important to simplify the SCM by using software based on a more holistic perspective. To this, one must add the *reverse Logistics* concerns, related to the recycling of unused stock and the traceability of products (Ritchie et al., 2000), as well as to the need of handling an increasing rate of product obsolescence (Blumberg, 1999) which can have serious environmental and economic implications, as it is the case of the end-of-life of computers (Knemeyer et al., 2002). A good text linking *reverse Logistics* with products *life-cycle*[40] can also be found in Tibben-Lembke (2002).

On concluding, it seems that the current trend of *transparency* reduces the friction between the elements participating in the processes, but at the same time reveals a much more complex world to be managed. *Flexibility*, *Agility* and *Pipeline Management* are concepts trying to respond to such a complexity as well as to the turbulence resulting from it. However, there is still a lack in metrics to effectively describe the Supply Chain performance, since different companies are usually involved in the same process of supply (Lambert & Pohlen, 2001).

1.3.2 IT and Simulation

With the popularization of powerful high-memory/high-speed microcomputers, which meanwhile also got extremely compact and started to incorporate direct interfaces to the *Internet*; as well as due to the wireless communications boom and the explosion of WWW oriented services and tools, *Information Technologies* (IT) have fast dominated our lives. A *light-and-portable* machine like, for instance, the one shown in figure 1.17, with 700 MHz of CPU speed, 256 MByte of RAM, and 20 GByte of hard-storage-disk, and access to *Internet* via LAN or telephone, is nowadays familiar even among students.

Fig. 1.17 Fujitsu-Siemens *LifeBook BSeries* (from 2002)

In truth, not only business is searching for agility. Also people do. And IT seems to deliver this "chance". Documents from the most important libraries in the world, public institutions, magazines, journals, etc., are now available "online", as well as a huge

[40] A *Life-Cycle Diagram* is commonly used to represent the potential sales in the product phases: *Introduction, Growth, Maturity* and *Decline*.

sea of "virtual" companies dedicated to every sort of business. In this "virtual" world, where the homologous of book became the *e-Book*, the commerce became the *e-Commerce*, and the business the *e-Business*, people are interfacing with one another and with institutions through email and online messaging, plus using WWW sites as outlets from where other more direct information can be obtained, including the download of files of all kinds, from plain text to high fidelity sound and digital video. Today, anyone can work on an isolated mountain and keep contact with the entire world, by means of a computer connected to a telephone line. Companies and institutions are also exploring all these opportunities.

In this atmosphere, simulation is once again following the trends of the epoch, but playing an increasingly significant role in it each time. In truth, people nowadays are building models of almost everything (yet not always with accuracy, as frequently they are built by the inexpert on modelling) as well as facing direct contact with them on the *Internet*, where *Agents* are specially busy trying to capture users profiles, as well as at bank cash machines (ATM), computer software or even cybernetic toys.

Also ERP, for instance, is supported by a collection of simulation packages tied to algorithms or heuristics of optimization which then output to other packages for enterprise management. But models are also used in projects and studies of bridges, vehicle tests, urban traffic flow, disease contamination, human behaviour and human performance, and several other matters, in a diversity of approaches that extends from the simple numerical analysis to complex methods like *Bond Graphs* (Paynter, 1961) or *Neural Networks* (McCulloch & Pitts, 1943). The modelling approach is usually chosen in accordance to the school of modelling, the ability of the analyst, and the type of system in study. In figure 1.18, the major topics at the EUROSIS[41] *European Simulation and Modelling Conference*, Naples 2003, are presented, which gives a reasonable idea of the diversity of article subjects offered in a single conference of simulation of nowadays.

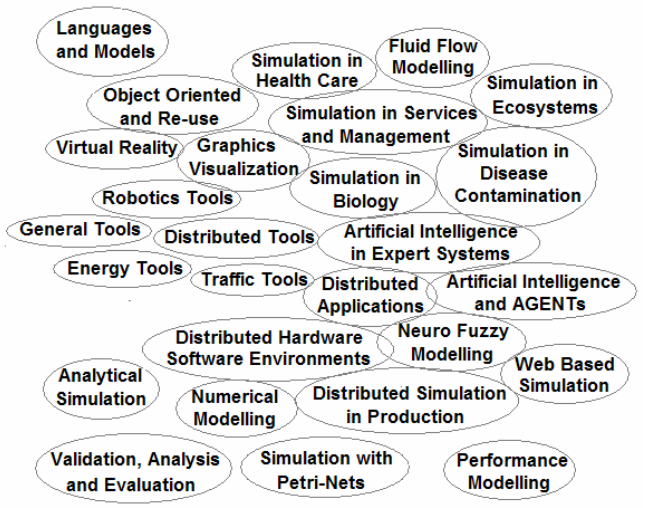

Fig. 1.18 Major topics at the EUROSIS *2003 European Simulation and Modelling Conference*, Naples, Italy

Visual and Interactive Modelling (VIM) is already an interface commonly used in many simulation tools, specially when they are directed to industry. Also, the *Object Oriented Paradigm* (OOP) became the most attractive programming approach, due to its modular structure and advantages of *hierarchy* and *inheritance*. At the same time, visual effects have also been brought to simulation in its most perfect technology, leading to the high quality and fast 3D

[41] The *European Simulation Society*, a division from the *European Technology Institute*. http://www.eurosis.org

graphics currently seen in virtual reality games and interfaces. This trend, however, even being very attractive and surprising to most people, can as well be considered an element with which often the real consistency of the model behind such excellent graphics is obscured. At the same time, since only in the end of a simulation process any serious *quantitative* information can be retrieved from the model, the present tendency is to look at 3D graphics simulations as excellent tools for marketing, easy communication between the modeller and its "clients" (Waller & Ladbrook, 2002), and, being the model consistent, to teach and train personnel and, probably, to gain some *qualitative* knowledge. Professional flight simulators, for instance, are perhaps the most notable machines based on virtual reality, at the moment.

Another bet of commercial simulators is interfacing with other applications as a way of better *integrating* the simulator in its environment of "work". This means offering interfaces with data-bases, CAD systems, packages for analysis or graphing as spreadsheets (usually the MS-EXCEL), etc., or even the ability of exchanging information with distributed applications, which can be spread through the *Web*[42]. XML data format is one of the most interesting contributions to this kind of interfacing. However, the consequence of all this is not necessarily more reliable simulations, but mainly more comfortable processes of simulating and, in principal, less time consuming on configuration and analysis. Yet, the simulation process lives in the core of the system, and strongly depends on the talent of the modeller.

Blocks and *add-ins* for optimization and decision automation are also sometimes appended to these integrated commercial simulation packages (in ERP systems, for example), simplifying obtaining filtered results from the simulation, instead of raw outputs many times difficult to interpret. But as those facilities already act like "filtering" the real results, they also impose on the analyst a certain mode of looking at the output data, that in certain cases can be damaging for a critical spirit of analysis and, at the same time, induce easier but erroneous conclusions. Hopp & Spearman (2001), for instance, upon referring to ERP software packages, defend that "*the rise of such monolithic software packages which purport to encapsulate "best practices" may prove to be a giant step backwards in terms of managers better understanding their practices*".

Many different kinds of algorithms and heuristics can be applied for this purpose, but, in general, as Robinson (2004) explains (please see figure 1.19), "*the problem is similar to that for any mathematical optimization problem. Given an objective function (expressed in terms of the simulation outputs), the aim is to find the optimum value of some decision variables (simulation model inputs), subject to a set of constraints (allowable range for the simulation model inputs). The difference from mathematical optimization is that no algorithms exist that guarantee an optimal solution will be found*".

[42] Nick name commonly given to the WWW.

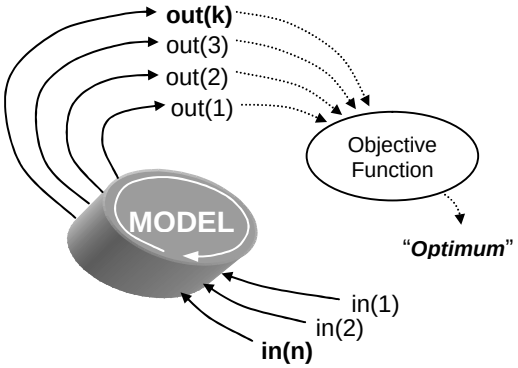

Fig. 1.19 A generalised simulation model followed by an optimization module, inspired by the comments of Robinson (2004)

Apart from this, powerful simulation packages are currently used in industry with very good performances, as is the case of *eM-Plant*, directed to manufacturing and business processes, which is used by BMW, *Boeing* or *Philips*, for instance (Fig. 1.20).

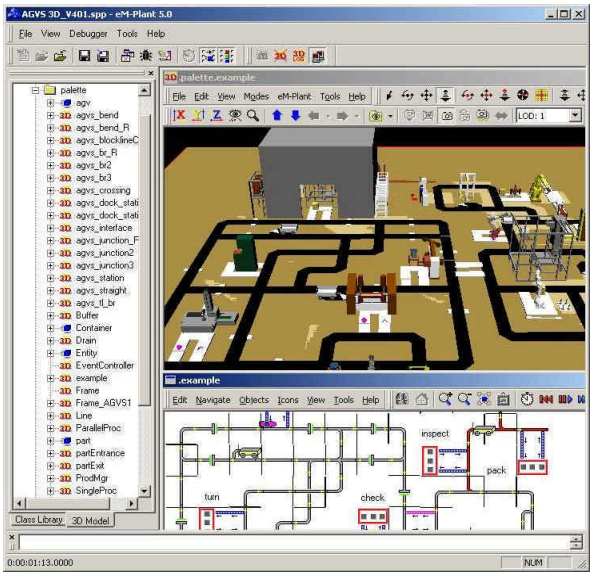

Fig. 1.20 A general view of *eM-Plant* simulator, from *Tecnomatix*® (http://www.emplant.de/simulation.html)

In this same level of integrated OOP technology, one can also refer to the WITNESS, used by *Virgin Atlantic* or *Volkswagen*; the *ShowFlow*, developed from *Taylor II* system; as well as *AutoMod*, *MicroSaint*, SIMUL8, or even *CPN/Tools*, among others.

All these tools are user-friendly OOP standing alone applications designed mainly to run in single processor computers. They usually communicate with other peripheral applications by means of data files, *Dynamic Data Exchange* (DDE), or even *Object Linking Embedding* (OLE), but the process of simulation is, in itself, sequential, not distributed. *CPN/Tools*, however, can also easily be used in a distributed environment, as *Coloured Petri nets* by essence provide a framework for the construction and analysis of distributed and concurrent systems (Kristensen et al., 1998).

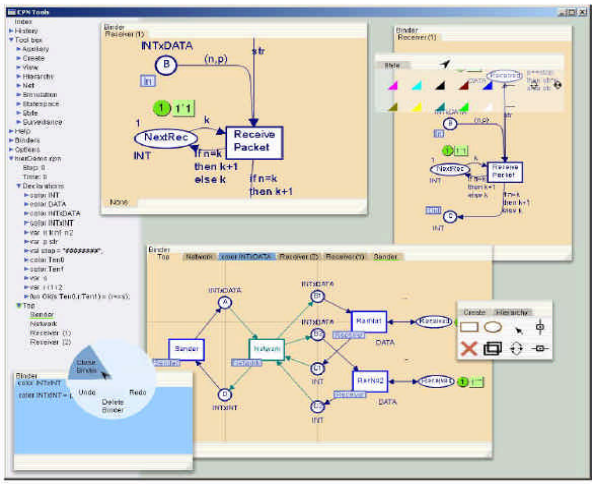

Fig. 1.21 A general view of *CPN/Tools*, a multipurpose simulation application (free software, can be licensed and downloaded from: http://www.daimi.au.dk/CPNtools/)

Anyhow, as we will see later, the Supply Chain simulator developed in the aim of this work was based on the OOP sequential technology, which at the moment is the most popular modelling technology in the industry, probably for being powerful, cheap and highly stable. Excellent OOP compilers are also spread to the community of programmers.

But the same cannot yet be said about some other interesting approaches like the *Agents*, or the *Web-based* simulation, or even about the *Distributed* or *Parallel* simulations. *Agents*, as a trial of merging *Artificial Intelligence* (AI) into distributed objects, see Nwana (1996), are still under discussion concerning the advantages of and even the need for control. Since these elements can be considered more free for action than the simple objects, and are said to exhibit the ability to learn, *Agents* naturally tend to impose a higher degree of uncertainty in the simulation process, what often complicates the interpretation of results and even the validation of the model. Perhaps due to this aspect, this technology is mainly used in research and to simulate systems where the dynamics emerge from the interaction between entities of the same kind, as, for instance, in artificial life studies, self-organizing structures, fractal mathematics, complex systems, etc. Their advantage seems to reside in the aspect of *interaction*. See also the PhD thesis of Travers (1996), presented at MIT.

Web-based simulation, on the other hand, basically focuses its interests on using common WWW interfaces to access remote simulators, in order to configure, run or even acquire results from them. These interactions are frequently done by means of a *Web* browser (Lau, 1997), the same kind of technology used in the *virtual enterprise*. Distributed users can therefore interact with the same simulation model. The online MIT *Beer Game* (MIT-FSCI, 2002) is a simple but good example of this type of approach.

Web-based modelling, anyhow, can also be seen as a collaborative network from where users would download basic blocks of simulation, and then build more complex models. Yet, little interest seems to exist in the industry regarding this idea, probably due to the difficulty of finding reliable models and to the instability in security while handling such resources in an open network like the *Web*. Integrated ambiences for modelling and simulation appear to be the preferable approaches for most industries of today.

In what concerns *Distributed* and *Parallel* simulations, the central value is generally the computational power that emerges from these solutions. At least while the *Quantum Computer* is not available — you can follow it at Wikipedia (2005). Therefore, these approaches are mainly used in projects involving large scale computing or high performance simulations, like the build and test of cosmological models, simulation of molecular motions and interactions, atmospheric and climate behaviour, as well as in military and space projects. A fast manipulation of high level mathematics and huge amounts of data are essential to these cases, rendering the power of processing fundamental.

Since long ago these architectures are being used by the scientific community, but lately they got more visible due to the interest revealed by the *United States Department of Defence* (DoD) on proposing a standard architecture to develop and run distributed simulations with high level of interoperation and maximum component reusability. The *High Level Architecture* (HLA) has resulted from this, see (DoD/US, 1997). More recently, as Robinson (2004)

notices, an international standards group based at Brunel University (HLA-CSPIF), is also looking to export the same idea for commercial simulation software. Names of relevancy on this kind of simulation are, between others, Fujimoto (1998), related with the distributed time management in the HLA/RTI framework; and, on promoting the spread of distributed simulation to the commercial sector, Zeigler et al. (1999), who explored the distributed DEVS/CORBA environment and even have presented the basis for a distributed Supply Chain simulator; and Sudra et al. (2000) as well as Taylor et al. (2002), who preferred the *Generic Runtime Infrastructure for Distributed Simulation* (GRIDS), which they also have proposed for distributed Supply Chain simulations. Between GRIDS and CORBA, however, the difference is mainly in the method used for time synchronization between components. CORBA uses a master-slave relation, while GRIDS uses what they call *ThinAgents*.

On the other hand, Jensen (1992) and his research group at *University of Aarhus*, Denmark, are the central references on parallel and distributed simulation with *Petri nets*, who recently were also adding *CPN/Tools* to the simulation tools used by NOKIA (DAIMI, 2003). This technology is also very popular in several German universities, and meanwhile has spread to Italy, Spain, USA and even Australia.

Although most of these distributed and parallel simulations traditionally required very expensive workstations or distributed computer systems, since the end of the last decade the attraction felt by people for building cheap supercomputers made of several ordinary computers, which become known as *Beowulf Clusters*[43], seemed to start to turn the attention to these powerful solutions, starting at research groups and universities. The recent interest of DELL *Computer* in these machines, who in 2002 was already testing *clusters* as an option for high-performance computing, see Shankland (2002), as well as the selection of these kind of machines to perform advanced weather modelling at the *Universal Weather and Aviation, Inc.* of Huston (linuxHPC, 2003), seem to suggest that supercomputing is probably preparing for soon approaching the industry. Besides, the release of *Microsoft dot-NET* framework can also be seen as another indication of the spreading trend of distributed computing…

Fig. 1.22 The AVALON, a *Beowulf Cluster* developed at *Los Alamos National Laboratory*, USA

In terms of modelling languages, the more relevant reference appears to be the *Unified Modelling Language* (UML), which proposes building the models using a standard graphical representation (OMG, 1999) very much inspired by the well known *activity diagrams*, and similar to what is

[43] Group of computers connected in network to perform parallel and distributed tasks. *Linux* running in common commercial computer machines is nowadays used to build these supercomputers at extremely low prices. Take a look at AVALON, from *Los Alamos National Laboratory* (USA): http://cnls.lanl.gov/avalon/.

already done since long ago in *Petri nets*, for instance. In that sense, UML is essentially a proposal to try to unify the practice of representing and developing models in a certain structured way, which can perhaps be an interesting cause to force analysts to express their ideas in the same form. But UML is mostly for model *description*, which comes before model *implementation*. For some interesting texts concerning this matter please see Fowler & Scott (1999), Arief & Speirs (2000), and, for instance, Barjis & Shishkov (2001) and DAIMI (2003).

On concluding, the field of simulation is nowadays a complex tree of approaches, proposals, theories and tools with which people are studying systems and optimizing processes, and some are even trying to face one of the most fascinating subjects: the prediction of the future. Till now, however, little more than good characterizations of certain systems have been achieved, as well as the creation of exciting and excellent virtual realities fast diverging from the true course of Time. Perhaps the future cannot be predicted by computational power, applying the ideals of Kant, but probably by simpler processes of intuition, more in the direction of what Bergson[44] suspected.

The tree shown in figure 1.23 represents the simulation approaches and some tools currently in use by the worldwide simulation community. It was build by the author based on data collected from proceedings of conferences, articles from journals and magazines, and the *Web*.

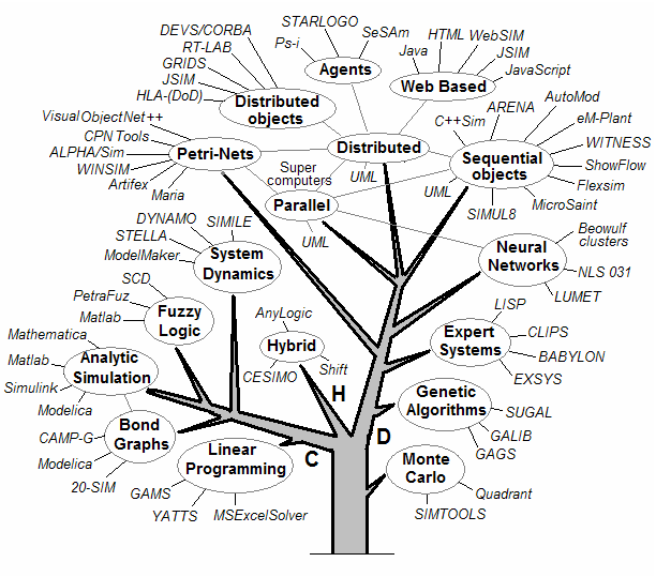

Fig. 1.23 A simplified tree of the simulation of our times

1.4 The future

The future is already embedded in the present, of course, but no one knows how it will be. Starting from the present, an infinite number of futures can emerge. Even if we could look at all them, the problem would be which one to choose. Thus, coming perhaps mostly from a *Bergsonian* inspiration, I would instead indicate what I expect future to bring us, from the current point of observation. Tomorrow, perhaps I will have a different position…

In terms of Supply Chain, it probably would be very interesting if the world would move more to the *react* (instead of *predict*) approach, getting closer to JIT philosophy. Even without the need for renaming it. This would imply a wider view on the systems and a mind-formation better focused on the reduction of waste and more effective and fair consumption and management of resources. To achieve this, anyhow, it would be crucial that traditional managers and

[44] Henri-Louis Bergson, a French philosopher from the beginning of 20th century.

researchers become "open minded" enough to truly understand such a paradigm, instead of wasting some more decades insisting on refusing it. "*The Earth is a finite system*", as Lehoucq[45] (2005) argues in a wise article regarding present energetic resources versus energy demand figures.

It terms of computer technology, my expectations would be that the *Quantum Computer* would soon become available, at least as a tribute to the exceptional mind of Heisenberg (1925). Finally, in the social: better health, and the move to the *Society of Knowledge*, not to the poorer *Society of Fantasy*.

[45] Roland Lehoucq is an Astrophysician from the *Commissariat à L'énergie Atomique* (Saclay, Paris).

References:

A.A.C.A. (2004). *Automotive History - Since 1960*. Antique Automobile Club of America. Retrieved Nov 2004, from http://www.aaca.org/history/cars_90.htm

ABX-Logistics. (2004). *ABX LOGISTICS (Deutschland) GmbH*. ABX LOGISTICS. Retrieved Nov 2004, from http://www.abxlogistics.com/DE/ENGLISH/history/index.aspx

Arief, L. B., & Speirs, N. A. (2000). *A UML tool for an automatic generation of simulation programs.* Paper presented at the the Second International Workshop on Software and Performance (WOSP2000), New York.

Barjis, J., & Shishkov, B. (2001, June 26-29). *UML Based Business Systems Modeling and Simulation.* Paper presented at the The 4th International EUROSIM Congress, Delft, The Netherlands.

Bell-Labs. (2002). *Bell Labs Early Contributions to Computer Science*. Lucent Technologies. Retrieved March 2004, from http://www.bell-labs.com/history/unix/blcontributions.html

Berners-Lee, T. J. (1989). *Information Management: A Proposal, in-house technical document*. CERN, 1989. 2003, from http://www.w3.org/History/1989/proposal.html

Blumberg, D. (1999). Strategic examination of reverse logistics and repair service requirments, needs, market size, and opportunities. *Journal of Business Logistics, 20* (2), 141-159.

Brito, A. E. S. C. (1992). *The Use of CAD Techniques in Configuring Visual Interactive Simulation Models: A New Approach for Warehouse Design.* Unpublished Ph.D., Cranfield Institute of Technology, Crandfield, UK.

Bruzzone, A., & Mosca, R. (1999, October 26-28). *Modelling & Simulation and ERP Systems for Supporting Logistics in Retail.* Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.

Buxton, & Laski. (1962). Control and Simulation Language. *Computer Journal, 5* (3).

C.H.M. (2004). *Time Line of Computer History: Computers*. Computer History Museum. Retrieved March 2004, from http://www.computerhistory.org/timeline/timeline.php?timeline_category=cmptr

C.I.I.C. (2004). *Space Industry of China*. China Internet Information Center. Retrieved Nov 2004, from http://www.china.org.cn/english/SPORT-c/77126.htm

CACI. (1986). *SIMFACTORY*. Los Angeles: CACI Inc.

Carrie, A. (1988). *Simulation of Manufacturing Systems*: John Wiley & Sons.

Chandrasekaran, U., & Sheppard, S. (1987). *Discrete event Distributed Systems - A survey.* Paper presented at the Conference of Methodology and Validation, Orlando, Fla., USA.

Chandrashekar, A., & Schary, P. B. (1999). Toward the Virtual Supply Chain: The convergence of IT and Organization. *Journal of Logistics Management, 10* (2), 27-37.

Chandy, K. M., & Misra, J. (1979). Distibuted Simulation: A Case Study in Design and Verification of Distributed Programs. *IEEE Trans. Software Eng., SE-5*, 440-452.

Chen, Y., Peng, Y., et al. (1999). *A Negotiation-based Multi-agent System for Supply Chain Management.* Paper presented at the ACM Autonomous Agents Workshop on Specifying and Implementing Conversation Policies, Seattle.

Christopher, M. (1992). *Logistics and Supply Chain Management: Strategies for Reducing Costs and Improving Services* (second ed.). London: Pitman Publishing.

Chryssolouris, G. (1992). *Manufactoring Systems, theory and practice*: Springer-Verlag.

CNN. (2001, December 10, 2001). *China's long march to WTO entry*. 2004, from http://archives.cnn.com/2001/WORLD/asiapcf/east/09/18/china.wto.timeline/

Curran, C. (1991). Integrated Supply Chain Information Systems: The Next Phase after EDI? *Logistics Information Management, 4* (1).

DAIMI. (2003). *UML + CPN @ Nokia*. DAIMI - University of Aarhus. Retrieved January 2005, from http://www.daimi.au.dk/CPnets/UMLCPNatNokia/

DoD/US. (1997). *HLA Overview*. Defense Modeling and Simulation Office (DMSO), DoD/USA. from http://www.dmso.mil/dmso/docslib/

Dong, M. (2001). *Process Modeling, Performance Analysis and Configuration Simulation in Integrated Supply Chain Network Design.* Unpublished Ph.D., Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Dutta, D. (2002). *Retailer @ the speed of fashion* (ZARA case study): http://www.3isite.com.

E.U. (2004). *Gateway to the European Union: the history of European Union*. European Union. Retrieved Nov 2004, from http://europa.eu.int/abc/history/index_en.htm

Eilon, S., Wantson-Gandy, C. D. T., et al. (1971). *Distribution Management: Mathematical modeling and practical analysis*: Griffin-London.

Ellram, L. M. (1991). Supply Chain Management: The Industrial Organization Perspective. *International Journal of Physical Distribution & Logistics Management, 21* (1), 13-22.

Eymann, T., Padovan, B., et al. (1998, July 2-8). *Simulating Value Chain Coordination with Artificial Life Agents, accepted Poster.* Paper presented at the 3th International Conference on Multi-Agent Systems (ICMAS'98), Paris.

Feigin, G., An, C., et al. (1996). Shape Up, Ship Out. *OR/MS Today (Operations Research and the Management Sciences) Online Edition, 23*.

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (1999). *Visual C++ Software for Warehouse Simulation (an overview).* Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.

Fiddy, E., Bright, J. G., et al. (1981). SEE-WHY: Interactive Simulation on the Screen. In *Proceedings of the Institute of Mechanical Engineers C293/81, I. Mech. E* (pp. 167–172). London.

Flood, M. M. (1956). The travelling salesman problem. *Ops. Res., 4*, 61-75.
Forrester, J. W. (1960). *Industrial Dynamics*. Cambridge, MA: MIT Press.
Forrester, J. W. (2004). *Origin of System Dynamics*. System Dynamics Society. Retrieved March 2004, from http://www.systemdynamics.org/DL-IntroSysDyn/orig_f.htm
Fowler, M., & Scott, K. (1999). *UML Distilled: Applying the Standard Object Modeling Language*: Adddison Wesley Longman, Inc.
Fox, M. S., Chionglo, J. F., et al. (1993). *The Integrated Supply Chain Management System*: Dep. of Industrial Engineering, University of Toronto, Canada.
Fujimoto, R. M. (1998). *Time Management in the High Level Architecture*: College of Computing, Georgia Institute of Technology, Atlanta.
Furey, T. (2003). *Pump Up the Pipeline: How to cure anemic revenue growth* (Report). Bethesda: MarketBridge Corp.
Gattorna, J. (1983). *Handbook of Physical Distribution Management* (3th ed.): Gower Publishing Company Ltd.
Goldman, S., Nagel, R., et al. (1995). *Agile Competitors and Virtual Organizations*. New York: Van Nostrand Reinhold.
Goldratt, E. M., & Fox, R. E. (1986). *The Race*: North River Press.
Gordon, G. (1961). *A general purpose systems simulation program.* Paper presented at the Eastern Joint Computer Conference, Washington, D.C.
Graham, M., & Wavish, P. R. (1991, June). *Simulating and Implementing Agents and Multiple Agent Systems.* Paper presented at the European Simulation Multi-Conference, Copenhagen.
Guedes, A. P. (1994). *An Integrated Approach to Logistics Strategy Planning Usig Visual Interactive Modelling and Decision Support.* Unpublished Ph. D., University of Cranfield, U. K.
Harris, F. W. (1913). How Many Parts to Make at Once. *Factory: The Magazine of Management, 10,* 135-136.
Harrison, A., Christopher, M., et al. (1999). *Creating the Agile Supply Chain*. Cranfield, UK: School of Management, Cranfield University.
Heizer, J., & Render, B. (2001). *Operations Management*: Prentice Hall.
Herring, C. (1990). *MODSIM: A New Object Oriented Simulation Language.* Paper presented at the Multi-conference on Object-Oriented Systems, San Diego, CA.
Hewitt, F. (2001, May 2001). After Supply Chains, Think Demand Pipelines. *Supply Chain Management Review, 5,* 28.
Hopp, W. J., & Spearman, M. L. (2001). *Factory Physics* (second ed.): Irwin McGraw-Hill.
Howell, F., & McNab, R. (1998, Jan 1998). *simjava: a discrete event simulation package for Java with applications in computer systems modelling.* Paper presented at the First International Conference on Web-based Modelling and Simulation, San Diego CA, USA.
Hurrion, R. D. (1976). *The design, use and required facilities of an interactive visual computer simulation language to explore production planning problems.* Unpublished PhD, University of London.
Jensen, K. (1992). *Coloured Petri nets. Basic Concepts, Analysis Methods and Practical Use* (Vol. 1: Basic concepts). Berlin: Spinger-Verlag.
Kemers, J., & Merkuryev, Y. (1999). *Simulation Application for Logistics Strategy Planning in the Baltic Market.* Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.
Kinnersley, B. (1995). *Collected information on about 2350 computer languages, past and present* (online document). Lawrence, KS 66045: Computer Science Department, University of Kansas.
Kleinrock, L. (1975). *Queueing Systems* (Vol. I). New York: Wiley.
Knemeyer, A. M., Ponzurick, T. G., et al. (2002). A qualitative examination of factors affecting reverse logistics systems for end-of-life computers. *International Journal of Physical Distribution & Logistics Management, 32* (6), 455-479.
Kristensen, L. M., Christensen, S., et al. (1998). *The practitioner's guide to colouredPetri nets*: Springer-Verlag.
Lackner, M. R. (1964). Digital Simulation and System Theory, *System Development Corporation*. Santa Monica, CA.
Lambert, D. M., & Pohlen, T. L. (2001). Supply Chain Metrics. *The International Journal of Logistics Management, 12* (1), 1-19.
Lau, L. (1997). *Dynamic Simulation Through the WWW.* Paper presented at the Australian World Wide Web Technical Conference, Brisbane, Queensland, Australia.
Law, A. M., & Kelton, W. D. (1991). *Simulation Modeling & Analysis* (second ed.): McGraw-Hill International Editions.
Lehoucq, R. (2005). Contagem Decrescente. *Le Monde Diplomatique (portugese version)*, 11.
linuxHPC. (2003). *Universal Weather & Aviation Selects PSSC Labs' Linux-Based Beowulf Clusters*. LinuxXHPC.org. Retrieved Jan 2005, from http://www.linuxhpc.org/stories.php?story=03/07/18/3255917
Little, M. C., & McCue, D. L. (1994). Construction and Use of a Simulation Package in C++. *C User's Journal, 12* (3).
Maersk-Logistics. (2004). *Maersk Logistics: Our history*. Maersk Logistics. Retrieved Nov 2004, from http://www.maersk-logistics.com/sw17894.asp
Mason-Jones, R., Naim, M. M., et al. (1997). The Impact of Pipeline Control on Supply Chain Dynamics. *The International Journal of Logistics Management, 8* (2), 47-61.
Mason-Jones, R., & Towill, D. R. (1999). Using the Information Decoupling Point to Improve Supply Chain Performance. *Journal of Logistics Management, 10* (2), 13-24.
McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys* (5), 115-133.
Metz, P. J. (1998). *Demystifying Supply Chain Management*. MIT-Industry Integrated Supply Chain Management Program. 2003, from http://www.econ.cbs.dk/organizations/logistik/images/myst.htm

MIT-FSCI. (2002). *MIT Beergame*. MIT Forum for Supply Chain Innovation. Retrieved Dec 2004, from http://beergame.mit.edu/default.htm

Müller, G., Eymann, T., et al. (1999). *Economic Risks of Market-based Supply Chain Coordination Using Multi-agent Systems*. Albert-Ludwigs-UniversitätFreiburg: Proposal to the SPP "IntellignetSoftwareagenten und BetriebswirtschaftlichAnwendungsszenarien" of DeutcheForschungsgemeinschaft, Institut für Informatik und Gesellschaft.

Müller, R. (2000). Scientific Models: Their Historical and Philosophical Relevance. In I. D.-D. J. Commission (Ed.), *The 13th International Conference on History and Philosophy of Science*. University of Zurich.

N.H.C. (2000). *Logistics & Support Activities, 1950-1953*. DEPARTMENT OF THE NAVY - NAVAL HISTORICAL CENTER, WASHINGTON NAVY YARD, WASHINGTON DC 20374-5060. Retrieved March 2004, from http://www.history.navy.mil/photos/events/kowar/log-sup/log-sup.htm

Nance, R. E. (1993, April 20-23). *A History of Discrete Event Simulation Programming Languages.* Paper presented at the The second ACM SIGPLAN conference on History of programming languages, Cambridge, Massachusetts, United States.

Nwana, H. S. (1996). Software Agents: An Overview. *Knowledge Engineering Review, 11,* 1-40.

Ohno, T. (1978). *Toyota seisam hoshikiti*. Tokyo: Diamond.

Ohno, T. (1988). *Toyota Production System: Beyond Large-Scale Production*. Cambridge, MA: Productivity Press.

OMG. (1999). *Unified Modeling Language Specification, Version 1.3*: Object Management Group (OMG).

OR/MS. (1996). Supply Chain Simulation Model. *OR/MS Online Edition, Lionheart Publishing, 23* (2).

Orlicky, J. (1975). *Meterial Requirements Planning: The New Way of Life in Production and Inventory Management*. New York: McGraw-Hill.

P.N.W. (2004). *Welcome to the Petri Nets World*. Petri Nets World. 2004, from http://www.daimi.au.dk/PetriNets/

Paynter, H. M. (1961). *Analysis and Design of Engineering Systems*. Cambridge, MA, USA: MIT Press.

Petri, C. A. (1962). Fundamentals of a Theory of Asynchronous Information Flow (pp. 386-390): IFIP Congress.

Pidd, M. (1984). *Computer Simulation in Management Science*. New York: John Wiley.

Pidd, M. (1992). *Computer Simulation in Management Science*. Chichester, England: John Wiley & Sons.

Ritchie, L., Burnes, B., et al. (2000). The benefits of reverse logistics: the case of the Manchester Royal Infirmary Pharmacy. *Supply Chain Management: An International Journal, 5* (5), 226-233.

Rizzoli, A. E. (2004). *A Collection of Modelling and Simulation Resources on the Internet*. Andrea Emilio Rizzoli, IDSIA, Switzerland. Retrieved Dec 2004, from http://www.idsia.ch/~andrea/simtools.html

Robinson, S. (2004). Discrete-event simulation: from the pioneers to the present, what next? *Journal of the Operational Research Society*, 1-11.

Schruben, L. (1983). Simulation Modeling with Event Graphs. *Communications of the ACM, 26* (11), 957-963.

Shankland, S. (2002, November 21). *InfiniBand reborn for supercomputing*. from http://news.zdnet.com/2100-9584_22-966777.html

Shannon, R. E. (1975). *Systems Simulation, the art and the science*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Shewhart, W. A. (1931). *Economic Control of Quality of Manufactured Product*. New York: Van Nostrand.

Silva, M. (1993). *Practice of Petri nets in Manufacturing*. London: Chapman & Hall.

SOLE. (1999). *SOLE - The International Society of Logistics: Logistics Bibliography*. SOLE - The International Society of Logistics. Retrieved Nov 2004, from http://sole.org/bibliography.asp

Stephens, J. (2003). Custodians of Memory: The Lost Figures of May '68. *Australian Review of Public Affairs*.

Sterman, J. D. (1992). *The Beer Distribution Game: an Annotated Bibliography Covering its History and Use in Education and Research*. Cambridge, MA: Sloan School of Management, Massachusetts Institute of Technology (MIT).

Stipp, D. (2005). *The Pentagon's Weather Nightmare* (Article): FORTUNE (online edition) http://www.fortune.com/fortune/technology/articles/0,15114,582584-1,00.html.

Sudra, Taylor, S. J. E., et al. (2000). *Distributed Supply Chain Simulation in GRIDS.* Paper presented at the Winter Simulation Conference, USA.

Sussams, J. E. (1992). *Logistics Modelling*. London, UK: Pitman Publishing.

Taylor, F. W. (1903). Shp Management. *Transations of the ASME, 24*, 1337-1480.

Taylor, S. J. E., Sudra, R., et al. (2002). GRIDS-SCF: An Infrastructure for Distributed Supply Chain Simulation. *SIMULATION, 78* (5), 312-320.

Tecnomatix. (1999). *Customer Success: ATS Automation-France*. Tecnomatix Technologies Ltd. Retrieved Dec 2004, from http://www.tecnomatix.com/showpage.asp?page=521

TheHistoryPlace. (1999). *The Vietnam War, Seeds of Conflict*. The History Place. Retrieved April 2004, from http://www.historyplace.com/unitedstates/vietnam/index-1945.html

Tibben-Lembke, R. S. (2002). Life after death: reverse logistics and the product life cycle. *International Journal of Physical Distribution & Logistics Management, 32* (3), 223-244.

Tocher, K. D. (1963). *The Art of Simulation*. London: The English Universities Press.

Towill, D. R. (1992). Supply Chain Dynamics - The Change Engineering Challange of the Mid-90's. *Institute of Mechanical Engineers on Engineering Manufacture,* 233-245.

Townsend, M., & Harris, P. (2004). Now the Pentagon tells Bush: climate change will destroy us. *The Guardian Newspapers Limited (online)* (February 22, 2004).

Travers, M. D. (1996). *Programming with Agents: new metaphors for thinking about computation.* Unpublished Ph.D., Massachusetts Institute of Technology (MIT).

UN. (2004). *United Nations Department of Economical and Social Affairs, Division for Sustainable Development*. United Nations. Retrieved Dec2004, from http://www.un.org/esa/sustdev/natlinfo/2004survey_E.pdf

Vossebeld, R. (2002, May 2002). Simplifying the Supply Chain. *Logistics Manager,* 25-26.

Waller, A. P., & Ladbrook, J. (2002). *Experiencing virtual factories of the future.* Paper presented at the the 2002 Winter Simulation Conference, IEEE, Piscataway, NJ.

Weber, A. (1909). *Uber den Standort der Industrien*: Tuebingen.

Whitin, T. M. (1953). *The Theory of Inventory Management*. Princeton, NJ: Princeton University Press.

Wikipedia. (2004a). *1950's Events and Trends*. Wikipedia, the free encyclopedia. Retrieved Mar 2004, from http://en.wikipedia.org/wiki/1950s

Wikipedia. (2004b). *Empowerment*. Wikipedia, the free encyclopedia. Retrieved Nov 2004, from http://en.wikipedia.org/wiki/Empowerment

Wikipedia. (2004c). *Globalization*. Wikipedia, the free encyclopedia. Retrieved Dec 2004, from http://en.wikipedia.org/wiki/Globalization

Wikipedia. (2005). *Quantum computer*. Wikipedia, the free encyclopedia. Retrieved Jan 2005, from http://en.wikipedia.org/wiki/Quantum_computer

Williams, J. L. (2003). *Oil Price History and Analysis*. WTRG Economics. Retrieved Nov 2004, from http://www.wtrg.com/prices.htm

Wilson, G. (1994). *The History of the Development of Parallel Computing*. PARALLEL.RU. Retrieved Nov 2004, from http://parallel.ru/history/wilson_history.html

Zeigler, B. P. (1976). *Theory of Modelling and Simulation*. New York, NY: John Wiley and Sons.

Zeigler, B. P., Kim, D., et al. (1999). *Distributed Supply Chain Simulation in a DEVS/CORBA Execution Environment.* Paper presented at the Winter Simulation Conference, USA.

Zobel, R. N. (2001). A Personal History of Simulation in the UK and Europe. *I. J. of Simulation, 1* (1-2), 69-79.

CHAPTER 2 51
ON MODELLING THE SUPPLY CHAIN 51
2.1 SYSTEMS AND MODELS 51
2.2 SUPPLY CHAIN OVERVIEW 52
2.2.1 Elements of the chain 52
2.2.2 Flows and resources 56
2.2.3 Management levels and policies 58
2.2.3.1 Inventory policies 60
2.2.3.2 Production policies 62
2.2.3.3 Distribution policies 65
2.2.4 Traditional metrics and flexibility 67
2.2.4.1 Facility related metrics 68
2.2.4.2 Fleet related metrics 71
2.2.4.3 The emphasis on flexibility 71
2.2.5 A step up to holistic metrics? 72
2.3 SUPPLY CHAIN MODELLING 74
2.3.1 System Dynamics 75
2.3.2 Petri nets 75
2.3.3 Sequential objects 76
2.3.4 Distributed objects 78
2.3.5 Agents 80
2.3.6 Web-based 81
2.3.7 Parallel 83
2.4 WHICH APPROACH TO CHOOSE 84
2.4.1 Level of detail 84
2.4.2 Where to stop 85
2.4.3 Simulation intents 87
2.5 MODEL CONSISTENCY, ACCURACY 89
2.6 GETTING RESULTS WITH THE MODEL 92
2.7 FINAL COMMENTS 94
REFERENCES: 97

Chapter 2

On Modelling the Supply Chain

Relevant issues, modelling approaches and technologies

2.1 Systems and models

Words like "*system*" and "*model*" are sometimes used with little distinction in simulation literature, mostly in articles, and at this point it is crucial to establish what each of these terms signify in the aim of the present text: a *system* will be considered the ultimate *object* of a study. A *model*, on the other hand, will be understood as any sort of *representation* of such *system*, in which experiments can probably be made and from which *indirect* conclusions can be obtained concerning the behaviour of the *system*.

As shown in figure 2.1, *systems* and *models* are in fact two different things that may not be confused during any moment in the simulation process. A prior, this remark could well seem superfluous, but the fact is that some confusion often takes place since simulation is a process where frequently the *system* is absent, not in real touch with the analyst, and this could easily induce the loss of a certain reference to reality. Mostly when using the excellent graphics simulators of nowadays, this psychological effect can easily become observable. Of course, also due to this particular feature simulation is many times used as a powerful instrument for marketing.

Anyhow, to be aware of this is essential to a reliable process of simulation. Not to lose the ground by confusing the *model* with the *system* is a basic obligation.

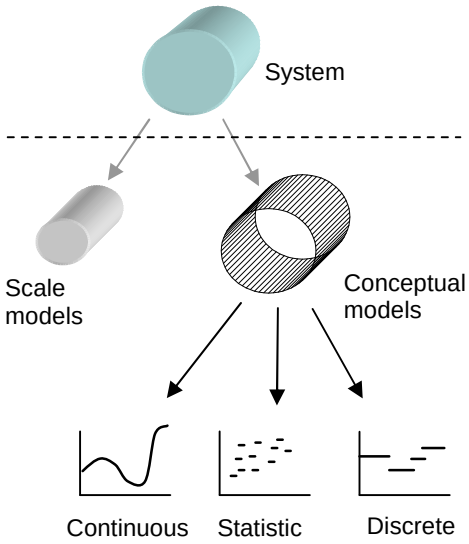

Fig. 2.1 Models as representations of systems

Somewhat inspired by the classification proposed by Law & Kelton (1991), figure 2.1 shows that a *system* can in principle be represented as a *model* in two different ways: by means of a *scale sculpture* (scale model), or by a *conceptual representation*, not physical, in which the model is built by mathematical formulae and relations. Then, based on Davies & O'Keefe (1989), one can also classify the *conceptual models* as: (a) *Continuous*, when the states of the system

are represented as a continuous set of values; (b) *Statistical*, when such representation is static and the results are achieved by statistical methods, like the Monte Carlo method, for example; and (c) *Discrete*, when the states of the system are represented as a discrete set of states, that usually leads to a discrete representation also along the time variable.

As we will see soon, all these approaches have been explored for modelling Supply Chain systems, and their choice strongly depends on the objectives of the analyst, which can be strategic, tactical or even operational. At the moment, however, let us just notice that since the usual sequence of actions in the process of simulation is (1) understand the system; (2) create a model of the system; (3) implement the model; and, finally, (4) simulate the model; this chapter is focused on boarding and making clear such issues, starting by an overview on the system.

2.2 Supply Chain overview

Theoretically, any simulation technology or approach could be used for modelling Supply Chain systems. Like other systems, the Supply Chain can be simulated as long as a model of it is previously built. Anyhow, as it happens in general in simulation, prior to building the *model* one has to understand how the *system* really works. Only based on such knowledge can a reliable "image" of the system later be constructed. That is the purpose of this section.

2.2.1 Elements of the chain

From the physical point of view, the Supply Chain is a network of companies connected with one another by means of three basic kinds of flows: *materials*, *information* and *money*. The first two flows give rise to what in the literature is called the *materials pipeline* and the *information pipeline* (Chandrashekar & Schary, 1999). However, perhaps due to an understandable modesty, *money* is usually excluded from this list, even though being the main engine for the other processes. Yet, in truth the Supply Chain is basically a huge system in which *money* flows from ultimate customers to the entire list of enterprises involved in the business. Of course this flow is in exchange of products (including services), and, in that sense, it is easier to recognize that ultimate customers are the true drivers of the Supply Chain activities. If they would stop buying products, the entire system would collapse. Figure 2.2 is an attempt to depict this idea.

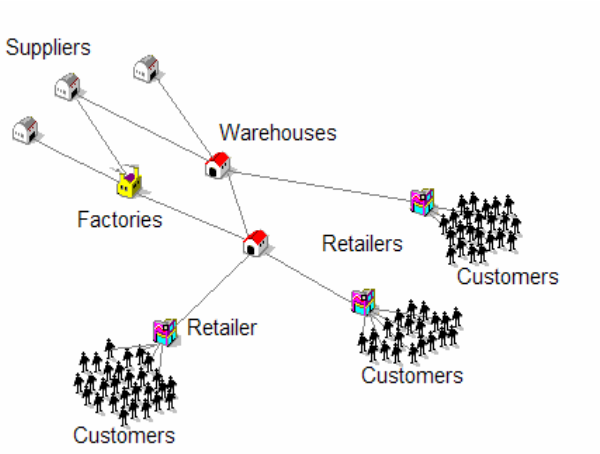

Fig. 2.2 The general structure of a Supply Chain, where ultimate customers are also represented (*Customers*)

Typically, the companies forming part of these kinds of networks are distinguished by the major activities they perform in the business, which usually are classified as: *suppliers*, when focused on the handling and delivering of raw materials or intermediate

products which later will be transformed into other products; *factories*, if the main activity is the manufacturing of finished products; *warehouses*, or *depots*, when the primary objective is to handle significant inventories of finished products as interface buffers for the last sellers located at the points of consumption; *retailers*, or last sellers, usually engaged in strategies of selling and handling small inventories; and, finally, the *ultimate customers*, habitually simply called *customers*, whose activity is to get the products and pay for them. The difficulty of predicting the behaviour of this mass of *customers* is actually a headache for most Supply Chain managers, and, in trying to turn demand figures smoother, many companies frequently take advantage of "spy" and "pressing" tactics, for instance, customer profile monitoring, advertisement, public exhibitions, as well as other psychological techniques embedded in marketing operations. As a curiosity, several enterprises are seriously interested in introducing *Radio Frequency Identification* (RFI) to track every item of its products worldwide by using a chip the size of a grain of sand that can be identified when it approaches a "reader" (Microsoft, 2003). The technology is from the times of World War II, but, as the same source reveals, *"now many experts believe that it will be found in anything and everything by 2010, and it will revolutionize supply chain, manufacturing and retail efficiency*".

It is a fact that the supply networks have turned more complex during the last decade, mainly due to the spread of *global* enterprises and worldwide procurement, but also due to the trend of replacing the concept of *vertical integration* by the idea of *outsourcing*. Instead of being owned by a single company, as in the old times of regional consumption, the supply network is presently a space of self-ruling enterprises among which *materials*, *information* and *money* must be shared. And this makes the dynamics of these flows more voracious and complex than ever. As we have seen in the previous chapter, *Supply Chain Management* (SCM) is committed to embracing the new challenges presented by this business reality.

In simulation, however, despite the usage of different blocks of logic to create specific models for the different elements in the Supply Chain, the entire Supply Chain system is rarely represented in practice, due to the increase in complexity a broader representation would add to the model and to the results. In most cases, Supply Chain models are reduced to a certain number of warehouses connected to retailers, or, at a maximum, including some factories. A few central warehouses delivering to an extensive group of retailers, these in numbers of 50, 100 or even 200, can be expected in a real case Supply Chain model, as in Guedes (1994). On the other hand, academic studies are commonly carried out in models of networks having less than ten facilities, as in the famous "*Beer Game*" (Mason-Jones et al., 1997), or even in the "*Cranfield Blocks Game*" (Feliz-Teixeira & Brito, 2005), for example.

For the same motivation of simplicity, it is also common to ignore the *primer suppliers* and the *ultimate customers*, being the retailer the driver for the demand in most Supply Chain simulations.

Apart from these tendencies, and since in the simulation approach presented in the next chapter all elements and facilities will be considered, it is at this point convenient to focus our attention on each element constituting the Supply Chain:

• **The customer**: after being "pressed" by every sort of advertisement, if possible, the customer effectively establishes the *direct demand* to the Supply Chain. Thus, the customer demand is the input to the system. Historical demand figures are usually treated as time-series resulting from the superposition of four basic kinds of components (see figure 2.3): *trend*, which represents a gradual and constant variation over a long period of time; *seasonality*, as a pattern repeating itself in a certain period of time; *cycles*, as patterns repeating after long periods of time; and *random variation*, as the unpredictable component of "noise" in the demand signal. Forecasts are based on the analysis of these kinds of signals (Heizer & Render, 2001, pp. 83).

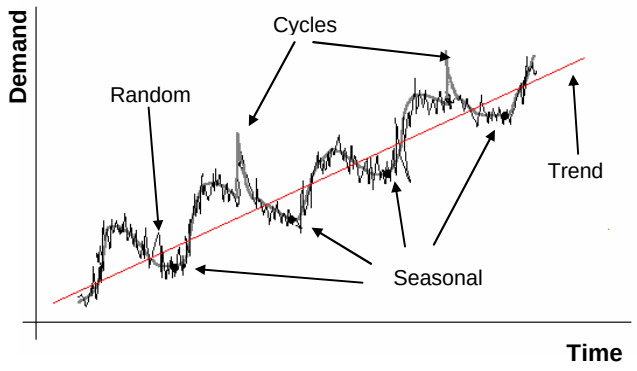

Fig. 2.3 A generic demand signal and its components

But, how the demand is generated is also important. Although traditionally customer orders are placed to retailers, they can be placed to other facilities in the Supply Chain as well, like central warehouses or certain factories, by using the classical method of ordering by catalogue or even the new *e-Commerce* services available by *Internet*, as what happens in the "Amazon.com" online bookstore or in the trendy "eBay.com" online market, for example. In practice, customer demand is injected in the Supply Chain model as a collection of historical values or by means of a distribution of probabilities, frequently in the form of a standard *Normal* function. For simplicity, customers demand is almost always generated as an accumulated demand per day, per week, per month, etc. Besides, the customer is considered not to own any kind of resources, like stock or fleet.

• **The retailer**: The retailer receives as input the demand from customers (orders) and, in reply to that, it is responsible for delivering the respective materials. To maintain the appropriate level of stocks, the retailer, in turn, will place orders for stock replenishment whenever necessary to its "suppliers", which can be warehouses, factories, or even other retailers, depending on the case. However, in general, retailers place orders to warehouses. The retailer owns a physical stock, or inventory, and, of course, an inventory replenishment policy. The amount of inventory is in general small, and usually only to cover the demand during the time needed for replenishment. The delivery process is normally done by some kind of transporters. Communication with "suppliers" is frequently made by electronic methods like *Electronic Data Interchange* (EDI), email, or even using *e-Commerce*. Nowadays, *Information Technologies* (IT) allow better synchronization between the players in the Supply Chain. As the retailer

operates a stock and is responsible for a delivery, it has at least an inventory refilling policy as well as a delivery policy. In some practical simulation cases, the retailer is modelled as a black box characterized by a list of parameters like the *lead-time*, *frequency of replenishment*, *fixed costs*, *stock operation costs*, *delivery costs*, etc., which can also be computed from standard distribution functions. As mentioned already, the retailer replenishment orders are often considered the first input to the Supply Chain.

• **The warehouse**: warehouses, or depots, are by excellence the elements centred on handling large amounts of inventory, and the facilities to which several retailers place their orders. Warehouse performance is, therefore, strongly dependent on the inventory management policy adopted and on the distribution policy, since the network of distribution significantly increases in complexity, in comparison with the retailer's case. A tied management on these policies is vital, and implies good synchronism between orders' fulfilment, the scheduling of their transportation and the moments of replenishment. Big automated warehouses, typical of the 1990s, could occupy the space of thousands of squared metres, where tens of thousands of palettes of materials where kept in stock. Physically, they included arrays of bays for input and output and truck alignment while loading and unloading, as well as special conveyor networks and crane vehicles for handling the palettes inside the complex warehouse structure. In general, EDI, email, and *e-Business* are nowadays technologies for communicating with partners and with the "outside world". Normally, warehouses place orders to other warehouses, factories or general suppliers. For the purpose of Supply Chain simulation, the warehouse is usually modelled as a box including a stock resource being managed by some standard inventory control policy. As in the previous case, a list of parameters allows the configuration of the element by the user, like the *lead-time*, *stock operation times*, *costs*, etc.

• **The factory**: Factories can be complex elements. They receive raw materials and make products, or receive intermediate products and make other more complex products, which in turn can later be parts for other products manufactured in other factories. The assembling, as well as the customization, from this perspective, may be seen as later manufacturing processes. As the most complex process of the Supply Chain, manufacturing has been intensively studied for a long time. An excellent book about the matter is "*Factory Physics*", from Hopp & Spearman (2001). Nevertheless, inventory flows in the factories are formally very similar to those of warehouses, since the main issue is to find the adequate quantity to produce (lot size) and the best moment to do it; in fact, identical to the warehouse problem of finding the quantity to order to the supplier and the best moment to do it. Many authors indicate no differences between the two cases, and, in some simulations, factories can simply be found modelled as warehouses. In other situations, however, and mainly if higher accuracy is expected, other specific manufacturing factors must be taken into account, like process times, maximum production rates, number and organization of production lines, lot sizes, etc. In

general, factories place orders for stock replenishment to the suppliers of raw materials, but in reality they can also order to other facilities, especially when the processes of manufacturing are based on assemblage, as, for instance, in the case of the *Assemble-To-Order* (ATO) practiced by DELL *Computer*[1].

• **The supplier**: In the present context, "supplier" is to mean "primer supplier", the kind of facility which acts as the interface between the world of *raw materials* and the world of *products*. Suppliers typically handle unprocessed materials, but in terms of inventory can be seen as warehouses that provide such materials to the manufacturers of products. The classification of "supplier", anyhow, is not very linear in practice, as it really depends on how one looks at the problem. For instance, a farmer producing vegetables for the food industry can simultaneously be considered a kind of "factory" and a "supplier". It depends on whether the vegetables are being delivered to the city market or to a factory of vegetarian pies, respectively. Besides, the raw material can already be a product, as in the case of DELL, for example; or when a constructor of airplanes orders a wing from a certain supplier which may be in itself a factory receiving steel from other industrial suppliers. For the purpose of simulation, "primer suppliers" are frequently ignored and substituted by simple sources of materials. So, few simulations count with the lead times and the delays caused by stock operations at the supplier level. Anyhow, if taken into account, this element is very much considered a warehouse being served by an infinite source of materials.

Apart from such definitions of the static elements (nodes) of the Supply Chain, sometimes it is not so simple to classify them in practice. In the previous chapter we have talked of certain emergent notions like "pipeline management", which transforms the concept of management in each Supply Chain facility, as well as about "moving factories", for instance, where some steps of the manufacturing process are to be executed while in transit to the client. With the constant evolving of manufacturing and delivering processes, certain nodal elements of the Supply Chain can exhibit a kind of mixed "behaviour", if expressed in terms of the traditional classification. If one asks oneself what in truth is DELL *Computer?* Is it a factory? A retailer? A warehouse? The answer appears to become less easy than expected. Even recognizing the importance of the usual classification, the fact is every node of the chain is in truth an element of business, which will organize or arrange its business in the form in which it will become more effective and advantageous. In reality, each of these elements is simply a weighted equation of the classical archetypes. As we will see in the next chapter, a proposal for a generalization of the representation of these elements in the simulation will be presented.

2.2.2 Flows and resources

Between the elements of the Supply Chain there is an interchange of *materials*, *information* and *money*, as it was previously

[1] DELL *Computer* is one of the most successful companies of our days using fast response logistics, derived from the JIT philosophy. It assembles Personal Computers and delivers them to the client's door.

said. The physical structures and resources that support these flows are, however, quite diverse.

Money usually flows throughout credits, bank transfers, cheques, and is therefore centred in bank operations. Despite the fact that this flow is ignored in most simulation approaches, the effectiveness of certain practical Supply Chain can especially rely on it, or even extremely depend on it, if credit degrades. For example, suppliers can stop delivering to their clients due to the lack of payment or excessive delays in the process. This may seem a "third world" reality, but it is a serious factor to take into account while modelling certain Supply Chains. Probably, any well organized company from the "developed world" would not be able to survive random blockades in the flow of money. Being less dramatic, one could at least contemplate how this flow affects decisions in the facilities and their responses, or even their reorder policies. As an example from the public domain show, some factories installed in certain countries have later opted to leave those countries not only due to the less expensive labour offered by recently joined eastern states of European Union, but also due to serious problems related to the flow of money. Delays at this level can easily induce delays in the channels of information and materials and, from that, generate dramatic conduct in the Supply Chain.

Information flow, as learned from the previous chapter, can be considered the radical reality of our times. Supported by the traditional telephone, post and FAX, and appended with modern technologies of EDI, email and *e-Business*, and enhanced with the cell-phone flexibility, the information flow also became decisive in Supply Chain activity. By means of these resources, partners exchange data about inventories, production schedules, product specification, transport coordinates, etc., with the intent to better synchronize their operations. Such communicative ambience naturally shapes itself in the form of a network of relations between companies, and, due to the weight the "virtual world" of *Internet* carries in such ambience, by *e-Commerce*, *e-Business*, *e-Procurement*, etc., soon was seen as a new business dimension named "*Virtual Supply Chain*" (Chandrashekar & Schary, 1999). Since then, information flow was also considered an interesting matter for simulation, but mostly focused on modelling the coordination and cooperation between partners and/or departments. *Agent-based* simulation (Kim et al., 2000) as well as *Web-based* approaches (Lee et al., 2001) were technologies commonly used for this purpose. The trend was also following the development of software for *Enterprise Resource Planning* (ERP). These approaches, however, tend to represent the companies of the Supply Chain by their organizational structure of departments, and not by their traditional physical processes.

On the other hand, when the Supply Chain is modelled based on its physical processes (including cycles of production, warehousing, distribution), the concept of information flow is normally absent and communications are considered with no delays. Yet, in reality the answer to an email is many times delayed, or even not happening at all, thus able to induce considerable effects on the performance of the system.

Materials flow is the central flow in the Supply Chain. It holds the merchandise to be exchanged for money; it is the physical generator of money. Therefore, the faster the merchandise will be delivered to the client the better. Materials flowing in the Supply Chain are indeed an interesting phenomenon: everybody needs them to sell but no one wants them in hand. Materials are like hot potatoes, or cartoon time-bombs.

The flow of materials goes along the *transport paths* that compose the physical network of the Supply Chain. Trucks, trains, airplanes, ships, pipelines, are transport resources commonly used along these paths to handle the materials on its distribution to the clients. These transport resources can either be owned by the distributor or be subcontracted as a service to a Logistics company dedicated to the transport of goods, usually known as *3PL* operators. The costs of these operations are therefore seen in different ways: in the first case, there is normally a *fixed cost* related with the existence of the resource and *variable costs* associated with its operation, for instance related to drivers, fuel, maintenance, roads, etc. In the latter, however, costs are normally of the *variable type* and depend on the volume of goods transported. The cost per unit of product transported decreases with the increase of volume.

Obviously, each *transport system* offers its own typical delivering speed and cargo characteristics that make it more adapted to certain situations. For instance, trucking is the most representative form of transport for finished products, since it is fast and even flexible to adapt to the frequent deliveries and small batches proposed by modern JIT. On the other hand, railways are mainly used for large batches of materials, especially for high value raw materials like chemicals, medicines and dangerous cargo, but also coal, food, machine spare parts, and even for large numbers of automobiles. The flexibility of the railway depends very much on its network type and on how it is operated. On the contrary, large amounts of low value materials, e.g. cement, sandstone, wood, iron, grain, crud, are often shipped through waterways like rivers, canals, the sea, etc. Being the slowest method of transport, the ship is nonetheless a very interesting option in terms of low costs for large batches. But pipelines are sometimes concurrent with ships and trains for continuous transport of chemical products, natural gas, oil, etc. And, finally, airfreights are used to achieve fast response on handling high value low weight products, from the spare parts for machinery to the transport of cloths and even flowers. This option is a trend nowadays, since some *3PL* operators extended their activity to the flight operation, like DHL, TPG/TNT, UPS, *Panalpina* and others. Airfreight is also the transport option used for achieving long distance JIT, and an essential service for world fashion suppliers like *Benetton* or ZARA, for instance.

2.2.3 Management levels and policies

As next figure (Fig. 2.4) shows, the management of a company is normally founded on a structure sustained by the following decision levels: *strategic*, *tactical* and *operational*. These levels define a kind of hierarchy of decision, with the *strategic* being the highest and the *operational* the lowest level.

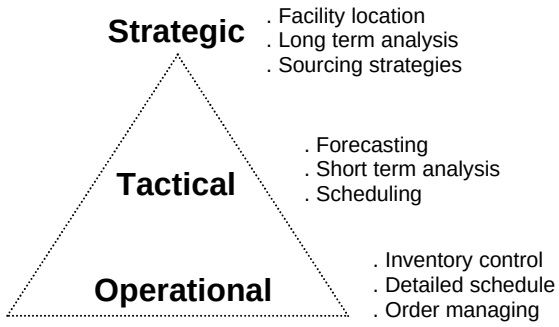

Fig. 2.4 The three management levels in a company

The ***operational level*** deals with the art of responding to the daily requests of the business, that can involve order managing, scheduling of detailed tasks, for instance, assembling, packaging, loading and unloading, as well as the calculation of delivery, cargo space optimization, etc., but, most of all, the management of the inventory by means of some inventory control policy. One can say that this is the level directly interfacing with reality.

On the other hand, the ***tactical level*** decides based on a more agglomerated view of the activity of the system, usually in the horizon of weeks, depending on the case, and based on historical records of the business activity. Decisions at this level are, therefore, founded on *forecasts* taken from short term analysis of the performance of processes. From such knowledge, monitoring of and quick adjustments in the scheduling of production can result, as well as in other activities like transportation, supply of materials, delivery plans, customer ratings, subcontracting, etc., within certain limits imposed by the upper level of strategic reasoning.

At the ***strategic level***, business activity is monitored and analysed in an even more agglomerated manner, in order to better adjust any decision to the long term objectives of the company. The horizon of interest can be of several months or even years, and data for analysis is frequently sampled monthly. The establishment of contracts with important suppliers, the calculation of the location of the facilities and their capacities, as well as the long-standing business decisions like expansion, change of products or technologies, etc., are the responsibility of this level. As in the previous case, decisions at this level are founded on *forecasts*.

An interesting picture about the kind of reports produced for *operational*, *tactical* and *strategic* purposes can be found in Schuelke (2001).

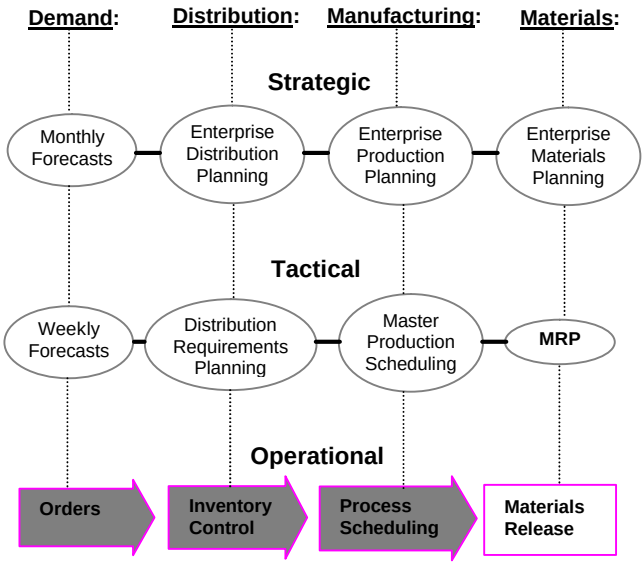

Fig. 2.5 Management functions associated with the *operational*, *tactical* and *strategic* levels

In a healthy company, such levels are in good coordination with each other through the exchange of data between sections and departments, as the diagram of figure 2.5,

inspired on Fox et al. (1993), purports to show[2].

For each management function, the activity at the *operational* level is restricted by certain *tactical* constraints, and, at the same time, *tactical* choices are confined and directed in accordance to *strategic* guidelines, in a manner that maintains the running of the business with the best possible harmony. *Tactical* and *strategic* levels can therefore be seen as "regulators" that create feedback in the *operational* level. Anyhow, shaped by *tactical* variables and these variables constrained by *strategic* factors, the *operational* level is the level where the real system runs. Operational data is based on the "real", while data in other levels is based on "forecasts".

This allow us to expect that modelling the Supply Chain from the *operational* level will lead us to obtaining "raw" output data which then can be "interpreted" not only *operationally* but also *tactically* or even *strategically*, depending on the period of time simulated and on the type of analysis made on such data. Besides, it is expected that *strategic* and *tactical* constraints can be indirectly imputed to the model on representing the configuration of each facility, by means of the definition of the policies of replenishment, production, distribution, etc., and, therefore, that studying different scenarios as *what-if* cases will also let us conclude about different practical situations.

The next chapter will expose in more detail this *bottom-up* modelling design, in which feedback will be injected in the model by means of adjusting operational policies after analysing the results of previous simulations. In such an approach, important replenishment policies will be derived from models known in the literature (see Hopp & Spearman, 2001, pp. 48) like the *Economical Order Quantity* (EOQ), the statistical inventory models (r, Q), (r, S), etc., but also considering some new trends like the KANBAN, used for achieving JIT, and the curious *BanKan*, an almost naïve method proposed by the author which establishes a kind of pipeline-like flow of materials. At the moment, however, let us pay some attention to the most relevant operational policies:

2.2.3.1 Inventory policies

Since the more important aspects of the supply policies usually belong to the strategic level (with the establishment of the suppliers and their contracts), the replenishment policy is practically what is left of the relations with suppliers at the operational level. Several methods for replenishment can be found described in any standard book of management, but in the aim of this work it seemed of more interest to expose such a problem in a lighter way, which will be done with the help of the scheme represented in figure 2.6. in this figure, *Q*, representing the inventory level, is usually expressed in *Stock Keeping Units* (SKU), a term referring to a specific product in inventory or in a catalogue (O.B., 2005); while the independent variable is the time (*t*), which is frequently expressed in days, weeks, months, etc. In practice, the SKU can also be related to the *palette* or even to the *container*, depending on the case, since

[2] As a curiosity, notice that there is not yet in this diagram a reference to *Enterprise Resource Planning* (ERP), but its structure already points in that direction.

these are dimensions more appropriate to transport.

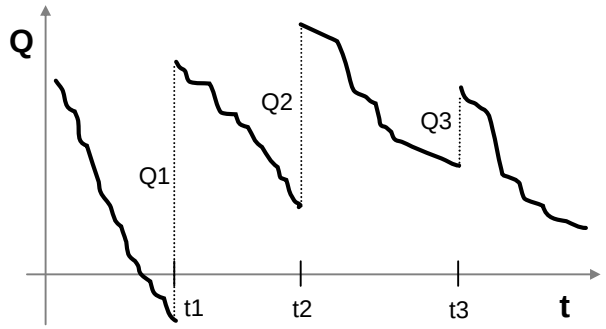

Fig. 2.6 General behaviour of an inventory system

From this figure, one can notice that at the instants *t1*, *t2* and *t3* the quantities *Q1*, *Q2* and *Q3* previously ordered to the supplier have arrived at the inventory, raising the stocks. However, at the same time, the inventory is continuously being "consumed" by the "irregular" demand from the clients. The problem of inventory management is precisely to answer the following two questions: (1) *which quantity (Q) to order*; (2) *when to place the order*. A good answer must ensure a good balance between the *costs of having the stock* in hand and the *costs of loosing clients* if stock is not available.

As can be found in the vast literature about the subject, the EOQ model for fixed lot sizes, the *Wagner-Whitin* procedure for dynamic lot sizes, and the statistical (r, Q) and (r, S) models are perhaps the most common mathematical bases to answer those questions, for inventory systems of *independent*[3] *demand*. In a fixed lot size, for instance, the quantity to order (*Q*) can be computed as:

$$Q = EOQ = \sqrt{\frac{2AD}{h}} \qquad (2.1)$$

where *A* is the cost of one order, or *setup cost*, *D* is the *total demand* in the period of interest, and *h* the *bank lending rate* in the same period, or *holding cost*.

On the other hand, the moment to place the order is when the inventory state will reach the *Reorder Point* (ROP) defined by:

$$r = ROP = DLT + ss \qquad (2.2)$$

where *DLT* represents the demand during the supplier's lead-time and *ss* a corrective amount of stock used to protect the system against *stockouts* (or negative stock levels). Figure 2.7 intents to represent this idea, where the moments of placing the orders are signalled as well as the lead-time (*LT*) of the supplier.

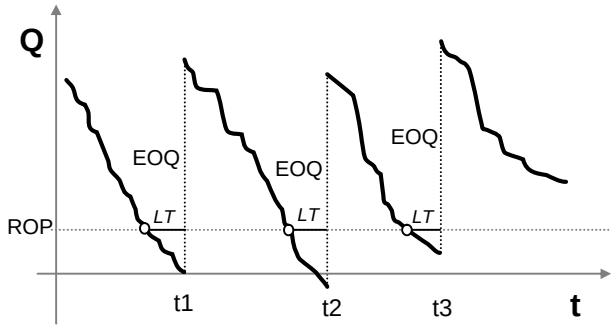

Fig. 2.7 General behaviour of a (r, Q) system with r = ROP and Q = EOQ, showing a little *stockout* at t = t2

This case can be considered a particular case of a more general (r, Q) model where *r* and *Q* could possibly be computed by other means, in order to better adapt not only to the lead-time variance but also to the variance of demand.

[3] Related to a product that is not a sub-product of any other product.

Apart from these, also the *Two-Bins* system and the KANBAN are methods frequently used to answer those same questions, as well as the *fixed-period* revision method[4], which simply establishes the revision of the inventory at fixed-periods of time. For systems of *dependent demand*, however, MRP is also widely used.

Detailed information about this can be found in Heizer & Render (2001), Hopp & Spearman (2001) or, written in Portuguese, Gonçalves (1999).

2.2.3.2 Production policies

Production policies are primarily strategic issues, since they shape the production processes running at the company, or factory, from the beginning. The definition of business itself is frequently based on how the process of fabrication is organized, in terms of resources, layout and dynamics. A different *variety* of products, *volume* of production and level of *utilization* of resources will result from different organizational alternatives. Of course, these kinds of issues must be faced and resolved before the installation of the company on the ground, although agile organizations struggle to achieve the ability of changing these strategies as required for reacting to the turbulence observed in markets. It is supposed, however, that the modeler of a Supply Chain may also use this kind of information to configure the facilities at the model. This is a simple way of inserting strategic feedback into an operational model. Once this information is known, the process of production can be established, as well as the way the factory will react to the orders coming from its clients.

[4] Also called *News Vendor Model*, since newspapers arrive at fixed periods of time (daily, weekly, etc.).

Most production systems are designed in accordance to one of the following four (4) process strategies (see Heizer & Render, 2001, pp. 234):

(1) **Process-focus**: in this approach, several basic *tasks* (or *jobs*) of fabrication are made available as a flexible option to allow the making of diverse products. For instance, in a *Job-Shop* facility, different kinds of machines ensure that different processes of fabrication can run almost in parallel (Fig. 2.8). The challenge is the scheduling of the *tasks* to optimize the utilization of the resources, while ensuring the intended variety of products output. Carpentry is a simple but didactic example pointed out by Heizer & Render (2001). Although several different objects of furniture can result from such a facility, the fact is that all of them have been made with the same kind of tools or machines, and, probably, with several employees working on different processes.

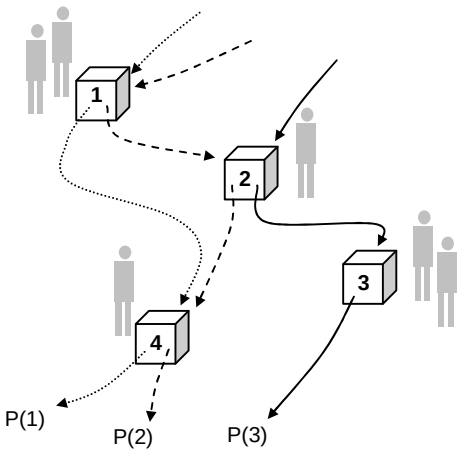

Fig. 2.8 Process-focus fabrication (Job-Shop) producing products P(1), P(2) and P(3) based on four different tasks

These companies frequently try to work on a *Make-to-Order* (MTO) basis, even if often maintaining low stocks, just in case…

This option is typical for low-volume of outputs, low-utilization of resources, and a high-variety of products.

To model and simulate these sorts of processes is usually more interesting to the manufacturer than to the Supply Chain manager, since it is a typical problem of production scheduling. Besides, it is also usual that in simulating the Supply Chain products are considered independently manufactured, therefore, to each of them an independent process of production is considered as a model. Of course, that is an *equivalent* model of the real process, which must be configured in accordance with the results obtained in practice.

(2) **Repetitive-focus**: in this approach, the fabrication basically consists of a sequence of operations of assemblage that, at the end of the process, gives rise to one or more products. This is very much the type like that of the assembly-line of Henry Ford. In each place of work, or work cell, one (or more) operation is completed before sending the product to the next work cell (see Fig. 2.9). Usually, these operations consist of the mounting of specific modules (*mj*) on the skeleton of the product that is being filled, from place to place.

The variety of products will result mainly from the number of combinations available with the modules and, in certain cases, also by the combination of colours, for instance. This was the scheme responsible for the *Mass Production Era* (1910-1980), used to build automobiles, appliances, and almost all of the items produced for the public in industrial volumes, but nowadays it is also used to achieve JIT manufacturing, since automation processes and modular design are now much more efficient than in those times.

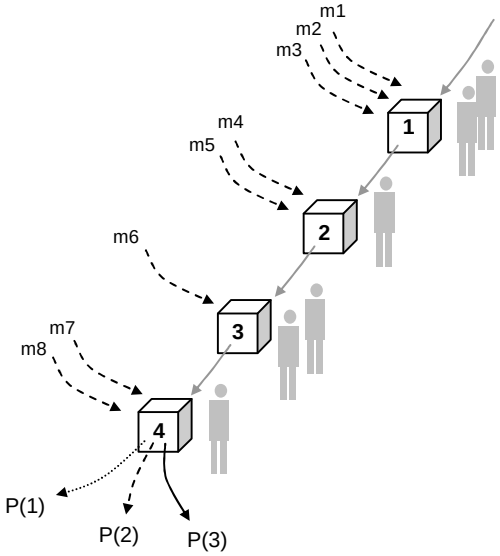

Fig. 2.9 Repetitive-focus manufacturing (assembly line) with output products P(1), P(2) and P(3) mounted up on four different assembling cells

Companies using this type of approach frequently work on an *Assemble-to-Order* (ATO) basis, typical of an average/high-volume, high-utilization, and low/average-variety. Modelling this type of processes for Supply Chain simulation purposes may be achieved by attributing to each product a *process-time* function, probably retrieved from a distribution of values.

(3) **Product-focus**: also named *continuous production*. In this kind of manufacturing, the final products typically result from a sequence of transformations applied to raw materials, usually by means of certain chemical or thermodynamic processes, that frequently finish in a process of shaping. It is the case in the production of paper, steel, oil, etc., which require processes with long runs and high energy consumption, and therefore a continuous operation to

optimize costs. In terms of process diagram, however, this sort of production line is similar to the assembly-line of figure 2.9, provided that the modules are replaced by operations of transformation on the raw materials and, perhaps, adding a final shaping. These systems are organized around a specific product (or class of products), usually run on a *Make-to-Stock* (MTS) basis, and the production is adjusted to the forecasts by means of speeding up or slowing down the production line. These are considered high-volume, high-utilization, low-variety manufacturing processes.

(4) **Mass-customization**: this is the trend for customer servicing of our time, once the option is possible. Products are made on a kind of a fast and flexible assembly-line in which assembling times are made the shortest possible, compared to other industrial processes, through the usage of a large variety of modules with flexible design and a superior and flexible scheduling of tasks. Fast and synchronized operations in the delivery to the clients terminate the process (Fig. 2.10).

This approach is predominantly used for fabricating computers[5], appliances, pizzas, mobile-phones, etc., where assembling times are naturally short compared to the time the customer is prepared to wait for the product. Lately, however, even books and automobiles are delivered in a JIT fashion using this manufacturing approach. Clients must obviously be prepared to wait for an automobile much longer than for a pizza, but compared to what was happening in previous decades, total *lead-times* have been dramatically shortened. A high variety of versions of the same *base product* allow adjusting the product in a way to fit into a wide spectrum of customer preferences. Customers can even sometimes use *e-Commerce* to virtually assemble their own versions from modules presented in an *Internet* catalogue, for instance, and then place their orders directly to the company. Examples of this practice are VW, Fujitsu-Siemens, Motorola, among others, but with the current expansion of *e-Commerce*, more and more companies seem tempted to follow such a trend.

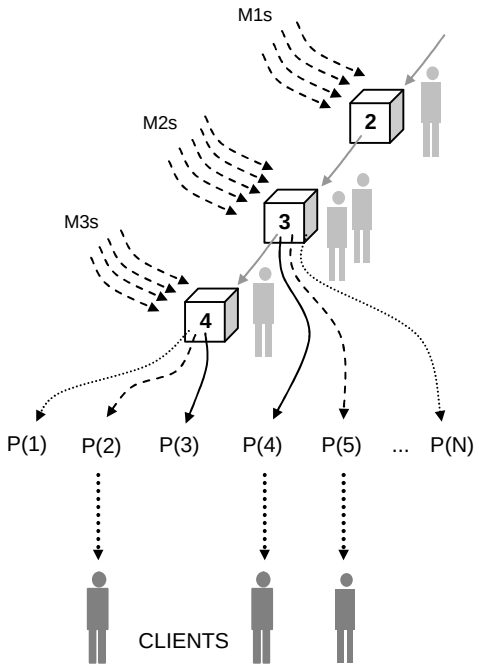

Fig. 2.10 Mass-customization of N products in three different assembling cells, using a large variety of modules

This kind of production is normally for average/high-volume, high-utilization, high-variety, and many times proposed for achieving JIT performances.

On concluding this section, in the figure 2.11 these four types of manufacturing are compared in terms of *volume*, *variety* and

[5] *DELL Computer* was probably the first company using this policy of production in order to deliver personalized (customized) computers to the door of its clients.

utilization of resources. Notice that the *utilization* is represented by the *area* of the circles. As it can be understood from the figure, the choice of the appropriate type of manufacturing is highly dependent on the type of business run by the company.

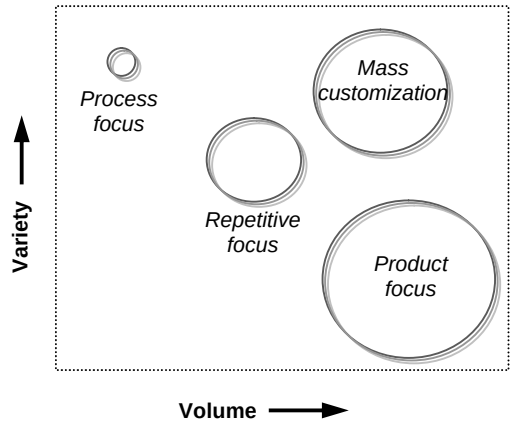

Fig. 2.11 Production policies in terms of *volume*, *variety* and *utilization* of resources (area of the circle)

2.2.3.3 Distribution policies

Distribution is in itself an art within Logistics. Military and 3PL companies are masters of this art. In the 1970s, with the eyes focused on the factory as the "centre of the universe", the *distribution* was considered the *out-bound logistics*, which is associated to the flow of materials going out the factory (the inverse flow, from the suppliers, was called *in-bound logistics*) plus the *delivery*, which was considered the last step of the *distribution* process, that is, the transportation of goods at the end of the chain to meet ultimate customers (see Eilon et al., 1971).

Today, with the eyes looking into every element of the Supply Chain with the same level of attention, the idea of *distribution* frequently absorbs the concept of *delivery* as it is a more generalist term. In addition to that, factories (or certain kinds of factories) are already *delivering* directly to its customers, what also contributes to this effect of absorption in nomenclature. In Heizer & Render (2001), for instance, the term *delivery* does not even appear listed in the general index.

We may therefore consider that in general the term *distribution* is related with the direct or indirect flow of materials from the facility to its customers. In that sense, a system of distribution usually has to face the two classical problems:

(1) the establishment of the number and the location of facilities in the distribution network, which has to do with the design of the network, constrained by some cost analysis usually captured from long-period plans (of several years, for example): consequently a *strategic* issue; and

(2) the management of the transportation through the network, which is related to the handling of materials, vehicle routing and scheduling, on the way to serving customers and satisfy demand: a *tactical* or even *operational* issue.

The first problem is normally addressed by representing the distribution system as a mathematical problem of a network of nodes and arcs where the sum of a group of weighted distances connecting the facilities to its customers is to be minimized, as shown in the next figure (Fig. 2.12). This is seen as being more or less equivalent to the total costs minimization. The facilities are considered as simple numbers (capacity for suppliers, demand for customers) while distances are weighted by the respective costs of transportation.

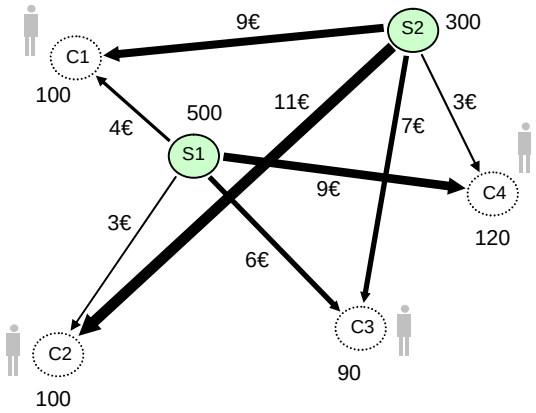

Fig. 2.12 Example of a classical facility location problem, with two suppliers serving four customers

Linear programming methods, numeric analysis, analytic and heuristic approaches and even electrical analogue[6] models form part of the heritage of techniques used to solve this kind of problem. But also hybrid techniques and statistic simulation started to be used in the 1990s with the advance of computer into the field of Logistics.

Guedes (1994), for example, is the author of a remarkable work on this subject in which the facilities are modelled as "black boxes" characterized by cost functions independent of time (static), that he calls "key drivers", with the lead times assumed previously known. STRATOVISION, the *Windows* application intended for *Logistics Strategy Planning* (LSP) was perhaps one of the most important results of his work, where the analyst is also expected to make use of "what-if" scenarios from which other empirical conclusions can be obtained.

To the contrary, dynamic simulation approaches (where all processes are time dependent) are rarely used for boarding these kinds of problems, since the development of such models is much more time consuming and demanding of programming knowledge. These problems, in themselves, being of the strategic type, already use previously "filtered" values frequently resulting from averages computed from long periods of observed or estimated operations, what extremely simplifies the cases and makes the dynamic approach too complex to be useful. That explains, to some extent, why dynamic simulation is used more frequently by consultancy experts, who in principle have the time and the resources available for in-depth investigation of problems, or, if integrated in the company, by the people at tactical or operational levels, where the complexity of the problems is experienced *in loco* and the need for detailed simulation approaches gets evident.

About the second issue, that is, the management of the transportation through the network, one can say that in the perspective of the tactical and operational analysts, the location of facilities and the topology of the network of transportation are just a part of the initial conditions of the problem. The problem in itself is now the management in time of the flow of materials, and so, the management of inventories and of transport resources. At these levels, the need for answering to new questions emerges: (1) when to deliver; (2) how much to deliver; (3) how to deliver[7].

Obviously, the answers to the first two questions strongly depend on the availability of transport resources (vehicles), type of

[6] Hitchings (1969), for example, considered the distance from the facility to the customer proportional to the length of a wired resistor, in his electric analogue modelling of the problem.

[7] From now on the term "delivery" will be used in its more general meaning of "taking to the client".

optimization of cargo space, due dates, priorities, choice of integer or factionary delivery, etc., giving rise to a problem of volume handling and scheduling. The short-term scheduling usually found in operative ambiences can easily turn so intense that one may consider that in those instances management really meets the art of improvisation.

The answer to the last question is more about the choice of transport modes (train, truck, etc., usually pre-selected by strategic directives), but also about the routing of the vehicle to the customer, the kind of delivery (multi-drop or single-drop), as well as the type of fleet to use (own or hired), for example.

In operational ambiences of independent demand, the choice of the response to a specific order defines, often from a variety of options, the particular distribution policy adopted. However, in cases of dependent demand, *Distribution Resource Planning* (DRP) is commonly implemented. This is a kind of MRP adapted to *distribution*, which uses the demand forecasts at the retailer level and further computes the dependent demands upstream in the supply chain facilities. From that, the transportation schedule is adjusted so that the materials arrive precisely when needed. Some authors consider this a PULL system.

A large number of tools for managing distribution at the operational and tactical levels is still based on spreadsheets, since these are extremely flexible and easy to handle computer applications. In many cases, they can be associated with *Gantt Charts* as visual aids for scheduling and loading, but with the recent proliferation of dynamic simulation this technique also started to be used by some enterprises. As we have seen in the last chapter, simulation is even a strong component of most ERP systems. Nevertheless, since dynamic simulation models continue to be complex, time consuming on development, exigent concerning the simulation skills, and, on the other hand, its usage rate is normally low, these sort of approach continues to be more used either by powerful companies or by private consultants. In its headquarters of Munich, BMW was recently using a dynamic simulation environment know as *eM-Plant* to support and optimize car production, for example. Many other strong companies use this sort of approach, but most of the others frequently restrict the usage of simulation to strategic proposes, by means of static modelling.

2.2.4 Traditional metrics and flexibility

An issue that continuously remains on the agenda is the metrics by which managers and researchers quantify the performance of Supply Chain systems. The natural evolution has been viewed from below, where the elements of a distribution network were considered independently (and using single company metrics), to an upper point of view, where those elements start to be seen as a new kind of super-entity with the same global final goals, giving rise to the concept of a network of shared and interdependent motivations, presently known as Supply Chain. This means that the previous metrics of a “node” were expected to evolve to metrics of a collection of “nodes” linked to each other through a complex “network”.

In reality, however, till now this was not what happened. The concept of Supply Chain is widely understood and practiced, but due to problems of complexity and *trust* such global metrics do not exist yet. Maybe one can conclude that markets, as we run them, are, by nature, perverse, and that no other law can superimpose the rudimental law of the strongest and fittest. In a certain way, it seems that there is a wish to turn those systems more transparent and "clean", but in fact there is also the danger that such a new model would become too fragile and give additional chances to some "partners" to take advantage by continuing to pervert the system in the background. There is, for that reason, a generalised caution about the cooperation and the transparency that a new level of metrics would involve, if implemented.

Another problem is the increase in complexity that could be expected from such a new kind of metrics, since it would mean a jump from a single company metrics to an upper level of metrics of a complex network topology and even overlapping structures.

Instead of going that way, as Lambert & Pohlen (2001) well emphasise, managers continue to apply old single enterprise metrics and, through such measures, look at the Supply Chain as simple agglomerates of companies, where of course the law of the most powerful directly or indirectly dominates.

Trials to advance towards new directions are, nevertheless, being made by some researchers, from which one can point out Lambert & Pohlen (2001), who argue that inventory *turnover* and other common metrics are inadequate for the evaluation of these networks of multiple companies, and therefore propose to translate performance into a financial measure related with the added value of each Supply Chain branch based on profit/loss statements; as well as, among others, the author of the present text, in his trial to establish a simulation based methodology to measure the Supply Chain flexibility (Feliz-Teixeira & Brito, 2004). Obviously, the metrics considered in the simulation approach presented in the next chapter also carry in themselves such anachronism.

Despite the fact there is a large number of enterprise measures that can be used in practice, experts normally recommend that managers should choose only few of them to represent the performance of the company, for many times using a large number of metrics introduces much more confusion than benefits. The following ones have been chosen and are automatically computed along the period of simulation:

2.2.4.1 Facility related metrics

These metrics are used specifically for describing the performance of the facility. Some of them are financial, and some are physical or operational. Financial measures usually lead to more useful indexes because most of the time they can easily be applied to multiple product systems.

• **Global costs:** is considered the sum of the overall components of the fixed and the variable costs, usually calculated at the end of the period under study. This value gives the manager the idea of the global amount of money needed for maintaining and running the company. Costs will be described later, in the next chapter.

• **Total variable costs:** is considered the sum of the various types of variable costs; therefore it is also a component of the global costs. It gives an idea of how much it costs to *operate* the company.

• **Income:** is the net amount of money earned by the company in the exercise of its activity in a certain period of time. It is also measured in the units of currency (€).

• **Cash flow:** is the difference between the incomes and the total variable costs. It is normally calculated at the end of a certain period of activity. It can be seen as the financial compensation for the exercise.

• **Turnover:** also known as inventory turns, is a measure of the frequency in which the materials in the inventory are renewed in one year. This is usually seen as an indication of the operational activity of the company. The higher the turnover, the less the company is stagnated. This is basically computed as:

$$Turnover = \frac{GoodsSoldInOneYear}{AverageInventoryInOneYear} \qquad (2.3)$$

Expected turnovers can vary significantly in practice, since they are very much dependent on the kind of process and business in question. But, as one can easily perceive, this metric gets inadequate when many products are considered in inventory. For that reason, a more general financial version of it is usually applied in practice, that is, in one year:

$$Turnover = \frac{CostsOfGoodsSold}{InventoryValueOnHand} \qquad (2.4)$$

Average turnover rates of our times are around 3 to 6 (PRTM, 2000), but many companies work with around 2 while some others can even reach the two digits, when using JIT, as is the case of *DELL Computer*, for example, with turnover rates around 30-40 times/year (Vijayan, 2001).

• **Service level:** is a very common measure that gives the idea of the extent in which the company is able to fully and quickly satisfy its customers, in terms of answering their orders. There are some variants of this measure, but in general one can compute the service level as the ratio between the material directly served from stock and the total quantity served, that is:

$$ServiceLevel = \frac{OrdersServedFromStock}{TotalOrdersServed} \qquad (2.5)$$

This is usually presented as a percentage.

• **Stockout ratio:** is another measure of the same kind of service level, but based on a complementary criterion. It gives the idea of how much the customers were not served. It is, therefore, a failure measure:

$$StockoutRatio = \frac{OrdersNotServed}{TotalOrdersServed} \qquad (2.6)$$

• **Customer Satisfaction**: this is a relatively new measure (Christopher, 1994) which results from the fact that the customer is nowadays considered the most important element of the Supply Chain. Therefore, it is reasonable to consider a measure centred on the customer. But this metric makes use of multi-dimensional criteria, and most of those criteria are subjective, as well as dependent on the type of business in question (see Cacioppo, 2000), making it

difficult to use in quantitative simulation systems.

To surpass the problem, it was decided to consider only a one-dimensional perspective of customer satisfaction which is based on the relative speed by which the customer receives the materials ordered. Considering ΔTo the customer expected lead time from its supplier, and ΔT the supplier actual lead time, this measure is computed as:

$$CustomerSatisfaction = 100 \times \min(1, \frac{\Delta T_o}{\Delta T}) \qquad (2.7)$$

And used as a percentage.

Rigidity: this is an entirely new concept applied to management. It was developed by the author during his Ph.D. studies and published for the first time in Paris in 2004 (see Feliz-Teixeira & Brito, 2004). Briefly described, it represents the energy spent on imbalance while the company is reacting to a sudden step on the demand. This is considered the inverse of flexibility and is measured in SKU/day in imbalance. In figure 2.13 is represented the basic idea behind this measure.

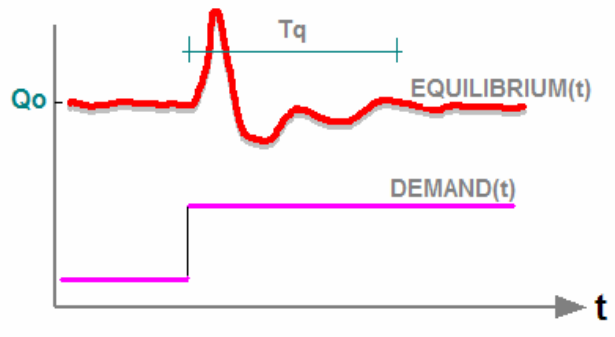

Fig. 2.13 Using demand steps to measure *rigidity*

The system, usually the inventory, needs a certain time (*Tq*) to recover from the equilibrium lost due the sudden step on the demand. During this imbalance period, the inventory instability is compared with the optimum response, in which the equilibrium would never be lost. This "excess" or "shortage" of SKUs due to imbalance is considered a measure of the rigidity of the system. This concept of rigidity made it possible to represent the overall rigidity of a Supply Chain as a matrix of the individual rigidities of the companies, as a true network metric. Still, to get this measure it is necessary to simulate the system.

Some other quantitative metrics exist to represent *flexibility*, property seen as a *potential* behaviour and not an *operating* behaviour (Slack, 1983), but such metrics have been developed for manufacturing systems and not properly for the Supply Chain. Even if some of them can be adapted to this kind of systems, they will not be used in our simulation approach for the reasons explained shortly.

Many other metrics for resources and outputs could be mentioned, like the *Return-On-Investment* (ROI), *Cash-to-Cash Cycle Time*, *Backorder Level*, *Supply Chain Cycle Time*, *Perfect Order Measurement*, etc., but it was considered preferable for this work to reduce the analysis to only some of them. Nevertheless, it is also worth referring to the *Balanced Scorecard* (Kaplan & Norton, 1992) as an important metric used for Supply Chain, even if we will also not consider it since it deals with some aspects of the company that are not of our concern, like *forecast errors*, *certification*, *hours of training personnel*, etc. Basically, this is a measure in which a group of appropriate measures are chosen to align

the company to its strategic objectives.

2.2.4.2 Fleet related metrics

It is also important to give some details concerning the metrics for transportation processes. As in the previous case, a significant number of different measures can be applied in practice, but for our intents we will consider only three of them that already give a reasonable quantitative idea on how the diverse vehicles of the fleet are being used. Notice that in our approach, to each facility in the Supply Chain is considered assigned a fleet of vehicles, which can be owned or not by the facility. Diverse kinds of vehicles may compose the fleet. Each referred metric will be applied to each of those vehicles:

- **Usage:** this represents the fraction of time the vehicle has been operating in a certain period of activity of the company. It is computed as a percentage, and gives an idea of the activity of that vehicle during that period.

- **Occupancy:** this metric represents the average vehicle occupancy with products during its time of operation. It is expressed as a percentage of the full capacity of the vehicle. This can be a useful measure of how the vehicle volume is being optimized.

- **Total time of operation:** measures, in *hours*, the accumulated time of operation of the vehicle. This can be useful to estimate maintenance costs or to help in the project of maintenance processes, for example.

Apart from these transportation metrics, we also keep in our simulation approach a continuous measure of the average lead times from suppliers as well as the average time of delivery of the facility, that gives the user some more interesting information about the system being simulated.

2.2.4.3 The emphasis on flexibility

Nowadays there is a great interest in concepts like flexibility and agility, even if more recently people seem to talk slightly less about them. Despite such concepts having been well absorbed even by Supply Chain management, the same cannot be said about their metrics, either due to a lack of general acceptance (the case of *flexibility*) or simply because they do not exist, as in the case of *agility*.

Flexibility, coming from manufacturing, is usually understood as the ability to quickly adapt production to different product mixes or new kinds of products, and is frequently associated with alternative ways of doing things. Once one way fails or is occupied, other ways are available. In fact, however, it is more than that, and some quantitative metrics have been proposed (see Beamon, 1999). Volume flexibility, for instance, measures the ability to respond to demand variations in volume; delivery flexibility, deals with the ability to manage delivery due dates; mix flexibility is about the ability of producing multiple products; and new product flexibility the ability to handle production of short life cycle products. As Beamon (1999) notices, some of these types could also be applied to Supply Chain, specially volume and delivery flexibilities, which are similarly computed.

The problem we find in these metrics is that they are computed statically, without the need of any dynamic computational process. For instance, the volume flexibility is seen as the "*range of volumes in which*

the organization can run profitably" (Sethi & Sethi, 1990), and is proposed to be calculated by representing the overall demand in the form of a *Normal* distribution function and, knowing the *minimum* and *maximum* limits of profitable demand, computing the ratio of two areas under such function. This can be done with any spreadsheet. And, as such *limits* probably result from enterprise analyses that are beyond any Supply Chain simulation results, we could not consider this metrics in our approach. Figure 2.14 is intended to help in clarifying such measures. The curve represents the *Normal* distribution of the demand.

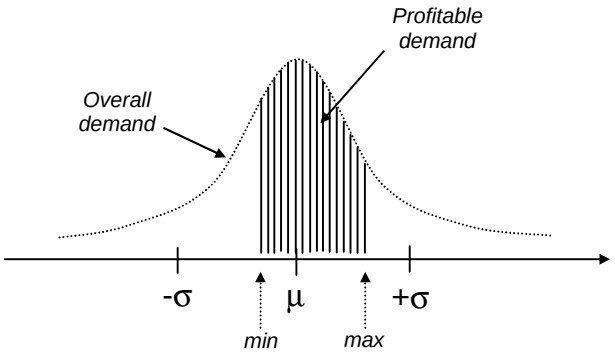

Fig. 2.14 How to compute *volume flexibility*, as proposed by (Sethi & Sethi, 1990)

Of course, enough demand samples must be considered for that m can represent the average demand during a certain period of time as well as s the associated standard deviation. Flexibility will then be simply given by the ratio of the two areas:

$$Flexibility = \frac{profitableDemand}{overallDemand} \qquad (2.8)$$

Agility, on the other hand, intends to be a wider concept, dealing with the ability of adapting to a "turbulent" outside world, not only concerning production, but as well other enterprise functions like distribution, procurement, etc., since also liaisons among partners in the Supply Chain can suddenly be established or broken. The term agility is normally applied to Supply Chain. Anyhow, if one asks a top manager in a top school of management what agility is, for example, the answer stays consistently vague and rather evasive. The impression one gets is that there is a tendency for talking about certain concepts in the same way sellers talk about love: "*love is… being this or that*", and stamp it on a T-Shirt just with the intent to sell the T-Shirt.

From the present discussion one can conclude that it is still hard, if not impossible, to introduce practical concepts of flexibility and agility in a Supply Chain simulator. The development of the idea of rigidity was, in this sense, a trial to achieve such a goal and, at the same time, a modest contribution for bringing management a little closer to science, in a tribute to Frederick Taylor, Henry Ford and Taiichi Ohno.

2.2.5 A step up to holistic metrics?

As we have noticed, there are presently very few notes on the kind of metrics that could be reliable and of practical interest when applied to complex systems. These systems are often based on intricate structures where a high number of entities interact with each other. Metrics are there for appropriately characterizing the nodes or parts of such structures, or small amounts of them, but when the intent is a measure for the complete structure either they fail

or appear to be too simplistic. That is certainly a good reason for modelling those cases using a strategic point of view, as in doing so the complexity is reduced *a priori*.

But when more complex representations are needed, this issue seems sometimes also related to a conflict between different scientific approaches: the classical western reductionism, of anglo-saxonic inspiration, which believes the best approach is to break the system into small parts and understand, model or control those parts separately and then join them together, therefore looking at the world in an individualist way; and a more holistic approach, a vision slowly spreading and largely inspired by oriental cultures, which considers that each part of the system must be seen together with the whole and not in isolation, and therefore locates the tone in how the interactions between such parts contribute to the whole behaviour. Hopp & Spearman (2001, pp.16), for instance, comment about this saying that "*too much emphasis on individual components can lead to a loss of perspective for the overall system*".

A significant number of authors defend this opinion, pointing out the importance of developing a more holistic point of view to interpret and study systems behaviour, in a way that analysis can maintain enough fidelity to the system as a whole. As Tranouez et al. (2003), who apply simulation to ecosystems, would say: a complex system is more than the simple collection of its elements.

In management science, for instance, the western approach frequently generates difficulties at the interfaces of elements, typically of inventory or communication type. On the other hand, as JIT gives better emphasis to the relations and interactions and is continuously improving, the overall movements tend to be more harmonious. JIT already looks at systems in a certain holistic way.

But, what concerning metrics? How can one measure such a high number of states typically found in complex systems, in order to effectively retrieve from them some sort of useful information?

As a metric is a characterization, we could think that maybe the modern *Data Mining* (DM) techniques could be extensively applied, for instance. These techniques use decision trees and other algorithms to discover hidden patterns in huge amounts of data, and are nowadays applied to almost any problem based on extensive data records, for instance, in *e-Commerce* for customer profile monitoring, in genetics research, in fraud detection, credit risk analysis, etc., and even for suspected "terrorist" detection (see Edelstein, 2001; Edelstein, 2003). However, they often imply the usage of high performance computers, sometimes with parallel processors, as well as huge computational resources to analyse *GBytes* or even *TBytes* of data. They are useful when any single record of data can be precious for the future result, and thus when all data *must* be analysed.

On the other hand, in many practical simulations a significant amount of data is not significant for the final conclusions, the simulation process is in itself a filter, and therefore such data may well be ignored in the outputs, even if it could have been essential to ensure the detailed simulation process to run. In the perspective of the author, maybe there is a way that could deserve some future attention: the idea is

to filter such data during the simulation execution and, at the same time, to turn the measures probabilistic by using an approach somehow inspired by *Wave Fourier Analysis* or *Quantum Mechanics*. That is, to represent the overall system state (ψ) in terms of certain base functions (ψ_i), and then to measure the probabilities (α_i) associated with each of these functions. The interesting aspect of this is that each base state function (ψ_i) could even be arbitrarily chosen by the analyst, and the probabilities (α_i) easily computed during the simulation. Final results would then be summarised in some expression of the form:

$$\psi = \alpha_1 \psi_1 + \alpha_2 \psi_2 + \ldots \alpha_j \psi_j + \ldots \alpha_n \psi_n \qquad (2.9)$$

which could be interpreted as: there is a probability of α_1 that the system will be found in the state ψ_1, a probability of α_2 that the system will be found in the state ψ_2, etc. This would be the final measure of the system, in a sort of characterization of expectations under certain conditions.

Although this matter could in itself give rise to a new research project, which is not part of the aim of the present text, a little more about this will be said in the last chapter, when referring to future research.

2.3 Supply Chain modelling

Since Supply Chain management became one of the most vital management issues, the various schools of simulation have naturally showed a great interest in the matter. A wide range of models for the Supply Chain is therefore announced in scientific journals, magazines and the *Web*, built with almost all the technologies available to represent and simulate systems. From the old but always powerful *System Dynamics* to the modern tendencies of *Web-based*, *parallel* and *distributed* simulation, passing by several other analytical methods and proposals, almost all the simulation approaches have aimed to enter this field.

As usual, each school faces the new challenge by proposing to apply its own resources and approaches. The result is a wide spread number of possibilities, which have principally been developed based on the approaches shown in figure 2.15. And this is only a part of the big simulation tree presented in the previous chapter (see figure 1.23).

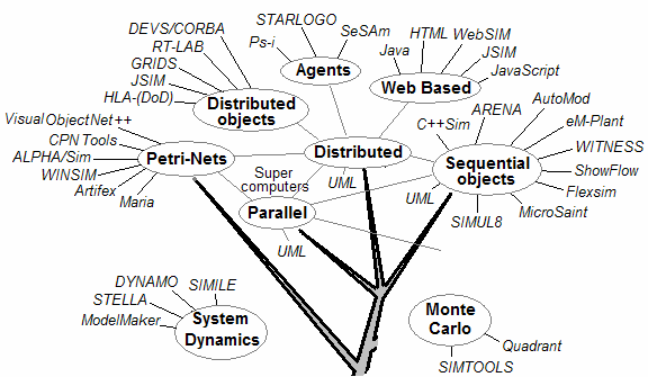

Fig. 2.15 Approaches and tools more commonly used for Supply Chain modelling and simulation

From this, one can already have an idea about what to expect in terms of Supply Chain modelling proposals. Except for the cases of *System Dynamics* and *Monte Carlo*, and the analytic methods not shown here, all the other approaches are primarily used for discrete modelling. *Monte Carlo* (statistic simulation) is popular in strategic Supply Chain models, while the other approaches are also suitable for dynamic representation and, therefore, they are also useful for tactical and operational modelling. We will present a brief overview of each of these tendencies.

2.3.1 System Dynamics

System Dynamics, for instance, is known to represent a system as a set of continuous equations interacting in loops of feedback where variables act and are represented through *causal diagrams*. The model is visualized as a casual diagram made of several feedback loops that define the dynamics of the system. For instance, Mason-Jones et al. (1997) use this modelling approach to study the *Bullwhip Effect* in the well known "*Beer Game*" linear Supply Chain, Minegishi & Thiel (2000) use it to model a small poultry distribution system, and Bruniaux & Pierreval (1999) for testing various scenarios of a Supply Chain, among other cases. A very simple causal diagram is represented in the next figure (Fig. 2.16), which was inspired by a simple layoff analysis described by Soderquist & Harris (2001).

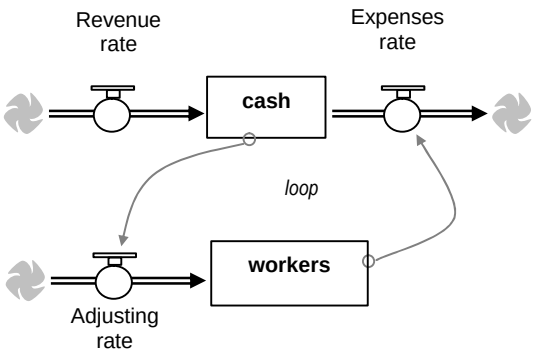

Fig. 2.16 Example of a causal diagram of a model for simple layoff analysis using *System Dynamics*

As it is easy to perceive, the system is modelled in terms of "recipients" and in/out "flow rates", as in a fluid like description of its dynamics. Variables are seen as "valves" that control those flows, thus controlling the behaviour of the model. Notice that loops naturally emerge from an approach like this. Many of the symbols used in this figure are reduced to the essential in practice and model diagrams may appear as almost a sequence of loops engaged with one another.

2.3.2 Petri nets

Petri nets is another important formalism born around the same time as *Systems Dynamics*, but in a different school. It can also be used for Supply Chain modelling. Davidrajuh (2000), for example, uses this approach to address issues related to agile Supply Chains, while Nikolic et al. (2004) have also studied the "*Beer Game*" and the *Bullwhip Effect*. Good attention can also be given to the more formal contribution of Landeghem & Bobeanu (2003) concerning this approach.

Petri nets represent systems as sets of *places* (symbolized by circles) connected by *arcs* to *transitions* (symbolized by rectangles or bars) which via *arcs* send output to other *places* (Fig. 2.17).

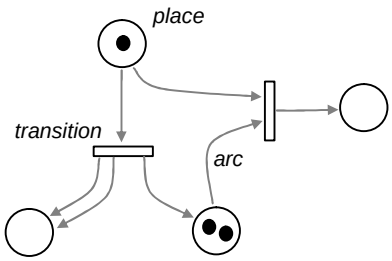

Fig. 2.17 A simple example of a *Petri net,* with some *tokens* inside some *places*

The *places* represent states, while the *transitions* represent events. So, this is a sort of *state transitions diagram*. The entities that will flow through this structure are called *tokens*. A very interesting aspect of *Petri nets* is that it is not only a formalism to represent systems but also

with which systems can be graphically modelled. This is simply achieved by representing the *tokens* as dots inside the *places* and moving them in accordance to the following basic execution rules: (1) a *transition* is "active" when each of its input *places* contains a *token*; (2) each active *transition* in the diagram is "fired" by removing 1 *token* from each input *place* and generating one in each output *place*.

The dynamics of the model emerge from these rules, and therefore the analyst can even form an idea of how the model will behave while building it. This is usually pointed out as an interesting property for debugging of *Petri nets*. Actually, based on this, it is very simple to deduce that the previous system will evolve to the state represented in the next figure (Fig. 2.18), as *transitions* can "fire"[8] simultaneously:

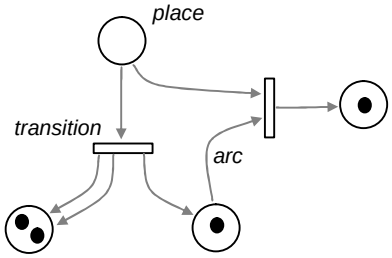

Fig. 2.18 The next state of the previous example

2.3.3 Sequential objects

State transition diagrams is another commonly adopt method for representing discrete models (similar to the one shown in figure 2.19, related with a simple inventory output process).

In this kind of representation each active element in the system is modelled as a set of *live states* (rectangles) and *dead states* (ellipsis) linked together by *arcs*[9]. Since in discrete simulation *live states* frequently are referred to as *activities*, these diagrams can also be called *activity diagrams* or even *activity cycle diagrams*. From time to time, expressions like *live activities* and *dead activities* are employed as well.

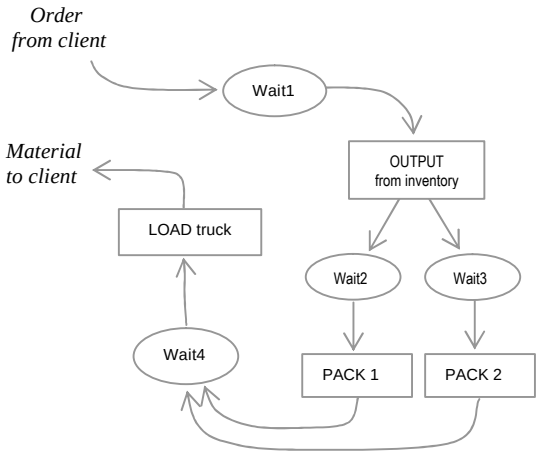

Fig. 2.19 State transitions diagram of a simple inventory output process with two packing lines and one output bay

The dynamics of the system emerge from the sequence of actions carried out by the active elements of the model, which are called *entities*. These actions will create the evolution of the system in time.

Thus, a system is seen as a collection of *entities* which are performing their *activity cycles* and interacting with other *entities* by means of common *activities*. The next figure (Fig. 2.20) better illustrates this idea, based on the same inventory process.

Notice that at least four *entities* could be detected in the present case: the *client order*, the *inventory operator*, the *packing operator*, and the *operator to load the truck*. There is, therefore, the same number

[8] The term "fire" is normally used to mean "action".

[9] Live states represent states which duration can previously be known or computed, while dead states are those where this cannot be done, as waiting states and queues, for example.

of *activity cycles* which then intersect at common *live activities*, as it is shown in the figure. Notice that we have chosen the symbol of a person to represent *entities* in order to suggest they carry some sort of "personality". *Activity diagrams* can then be "translated" to computer code, usually by means of one of the classical modelling paradigms: *process-interaction*, *activity-scanning*, *event-scheduling* or *three-phase*.

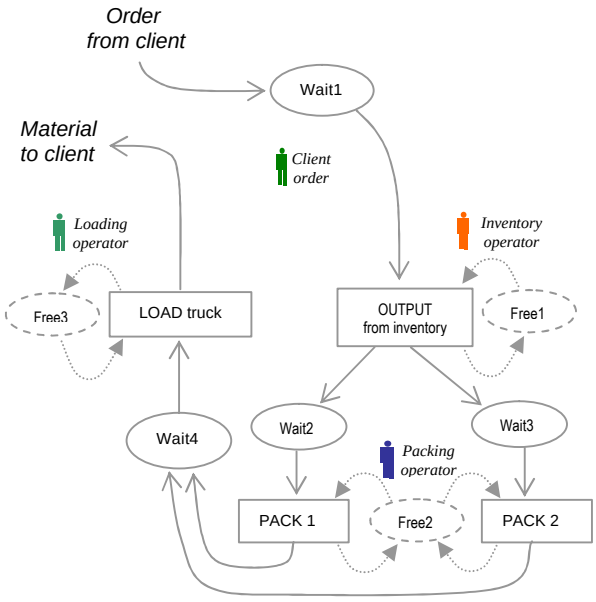

Fig. 2.20 The entire activity cycle diagram for the same inventory output process, showing the entities of the system

This kind of representation is frequently used to implement entities as classes of objects in *Object Oriented Programming* (OOP) simulation tools. The entire model is then a computer application that "contains" those objects, as exemplified in figure 2.21. Such technology is sometimes referred to as *sequential objects*[10].

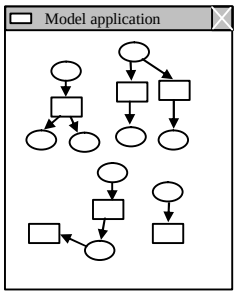

Fig. 2.21 Sequential objects application

As in this case objects communicate from inside the application, there is no need for any communication protocol since they are all confined to the same space of variables and functions (methods) that belongs to the application.

This is perhaps the most wide-spread approach for modelling in the industry of our days. It runs in cheap single processor computers and supplies the most popular and general tools for simulation like SIMCSRIPT, ARENA, *FlexSim*, *C++Sim*, *Simul8*, among many others. Small, didactic or academic models, frequently of linear Supply Chains, can easily be found described in literature developed with such multi-purpose tools, but concerning cases of real Supply Chains the situation is slightly different. Much less interesting articles are available, and also few tools are specific for Supply Chain simulation.

An exception is perhaps the *IBM Supply Chain Simulator* (see Bagchi et al., 1998), with which some real Supply Chains have been modelled and studied in terms of site location, replenishment policies, inventory levels, lead times, customer services, etc., using a dynamic simulation perspective. This is a tool used by experts for consultancy. As the same authors noticed, usually clients do not want their results revealed, in order to keep business advantages, and therefore

[10] In truth, this term is not frequently used and in practice people refer to this approach simply as "objects". Even so, it is used sometimes in order to distinguish the idea from "distributed objects".

few models of real Supply Chain cases appear published in articles.

Another exception is the powerful *eM-Plant* simulation tool, also for *Supply Chain Management* (SCM), which for instance is used by BMW in its international automotive manufacturing Supply Chains (Crooks, 2002), or even in other real case studies (Byrne & Heavey, 2004). This is a commercial tool allowing a good interface with databases through *Open Database Connectivity Link* (ODBC) and the possibility of using the OOP language *SimTalk* to customise its objects realistically. With a tool like this, one can already achieve the simulation of Supply Chains where products are in hundreds or even thousands and work centres in dozens, for example. Of course, that must be a job for a team of experts.

References to some other Supply Chain specific simulation tools based on this *sequential objects* technology can be found in exploring the *Web*, such as the *Supply Chain Builder* form *Simulation Dynamics Inc.*, or *GoldSim* from *GoldSim Technology Group LLC*, but also the user will find it difficult to get enough detailed information about them. On one side, they usually either belong to consultancy enterprises or are extremely expensive. On the other hand, results obtained with them are also often restricted, confidential, since the client managers understandably prefer them not to be published.

The Supply Chain simulator described in the next chapter intends to be of the same nature as these dedicated simulators, even if it stills in development and some more improvement would be necessary to make it a commercial tool. Yet, a real Supply Chain case has already been analysed with it, related to the procurement of additives for lubricants in the oil refinery of Porto, Portugal (Feliz-Teixeira & Brito, 2005). This system included a factory and 10 suppliers working with around 200 different products. The simulation of a one-year operation was achieved in 15 seconds.

2.3.4 Distributed objects

Distributed objects is another approach presently entering the phase of maturity. As the name suggests, this technology aims to distribute the objects, that is, to build the overall model based on a collection of *sequential objects* interconnected by means of a protocol of communication (see Fig. 2.22). Such objects, also commonly referred to as *components*, can be written using interfacing technologies like COM, .NET, *Win Sockets*, etc. Of course now an object can be even more complex than those of the previous case, since it is seen as an individual application. Thus, an object can now be a detailed model of a warehouse, factory or any other element of the Supply Chain, for example, which then will be connected to the others by some kind of protocol of communication.

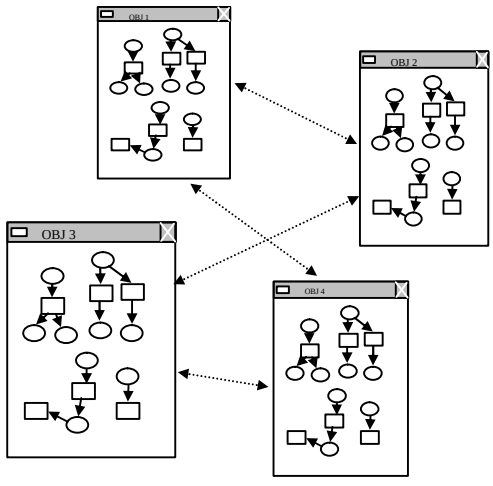

Fig. 2.22 Distributed objects modelling approach

Since each of these objects owns a private space for its variables and methods (*distributed memory*), all of them need to communicate significantly more than in the previous case (*shared memory*) in order to exchange the indispensable information. A large amount of time is therefore spent on serial communications, what will always be slower than in a shared memory approach, especially when the communications traffic increases, as is expected to happen in most complex systems. This often results in slower processes of simulation, especially if running in a single machine. But this approach also leads to difficulties in synchronizing causality among components, on facing dependence on the reliabilities of different objects, etc. These are certainly some of the reasons why the industry does not seem interested enough in this proposal, despite the obvious advantages of model reuse and interoperability. But it is an attractive approach for certain high performance simulations based on powerful computer systems, like clusters of PCs, networks of workstations, etc. Mainly, this approach is widespread in the military industry. Opinions of its major supporters can be found in Taylor et al. (2002).

There were also some trials on promoting this trend for Supply Chain modelling (Gan et al., 2000; Taylor, Sudra et al., 2002; Zeigler et al., 1999), despite the fact that the inexpensive *sequential objects* appear more compact, more controllable and much less computer resource dependent. Despite the promises of fast simulation execution, interoperability, geographical distribution and scalability, which are very attractive characteristics for battle simulations, multi-player distributed games, air traffic control, etc., the fact is that in simulating 300 days of operation of a Supply Chain with 10 suppliers and one manufacturer[11] Ping et al. (2001) reported simulation times between 100 and 250 seconds, depending on the protocol of communication used. On the other hand, with the same Supply Chain structure distributed between Singapore and the UK, Turner (2001) reports simulation times of the order of 500 to 1000 seconds, or even more when using a WAN network. This would be considered much more than 10 times slower, compared with the performances obtained with the previous paradigm. The model detail of each facility and the number of products considered were not revealed, however.

To better understand this tendency, we represent in figure 2.23 the scheme of a generic environment of *distributed objects*. Each dashed zone represents a processing unit, which for simplicity can be thought as a single computer.

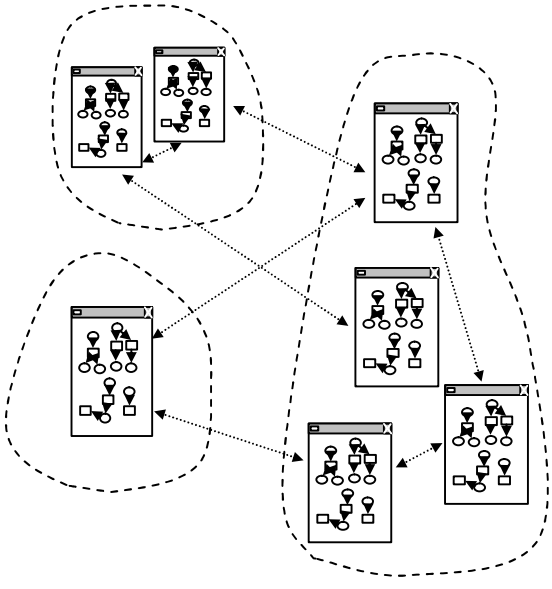

Fig. 2.23 Distributed objects in different processing units

[11] This structure is very similar to the one of the refinery simulated with our *sequential objects* approach, with which one year of operations could be simulated in 15 seconds.

In the case of this figure there is a computer running one model, another computer running two models and a third one running four models. These computers can be connected by some sort of a network, or even the *Internet*. Of course, there are two obvious interesting aspects in this configuration: the full model is made of smaller independent models (objects, or components), and these models are spread by more than one computer, that in principal speeds up the execution of the entire model. But, at the same time, this scheme implies the need for a strong synchronization between the times of the individual models and the simulation time, if running a dynamic simulation[12]. Even if there are some methods of achieving such synchronization (*conservative*, *optimistic*), the fact is that this is a weak link in the proposal, as the efficiency of such methods is not so high (Porras & Ikonen, 1999). Actually, *conservative* methods need to block the time advance in the fastest models in order to follow the slowest, while *optimistic* methods need to unschedule events and to rollback the simulation in those models that run too fast. Slowness or instability is what results from this.

Another weak characteristic is that since the full model is not running in the same machine, if any problem of communication exists, even in a single connection, the whole simulation is affected. So, this seems an interesting approach for organizations that make use of high performance computing and, at the same time, can have physical access to all the machines running the models. That is, of course, the case of military organizations and certain scientific research institutions. To the contrary, the industry of our times seems to maintain a preference for the robustness offered by *sequential objects* applications.

2.3.5 Agents

Generically, *Agents* can be thought of as a kind of *distributed objects* with embedded *Artificial Intelligence* (AI). They include techniques for learning and for reasoning with which decisions are taken frequently based on risk/benefits analysis. There is a long list of different kinds of *Agents* — please refer to what is considered the "Bible" of *Agents* (Nwana, 1996) — but, when applied to Supply Chain, *Agents* are largely seen as "intelligent" components for automating actions of planning, optimizing, controlling, etc., playing in an ambience of frequent exchange of data with other *Agents*, that can include negotiation and collaboration. *Agents* communicate with each other by means of messages.

Swaminathan et al. (1998), for example, were proposing a multi-agent framework for Supply Chain analysis that later have been used by IBM for modelling some real Supply Chains. In this framework they used a combination of simulation modelling and analytical modelling to provide the user with resources to analyse both static and dynamic issues in the Supply Chain. Two kinds of elements were used as modelling primitives: structural, which comprised the different *Agents* for assigning to retailers, distributors, manufacturers, suppliers and transportation; and control, in which other *Agents* were made responsible for the policies of information, demand, supply and materials flow.

[12] This problem does not exist while running Monte Carlo simulations (static simulations), so these systems are extensively used for such sort of approaches.

Recently, however, *Agents'* technology seems much more focused on dynamic information systems to assist in on-moment and online *Supply Chain Management* (SCM) (Ahn & Lee, 2004; Benisch et al., 2004; Chandrashekar & Schary, 1999; Pardoe & Stone, 2004). Ahn & Lee (2004), for instance, present a dynamic information network based on multi-agents in which the *Agent* behaviour is modelled with *Petri nets*, and messages exchanged using *Internet* protocols like TCP/IP, HTTP and SMTP. In this case, each company in the Supply Chain is managed based on four basic *Agents*: the *Market Estimation Agent* (MEA), which directly observes markets and estimates demands; the *Order and Production Agent* (OPA) which is focused on handling orders and production; the Supply *Chain Structure Agent* (SSA), responsible for constructing information networks; and the *Planning and Scheduling Agent* (PSA) who plans and schedules orders and production.

This is, of course, slightly different from the traditional simulation approaches where *warm-up* times must be considered and several *replications* must be completed before retrieving any sort of quantitative information for characterizing the system. To the contrary, in these SCM proposals the goal is essentially the optimization of decisions, the help in on-moment SCM and *e-Commerce*, very appropriate to our times of expansion of *Internet* through the planet. Anyhow, there, what is simulated is management decisions, based on some static functions constantly updated and inspected by "intelligent" algorithms, not the physical operations of the Supply Chain.

In addition to this, it is common that researchers refer to the difficulties found in interpreting results obtained with *Agents* in dynamic simulations, as the uncertainties rise with the self-ruled decisions taken by such modelling components. As it is known, uncertainties automatically propagate from the *inputs* to the *outputs* of the models, so, naturally things are expected to get worse when sources of uncertainties exist within the models, as easily happens in dynamic simulations based on *Agents*. This is one of the reasons not to include this technology in our modelling approach.

2.3.6 Web-based

What is habitually called *Web-based* simulation is a sort of simulation service provided within the *Web*. While the idea of *distributed objects* intends to introduce the advantages of a certain multiprocessing structure, the *Web-based* proposal aims mainly to introduce the advantage of a multi-user system (Fig. 2.24).

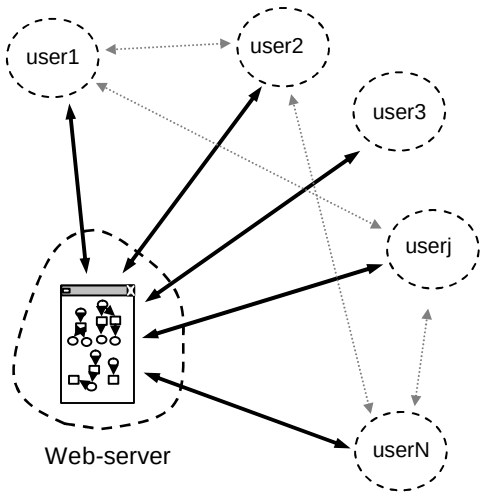

Fig. 2.24 The basic idea of Web-based simulation

Almost involuntarily, this makes us think of the 35 years old UNIX operating system, which included all these attributes in that time. How developed would operating systems and networks be at present if we

would have explored better this direction? Metrics of evolution must certainly focus on what we have achieved, but probably they would even be of a superior kind if they would also focus on the excellent opportunities we have lost, and measuring one against the others.

Web-based simulation is very much following the tendency of *Internet* service providing, as observed. Any special debate about this matter in order to understand or justify the great advantages of such an approach is therefore dispensable, since they appear obvious from the kind of network topology in question (see figure 2.24).

The simulation engine is made accessible on *Internet*, usually on an *Internet* site, and distributed users can use it from remotely as a resource of any ordinary *Webpage*. Notice that each user can also communicate with other users by any of the means provided by the *Internet*, as email, SMS, chat, messaging, etc., making this approach especially interesting for distributed groups of decision, as in many cases is the Supply Chain system.

Frequently, the simulator engine is a JAVA applet embedded in the HTML code of the site; the communication between the users and the model can be very simple, configuring the model by introducing values into appropriate fields and receiving the simulation outputs in the form of tables of values or charts, or even by email, for example. More recently, exchange of information is sometimes achieved by means of XML file transfers between the *Web-browser* and the simulation engine (Kuehn, 2005). Chatfield et al. (2004) also used XML to transfer Supply Chain models to a remote simulator, where these models were finally interpreted and executed.

Commercial *Web-based* Supply Chain simulation engines are not yet so frequent, but one can mention the *scSimulator*^{TM} from *DecisionCraft*[13] company as it seems to be one of those tools deserving reference (see also Lau et al., 2002). It already allows the representation and simulation of average-complex Supply Chains, at least with a level of detail that seems appropriate to the extended enterprise.

This technology, however, seems not to be so attractive to the industry too, maybe due to some important issues arising from the fact that it is committed to run within a public communications network. Industrial espionage, low versatility in model development and configuration, poor simulation speed, when compared with standard *sequential objects*, could be pointed as some of these questions. The disadvantages are quite similar to those of the distributed option previously discussed. Such approaches can still be very interesting either for vertical institutions or in environments of trust, of course.

In reality, this technology has showed more impact on academic and educational purposes, being applied in *Web-tutorials*, interactive teaching of complex dynamics, small-medium simulation studies, blend learning, etc., (Kuehn, 2005; Veith et al., 1998), and also in interactive multi-user simulation games, as in the *Web* version of the "Beer game"[14], or in the more complex LINKS dedicated to SCM (Chapman, 2005)[15].

There is sometimes the inclination to call *Web-based* simulation also to a certain

[13] http://www.decisioncraft.com/scsimulator/
[14] It can be played at: http://beergame.masystem.se:8000/
[15] Game LINKS at: http://www.links-simulations.com/

distributed simulation in which the support network is the *Web* (Orsoni et al., 2003), but that probably represents, at least in our opinion, a little inflation of nomenclature.

2.3.7 Parallel

This approach is often mentioned along with distributed simulation in the literature. Distributed computer systems can in fact sustain distributed and parallel processing. Although the nomenclature is somewhat diffuse on this issue, one can always follow the clear separation that exists between the hardware processing structure, and the software processing method. In the first case, either there are some interconnected machines each one with its own operating system (distributed hardware), or there are several processing units running under the same operating system (parallel hardware). This last option is considered the classical parallel system, where diverse software jobs run in different parallel processors, usually in an environment of shared memory.

Nevertheless, in a distributed hardware structure one can also assign different jobs to different processing units in order that they run independently of each other, and that is also considered parallel processing. Parallel therefore refers to *time*, while distributed refers to *space*. As one can see, nomenclature can get considerably boring, in its subtleties.

Thus, instead of talking of distributed-distributed, distributed-parallel or parallel-parallel, we prefer to look at the issue based on how time and synchronization are handled among the distributed components of the model. If two components can run independently of each other, we say they run in parallel. To the contrary, when they need to wait for each other to synchronize the time or share information, we say they run in a distributed manner. Of course, some distributed component can also be broken down into smaller components which can then be made to run as parallel tasks in a multiprocessor computer.

This view on classification also explains why Monte Carlo simulations, being of a static type (there is no need for time-marks or synchronization), are the simulations often running in parallel machines. And it also helps explaining, indirectly, at least, why it is difficult to find dynamic parallel Supply Chain simulations: once they are dynamic, they need time synchronization, which automatically leads to the distributed classification.

Apart from this, parallelism is being used as a way for improving speed in distributed simulations, by breaking certain components into smaller parts and making them run in parallel machines (Ping et al., 2001), for example. But principally this approach is being empowered by the modern tendencies of *Grid* computing, a world wide distributed computing that makes use even of the *Internet*, specially focused on parallel processing of huge computational tasks. This is seen as very promising for Monte Carlo simulations (Hill, 2004), as on the *Internet* thousands of machines can be made to run *Grid* jobs simultaneously. As the same author notices, 50000 protein sequences could be compared with distributed *Grids* software where more than 75000 Internet users have participated. The results could be achieved in 2 months, while 1170 years would be necessary to achieve the same in a single computer. However, he also notices that many applications cannot be parallelized.

Finally, also in this case one must be careful with nomenclature, since the term *parallel simulation* is sometimes used to classify common discrete simulations made to run side by side with the systems in order to help managers in optimizing on-moment decisions. Recent news about this can be found in Bodenstab (2004).

2.4 Which approach to choose

With such a diversity of methods to create simulation models and to translate them into computer applications, one could imagine that perhaps some difficulties would arise when deciding which approach to choose. In reality, however, such a problem almost does not exist in practice, since usually people naturally follow the approach of the simulation school they belong to. Things seem to be slowly changing with the globalization of methods and ideas, but in fact the first impulse is still given by the school, or the culture of simulation that surrounds the analyst.

2.4.1 Level of detail

Apart from that, it makes sense to discuss other motivations that can influence the choice of the simulation approach. And the most important is, probably, the level of detail required in the modelling. This is also related to the level of management in question and the answers one expects from the simulation.

Concerning the Supply Chain, and using as support the three nodes example of figure 2.25, the level of detail can go from the consideration that each node is simply described by a function $f_j(t)$ with a certain number of input variables (the little black spots pointing at the nodes in the figure), to the consideration that each node is described by a highly detailed model of the correspondent facility.

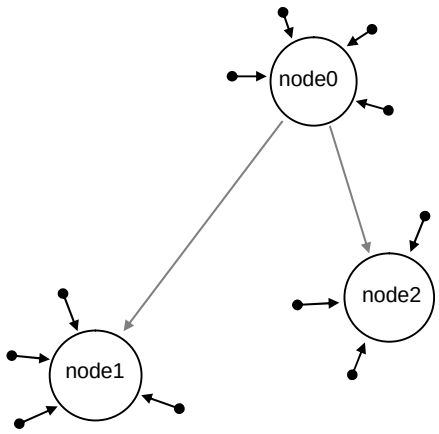

Fig. 2.25 Elemental model of a 3 nodes Supply Chain

Obviously, both the model complexity and the time needed for processing it increase with the increase in detail. The least detailed way of looking at the problem is to attribute to each node a single value (as well as to each arc) and transform it into a *Linear Programming* (LP) problem, a method since long ago used for optimizing the location and capacity of the various facilities. At such level of detail, any standard commercial computer has enough processing power to fast resolve a network structure of a realistic Supply Chain, with the number of nodes varying in the order of a dozen to the order of some hundreds.

As the detail increases, the node will be represented first by an analytic function independent of time (static simulation); then as a sequence of some time dependent internal processes (*System Dynamics*, if continuous; *Petri nets*, *activity diagrams*, *sequential objects*, etc., if discrete); and finally as a complex object with a level of detail depending on the case. Obviously, the

more complex the node gets the more the slower the simulation becomes, since more and more events must be processed per unit of time. Roughly, if the node is 10 times more detailed, the simulation speed will be at least 10 times slower. This explains why approaches like distributed and parallel processing can become extremely attractive to simulating systems where highly detailed nodes are considered. Higher detail in the nodes also implies a higher number of input variables to configure the model, of course, and that contributes for the increase of the time spent on preparing the model to run.

2.4.2 Where to stop

Apparently, there are no limits on the level of detail other than those imposed by the resources to create and to run the model, in which time and money are of course included. For many years a certain idea that detail could be good *ad infinitum* has been established, but presently analysts start to be more prudent and moderate on this issue (see for instance Clay, 2000; Kulick & Sawyer, 2000). Here we will expose some of the most significant aspects that can justify limiting the model details.

Uncertainties: let us consider an illustrative case. Suppose that a certain *live activity* named *Activity*$_0$ has a stochastic duration T_1 estimated from a probability distribution, which therefore imposes an uncertainty ΔT_1 in the final measure, as represented in figure 2.26. Suppose now that in a more detailed version of the model this same activity is substituted by a sequence of three new "smaller" activities (bottom part of the figure), each one with its own stochastic duration.

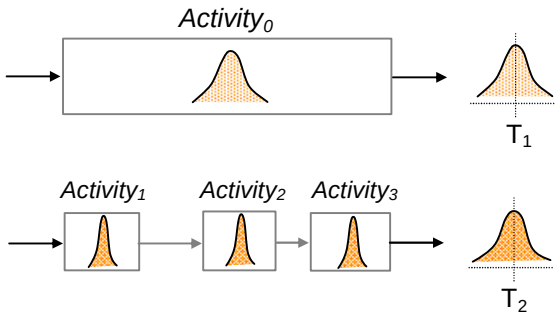

Fig. 2.26 Illustrative example of increasing detail in the case of a simplistic stochastic model

The final duration T_2 resulting from this "composed" activity will of course be affected by an uncertainty ΔT_2 which roughly results from the summation of the individual uncertainties. Clearly, if this sum is inferior to the previous case, the detailed model is contributing to the clearing of the output signal (results); if not, the detailed model is contributing to the increase of the "noise" in the output signal (results). This simple case demonstrates that the detail itself can make the simulation lose "resolution", due to an increase in the induced "noise".

So, what in fact is relevant is to ensure a reasonable *signal-to-noise ratio* (to use the nomenclature of the signal processing field), and that must be an important element of the analysis before deciding on the kind of modelling approach to use. This largely depends on the quality of the input data available for the simulation. In practice, it does not make much sense to develop a highly detailed model of a system when the stochastic input parameters are only known within a wide range of uncertainty.

Output complexity: another aspect is the increase of complexity of the outputs when detail is increased. Even supposing the

problem of uncertainty resolved, increasing the model detail frequently results in making "visible" certain effects that previously were filtered by a less detailed version. Principally, this happens when modelling systems of intricate topology, as is the case of most Supply Chains. If the system is very simple, with one, two or three nodes, the possible combinations of states are also small and such effects are not so significant. The problem becomes evident when several nodes interact. A common case that can be used as example is the time variation of inventory in a factory connected to several retailers, for instance. With a low level of detail in sampling inventories, say monthly, the analyst still have the chance to "follow" and mentally "understand" the dynamics of the facility. But when the inventory is represented daily the "signal" starts to be so complex that few more information can be retrieved from it. Figure 2.27 represents a simulation output obtained from a similar case using a didactic Supply Chain of a *single* product. The inventory is computed daily.

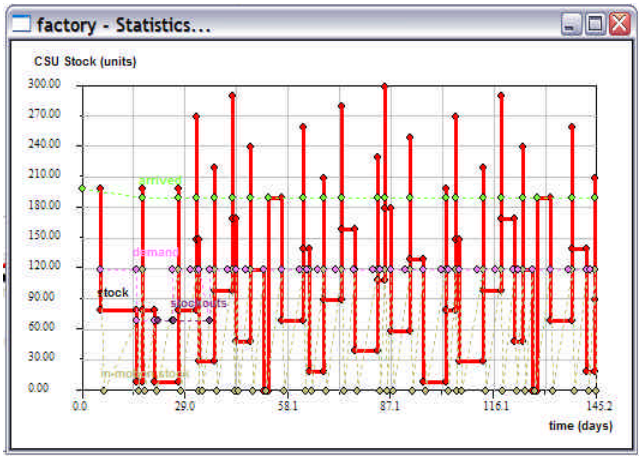

Fig. 2.27 Example of an average complexity in the daily inventory of a factory in a relatively simple Supply Chain

Since complexity is not (yet) understood as a whole, we need to break it down and classify it by means of simpler concepts, which then we can handle easier and follow its dynamics with less difficulty. In a sentence, we need abstraction to think and to execute decision operations.

This means that when the detail of a model is too high it can turn too difficult to classify its responses in all the diversity of its states, and therefore to use abstraction to understand the system. In such circumstances, the results obtained start to contribute much more to confuse the mind of the analyst than to clarify it. Of course one can always filter such results again, and retrieve from them the same sort of information that could be retrieved with a less detailed model. Obviously, the point is that frequently one trusts more in a detailed model than in the opposite.

Sampling rates: although this aspect is not found in literature about modelling and simulation, at least as much as we know, it can be mentioned as an additional help for establishing the level of detail of the model. This is based on the old Nyquist sampling law (Nyquist, 1928) which since long ago has been intensively used in the diverse kinds of signal processing. This law basically states that *a signal must be sampled at least with a frequency equal or superior to two times its bandwidth in order that it can be later recovered from the sampled version*. Indirectly, this also means that if one is interested in analysing the system's dynamics based on events separated in time by ΔT_0 one has at least to "sample" the behaviour of such a system with a sampling rate double that, which is $f = 2/\Delta T_0$. This means, that when we are interested in

analysing a system monthly, we must at least collect some data fortnightly. Of course, a weekly sampled data can help even better in the reconstruction of the original behaviour, but the fact is that more than, lets say, 10 times the desired rate, is almost more than enough to build a "representative" sample of such behaviour. Therefore, after a certain level of detail (say, 10 times the highest delay to be perceived), the more the detail increases the less it contributes to the definition of the output signals.

This can be translated to modelling in a very simple way. Suppose the minimum delay we want in the simulation is 24 hours (1 day), like in some operational models of the Supply Chain. So, at least we have to sample the system with events separated by 1/2=0.5 day, if there is any change of state. This means we have to choose details that at least include live activities with durations of this order. In the maximum detail, however, it makes sense to represent such activities with duration of 1/10 of a day, but not much more than that. More than that will probably become redundant.

Future is not predictable: finally, this is what I consider the first thought to have in mind when building a stochastic model. The future is not predictable by manipulating stochastic data from the past, that is, by the usage of statistic methods. In fact, what is estimated with these methods is how much the future will be similar to the past. In that sense, it is not a prediction of the future, but a prediction of the past that will probably repeat. This is important to be referred since sometimes people increase model details as if they automatically would enhance the ability of forecasting in terms of prediction and not of such an estimation of a repetition. Perhaps this is due to an excess of optimism, but the fact is that it still causes some confusion in certain minds, mainly when high quality graphics are used to visualize simulation. Besides, probably the level of such "predictions" is not much higher than that obtained by applying simple good judgement, and with a little more of wisdom perhaps it is even surpassed. For instance, a large number of ordinary people have been able to understand that the recent military operations in Iraq would turn into something very different from what the USA and its allies were expecting, while it seems the sophisticated simulations running in *DoD* powerful super computers were not able to perceive it. Common sense, in this case, showed to be more reliable than high-tech extremely detailed simulations. As Schrage (2000) observes, "*truly effective models aim not to predict the future but to envision possible futures that can be managed successfully*". I believe as well that simulation can be a very useful tool for analysing and to characterize systems, but not to get any reliable conclusions based on parallel realities running as a sort of predictive game.

2.4.3 Simulation intents

The objectives behind a simulation study are perhaps what influence the choice of the modelling approach the most. Such objectives can of course be diverse, as they frequently depend on a wide range of issues, going from the simple purpose of promoting or advertising some system to the need for executing precise and complex calculations while projecting an expensive structure, for example. In this section will be pointed out the sort of objectives that have inspired the Supply Chain simulator described in detail in the next chapter.

Together with the arguments discussed in the previous sections, this is expected to contribute to justify the sort of approach chosen.

During my first visit to the Management School of Cranfield University, UK, the idea was to be aware of and absorb the actual tendencies and needs of the Supply Chain manager. Concepts of flexibility and agility applied to the Supply Chain seemed the most interesting issues at the moment, due to the surge of turbulence in globalized markets and the need to frequently readjust both the structures and the policies. Some review of the literature, some discussions on the various perspectives of simulation and some brain-storm sessions with a highly skilled team in *Supply Chain Management* (SCM), have finally led me to condense the trends in the two major directives: (1) react fast and (2) be aware. These were, if one can say, the vectors which would establish the design of the future Supply Chain simulation tool (Figure 2.28 shows the original sketch of the idea).

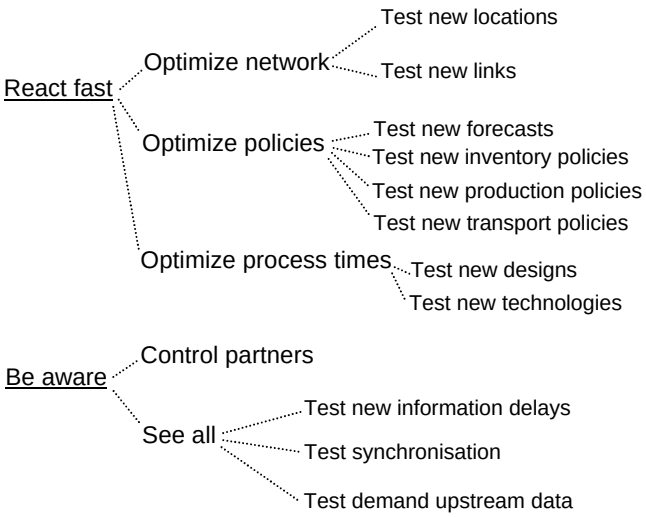

Fig. 2.28 Sketch of the primer SC simulation objectives

With the current inclination towards JIT, the ultimate customer is the one who triggers the actions, which explains the importance of these two basic principles. "*Be aware*" gives the managers a sense of vigilance, on monitoring how the system is running and what kind of trends are there; while "*react fast*" gives the ability to effectively answer in terms of actions, or movement. This clearly results from managing what is called the *information pipeline* and the *materials pipeline*. In order to test *fast reactions* the analysts must be able to simulate different kinds of network structures, different policies, different process technologies, etc. On the other hand, related to "*be aware*", the analyst must also be capable of testing different information delays, different visibilities of demand, etc. This led to the choice of a simulation approach that could at the same time allow trying different system topologies and policies, as well as different information delays and even some processes of ordering material based on *Internet* ordering and *e-Commerce*.

Other aspects like cooperation and visibility would not be fully implemented in the simulator due to some difficulties in establishing mechanisms for synchronising partners, in an approach that does not consider centralizing decisions. In reality, in this version of the simulator there is not any element in charge of the integrated management of the Supply Chain. Instead, the Supply Chain system is considered horizontally managed, with each facility being independent from the others as in a truly free-commerce structure. At the same time, as the race now stands between Supply Chains, which many times in practice overlap their structures, it makes less sense

to contemplate a centralized management, as several elements could be captured in the "crossed fire" of dissimilar policies of neighbour Supply Chains. Thus, if in reality the competition stands between Supply Chains, a simulator using a horizontal management structure will probably be more interesting for analysing real cases.

Finally, since the intention was to build a simulator founded on the dynamics of the operational field, allowing fast modelling and execution of simulations, and since it was considered more relevant concentrating the analyst in a single application where managers also could focus on examining the Supply Chain, the approach chosen was the *sequential objects*. Novel contributions for modelling and simulation of Supply Chain systems will consequently result much more from the modelling structure proposed than from any announcement of a new software or hardware technology.

2.5 Model consistency, accuracy

A model must of course be consistent with the system it models. If not, the results obtained with it will diverge from the reality and no legitimate conclusions can be retrieved from the process of simulation. In games, this issue is not so critical, but when dealing with "serious" simulations this is of major importance. Figure 2.29 intends to represent five different moments in the time evolution of a system (smoother violet line) and its model (lighter grey line). Notice that the model is represented never exactly fitting to the system. This is a general representation, thinking of the system as a spatial component that evolves in the time.

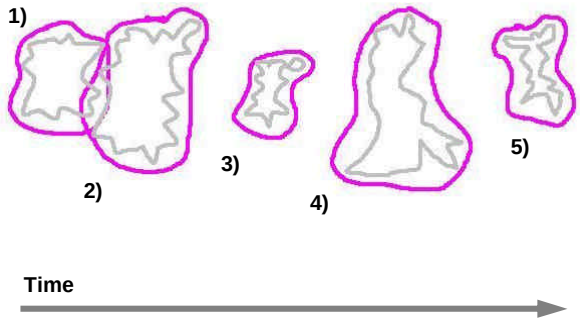

Fig. 2.29 System and model evolutions with the time

Obviously, despite the discrepancies that can be expected when comparing system values and model values, this image shows a good consistency along time, since such discrepancies seem to preserve the same magnitude. The *accuracy* of the model is simply inversely proportional to these discrepancies, as we know.

The problem is that most of the time the evolution of the model will not always "fit" so well to the evolution of the system, as these discrepancies usually increase with the number of cycles simulated, resulting in an inconsistent model evolution (Fig. 2.30).

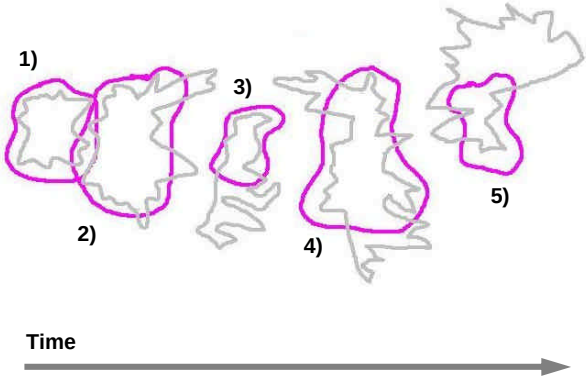

Fig. 2.30 Inconsistent model evolution

Two main reasons for this can be pointed. On one side, there are some deviations that in certain processes accumulate with the time, since discrete simulation processes are basically founded on procedures which have strong iterative or recursive logic. This happens, for example, in the extremely complex models used for "predicting" the weather.

In the Supply Chain, for example, it can mean that if one calculates the next value of the inventory based on its previous value and a mathematical function, for example, it is expected that after some cycles of calculations the results will be diverging from those expected in the system and that such deviation will depend on the deviation introduced by the function. This can give rise to cumulative problems which probably could be minimized by "reconfiguring" the model with values regularly acquired from the system, as in a kind of reset process. Though, in most complex and extensive simulations this is evidently not possible, since the system is often either not accessible or not available for retrieving such measures. One must rely on the accuracy of the mathematical functions and relations which make up the model. They must be as error free as possible. And such exactness has to be considered with regard to both the spatial and the temporal components of the model.

On the other side, the propagation of uncertainties in stochastic models affects to a kind of blur the parameters related with this type of variables (Fig. 2.31).

After a certain time of simulation, this effect may turn certain variables useless for retrieving any reliable conclusion. When the uncertainty of a value is of the order of the value itself already no conclusions can be retrieved.

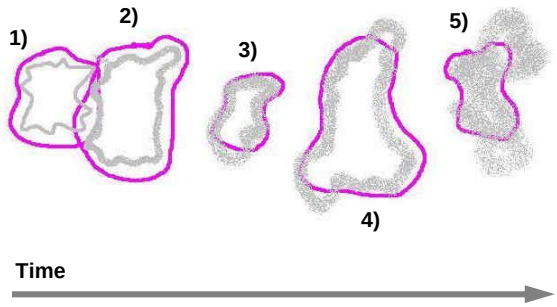

Fig. 2.31 Somehow a consistent model, but blurred

This problem is more common in complex models using statistical input variables, as well as when the results are computed based on long processes of simulated activities. Since there is often little precise information about the values of certain factors when modelling real systems, these factors are often estimated also by means of stochastic variables. Once again, this uncertainty can grow with time and, after a certain point, make the outputs of the model a confused space of blurred values. To minimize this, it is important that the stochastic variables are as exact as possible, that the model will not be used in an excessive number of cycles of simulation, and that the processes involved in the calculation of the results are the shortest possible, as uncertainty naturally increases while entities cross from state to state (or activity to activity). In some circumstances, as we have seen previously, it may be more advantageous to represent processes with less detail than as a chain of detailed activities.

These two problems are concurrent in practice, and the results of a simulation can

transform into a serious disorder if no strategy is previously established to minimize such effects, which, of course, are much less observable when simulating simple systems or short periods of time. The strategy that normally confers some confidence to the model is based in the following sequence of procedures: *verification*, *validation* and *accreditation*. Notice, however, that in reality only after all the aspects of a model would be compared with the same aspects of the system one could ensure if the model is a good representation of the system or not, but this is an impossible task to finish in practice, particularly for dynamic models where the states vary with time. Thus, it is important to keep in mind that the decision to consider the model suitable or not suitable will frequently result from an *inference*. *Verification*, *validation* and *accreditation* will confer some legitimacy to the model, but do not ensure that all the results will in fact be repeatable by the system.

Verification is the procedure of verifying the logical consistency of the model so that it can be used as a good representation of the system. This is normally done during the phase of development and includes all the reviews and adjustments considered as "debug". As there is no especial procedure for doing this, the same methods are used as for testing and debugging any ordinary computer programme. In the end, the logic of the model must appropriately mimic the logic of the system. Some tests with small examples of models can be done and their results analysed and compared with those expected mathematically, for example, in order to gradually build up the confidence in the model functionality. Once the model is implemented and verified, it can pass to the phase of confrontation with reality, that is, to *validation*.

Validation is the procedure of comparing the results generated with the model with those generated by the system. The model is first configured with real data, and then made to run. Obviously, this implies some difficulties, as usually not all the output variables from the model can be confronted with values generated by the system. The available data collected from the system is usually less detailed and in much smaller amounts than that generated by the model. Thus, comparing these two sets of data can require substantial manipulation, and in that case the *validation* becomes more indirect. A good idea is to base this process on the comparison of variables that represent accumulated values, if possible, as their deviations are more sensitive to any discrepancies between the model and the system. Point to point comparisons may also be used, and they are useful to enhance the validation process, but this needs detailed data collected from the system. Another way is to run the simulator with historical data and compare the model results with the respective chronological system results. This is an excellent technique and it is recommended to be used whenever possible.

In the validation process the ultimate idea is to calculate "*an error rate based on data independent of that used to estimate the model. This exercise gives a statistically valid estimate of the true error rate that the modelling procedure produces. It does not guarantee that the model is correct in any way. It simply says that if the same*

technique were used on a succession of databases to build a model, the average error rate would be close to the one obtained this way" (Small & Edelstein, 1997).

Accreditation is by some specialists, among whom the *Department of Defense of USA* (see DoD/US, 2002), considered the desirable last step to guarantee the accuracy of a model. It is a process that requires an institution with recognised authority to test and certify models. As Balci & Saadi (2002) made noticed, the well known *International Organization for Standardization* (ISO) prefer to distinguish accreditation and certification as follows: *accreditation* is a "*procedure by which an authoritative body gives formal recognition that a body or person is competent to carry out specific tasks*"; while *certification* is seen as a "*procedure by which a third party gives written assurance that a product, process or service conforms to specified characteristics*". Ant this is what is usually accepted worldwide.

In practice, however, *accreditation* and *certification* are difficult tasks to achieve due to the complex issues converging on them, ranging form the fact that each type of modelling and simulation poses its own technical challenges (see Balci, 2003), to the fact that such issues generate obvious controversy related to the power given to those who would dictate. At present, these procedures are simply substituted by the credibility and skills of the model developer or consultant who directs the simulation project. Such "distributed" and "divergent" responsibility about model accuracy appears to be adequate for the diversity of civil projects, although justifiably it may be less interesting for military purposes, where models are intended to supply the same field of objectives and to contribute to a "convergent" goal.

2.6 Getting results with the model

In general, one could say that the most challenging aspects the analyst faces when confronted with the outputs of a complex simulation are: *complex behaviour* and *variability*. Complex behaviour urges the analyst to find ways to understand the output data, usually making him or her react by classifying such behaviour in terms of *categories*, as we often do with reality. Seeking relations or patterns already induces some abstraction and helps in getting some order out of the complexity.

There are several methods for assisting this process, as in the case of simply graphing the data using suitable charts; searching for *attractors* using (x_n, x_{n+1}) correlation graphs; comparing the data with a threshold and extracting only some particular information from it; using the *Fast Fourier Transform* (FFT) or *Discrete Fourier Transform* (DFT) to examine the data in the frequency domain; counting events so as to build histograms; applying *Data Mining* (DM) techniques for searching for relations and reorganizing the data in certain spots, etc.

These are some examples of how people face and try to react to the complexity of certain simulations. *Sensitivity analysis* and *antithetic variates* are other forms of searching for possible relations among the data by experimenting with the model in different simulation runs, for instance. A unique solution, however, does not exist. Final results depend even on the criteria applied to the raw output data. For an

extensive text on the traditional methods of analysis used in simulation please refer to Law & Kelton (1991).

Another difficulty about the results generated by complex stochastic models is the *variability* due to the propagation of uncertainties during the simulation process, responsible for the level of confidence of the output values. For the purpose of illustration, the next figure (Fig. 2.32) shows the same simplistic 3 nodes Supply Chain mentioned earlier, this time presented with some stochastic input variables. Each stochastic variable x_j is associated with an initial uncertainty dx_j, represented in the figure as a little blur. Also represented are two generic output variables Y_{01} and Y_{02} dependent on the same input variables x_2 and x_5 and on a generic constant *k*.

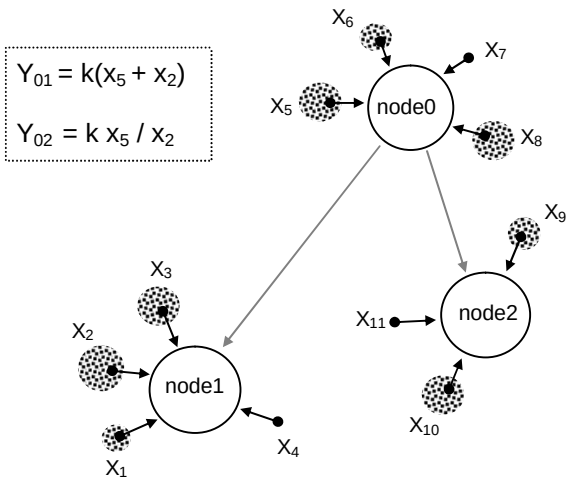

Fig. 2.32 A simple 3 nodes Supply Chain, with inputs x_j

Since each generic output variable *Y* will result from a function dependent on the input variables x_j, that is:

$$Y = f(x_1, x_2, \ldots x_j, \ldots) \tag{2.10}$$

at the end of each simulation run the general uncertainty in *Y* will be given by:

$$dY = \frac{\partial f}{\partial x_1} dx_1 + \frac{\partial f}{\partial x_2} dx_2 + \ldots \frac{\partial f}{\partial x_j} dx_j + \ldots \tag{2.11}$$

which shows that the final uncertainty is dependent not only on the initial uncertainties of the input variables but on these multiplied by their derivative factors. Different outputs depending on the same input variables can therefore exhibit different uncertainties, as will be shown by calculating the uncertainties of the Y_{01} and Y_{02} examples. In fact, these are:

$$dY_{01} = k(x_5 dx_2 + x_2 dx_5) \tag{2.12}$$

$$dY_{02} = k\left[\frac{x_5}{(x_2)^2} dx_2 + \frac{1}{x_2} dx_5\right] \tag{2.13}$$

And, with some manipulation, one gets:

$$dY_{02} = \frac{k}{(x_2)^2} dY_{01} \tag{2.14}$$

This means that as x_2 goes smaller than *k* the uncertainty in Y_{02} can get extremely high (please refer to Weisstein, 1999).

Using a similar type of graphic notation, one can consequently say that after *N* independent runs, a generic output measure *Y* will be represented as a group of *N* blurred spots from which a final value must be computed or inferred (see Fig. 2.33).

There is obviously a cloud of uncertainty associated with the measure of *Y* after the replications. Different averages of *Y* result from the different *seeds* with which the random number generator is initiated, in the beginning of simulation.

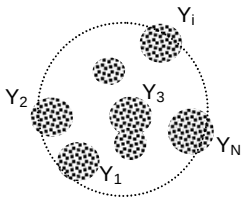

Fig. 2.33 Results of output *Y* after *N* simulation runs

Frequently, however, the propagation of uncertainties from the inputs to the outputs in each replication is ignored and only the average effects resulting from different replications are considered. That is, frequently each Y_i is seen only as an average value. The final result is presented, with its variance *var(Y)* and the confidence level required *G(q)*, as:

$$Y = \overline{Y} \pm \sqrt{\text{var}(Y)} \times G(q) \qquad (2.15)$$

where:

$$\overline{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i \qquad (2.16)$$

$$\text{var}(Y) = \frac{1}{(N-1)}\sum_{i=1}^{N}\left(Y_i - \overline{Y}\right)^2 \approx s^2 \qquad (2.17)$$

G(q) is a value retrieved from a *t-Student* distribution when *N<20* or from a *Normal* distribution for higher values of *N*. For a confidence level of 95% and *N>20* we have *G(q)* around 2.

Inserting now the contribution of the *average variance* observed at Y_i, and considering the replications independent, a more accurate estimation of *Y* will finally be given by:

$$Y = \overline{Y} \pm \sqrt{\text{var}(Y) + \overline{\text{var}(Yi)}} \times G(q) \qquad (2.18)$$

2.7 Final comments

Before entering the next chapter, it is important to mention that two other simulation approaches have been used in the aim of this work of research. The first was a test made on the *distributed objects* paradigm; the second, a distributed game somehow close to the idea of *web-based* simulation. These "tools" were intended for different purposes.

Distributed simulator: before deciding on the *sequential objects*, there was a period of hesitation regarding the way to go. Some distributed tests have, therefore, been arranged based on the interconnection of two detailed warehouse simulators (Feliz-Teixeira & Brito, 1999), using the TCP/IP protocol and the *WinSockets* as interface (Fig. 2.34).

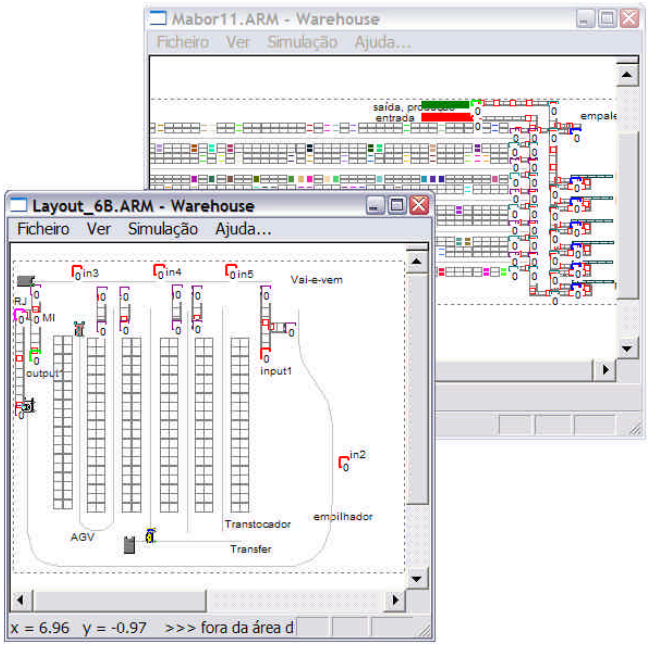

Fig. 2.34 Two detailed warehouse models have been made to run in a distributed mode, using TCP/IP

It was observed that this scheme (based on a conservative time synchronization method) worked, and even the visual effects of each simulator could be perceived, but, despite the speculative interest of the approach, it was substantially slower than the *sequential objects*.

What was interesting about this approach was that the time increment for synchronization may be chosen by the operator (for instance, set to 1 day), so, each simulator could run independently till that moment, when it would have to wait for an authorization to continue till the next time stamp. The time advance method was therefore a kind of "*time-slicing waiting for the slowest*", if this term can be used. However, it was observed that the approach was evidently slow even when simulating two facilities, leading to suspecting that it would be even less efficient when analysing a real Supply Chain. It could of course be improved, if each model would be less detailed and the time increments longer, but in that case one could also use a single application and concentrate the goals more on the aspects of *Supply Chain Management* (SCM) instead of on a simple technique of simulation. In effect, rough calculations after some cycles of experimentation led to estimate a speed of simulation around 40 times slower than in the case of the *sequential objects*[16].

Another disadvantage of this idea was that the project to build the simulator would be mostly focused on developing methods for transferring and managing data and ensuring the stabilization between the various simulators, rather than on conceiving some kind of interesting structure to model and simulate Supply Chain systems in a faster and easier way. In the end, this last aspect was seen as the most relevant.

Supply Chain game: apart from this, also developed was a *distributed* simulation application for SCM training, at least for the operational and tactical levels. A manual version of this game, known as "*Cranfield Blocks Game*", was first introduced by Richard Saw (Saw, 2002) to assist the students at the *Cranfield School of Management*. It is a multi-level Supply Chain structure (shown in figure 2.35) in which each group of students is in charge of one of the seven facilities, from the *Depot1* to the *Factory*. Certain demand patterns are injected at the customers level (*C1* to *C4*), and the students have to respond to this demand in an appropriate way, serving the clients and ordering new materials to the suppliers while adequately managing the inventory. More detailed information about this game is presented in chapter 5, and in Feliz-Teixeira et al. (2004).

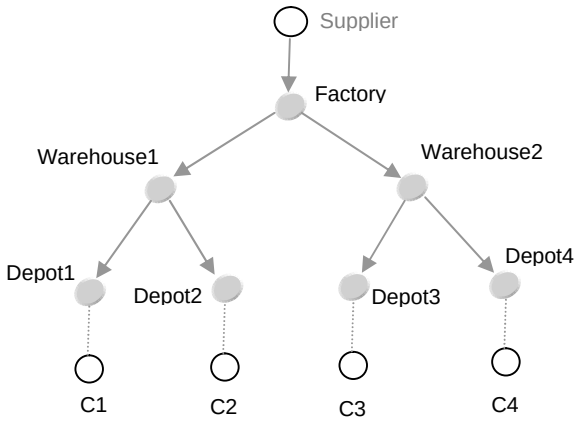

Fig. 2.35 Structure of the *Cranfield Blocks Game*

[16] It is interesting to verify that this estimation is not so far from the comparisons made in section 2.3.4.

This simulation software is made up of a *server* application, which must be running in a computer accessible by TCP/IP, and a *client* application, which is the interface provided to each group of students. The *server*, of course, controls the game.

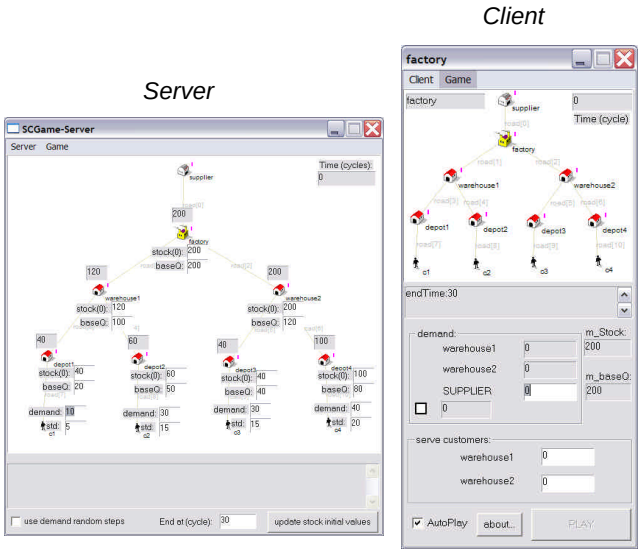

Fig. 2.36 *Server* and *client* applications for the computer version of *Cranfield Blocks Game*

Although the game can be configured to play on its own, in the *Auto-play* mode, as a sort of a distributed simulation tool, in reality it was conceived for being operated by human players, therefore belonging to what Balci (2003) would classify as a *human-on-loop* type of simulation. Such simulators are strongly intended for training personnel.

In this particular case, instead of playing the game by hand on a big table as it was the case of the manual version, the seven groups of players are distributed by several computers running the *client* application, as figure 2.37 suggests, and these computers are in turn connected to a SERVER where the *server* application is active.

This scheme is frequently mounted in a single room, with a *LCD-projector* attached to the *server* since this has the ability of displaying the inventory records along the time.

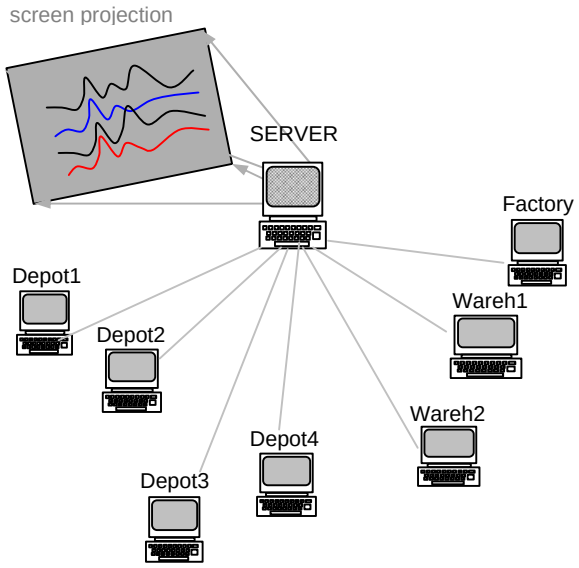

Fig. 2.37 *Cranfield Blocks Game* playing structure

Turning the projector on or off can be considered turning on or off the *visibility* to all the Supply Chain partners. Experiments can be made based on these kinds of issues. Some interesting tests have been made with this game in *Escola de Gestão do Porto* (EGP) with a group of students attending a Masters course on Management. The results obtained with this experiment will be shown in chapter 5, and have also been published in Feliz-Teixeira et al. (2004).

References:

Ahn, H. J., & Lee, H. (2004). An agent-based dynamic information network for supply chain management. *BT Technology Journal, 22* (2).

Bagchi, S., Buckley, S. J., et al. (1998). *Experience Using the IBM Supply Chain Simulator.* Paper presented at the The 1998 Winter Simulation Conference.

Balci, O. (2003). *Verification, Validation, and Certification of Modeling and Simulation Applications.* Paper presented at the 2003 Winter Simulation Conference, USA.

Balci, O., & Saadi, S. D. (2002). *Proposed Standard Processes for Certification of Modeling and Simulation Applications.* Paper presented at the 2002 Winter Simulation Conference, USA.

Beamon, B. M. (1999). Measuring Supply Chain Performance. *International Journal of Operations and Production Management, 19* (3), 275-292.

Benisch, M., Greenwald, A., et al. (2004). *Botticelli: A Supply Chain Management Agent.* Paper presented at the AAMAS'04, July 19-23, New York, USA.

Bodenstab, J. (2004, October 4). *ToolsGroup First to Benchmark and Control Inventory Optimization - Introduces a new Parallel Simulator*. ToolsGroup. Retrieved July 2005, from http://logistics.about.com/library/weekly/mpreviss.htm

Bruniaux, R., & Pierreval, H. (1999). *Simulating Supply Chains with System Dynamics.* Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.

Byrne, P. J., & Heavey, C. (2004). *An Industrial Supply Chain Simulation Model.* Paper presented at the Industrial Simulation Conference 2004, Malaga, Spain.

Cacioppo, K. (2000). *Measuring and Managing Customer Satisfaction*. Quality Digest. Retrieved June 2005, from http://www.qualitydigest.com/sept00/html/satisfaction.html

Chandrashekar, A., & Schary, P. B. (1999). Toward the Virtual Supply Chain: The convergence of IT and Organization. *Journal of Logistics Management, 10* (2), 27-37.

Chapman, R. G. (2005). *LINKS Supply Chain Management Simulation*: LINKS-simulations.com.

Chatfield, D. C., Harrison, T. P., et al. (2004). *XML-based Supply Chain Simulation Modeling.* Paper presented at the 2004 Winter Simulation Conference, USA.

Christopher, M. (1994). Logistics and Supply Chain Management*. Financial Times*.

Clay, G. R. (2000). *Venture Launch: Use of Simulation to Support Strategic Operational Decisions.* Paper presented at the 2000 Winter Simulation Conference, USA.

Crooks, K. (2002). *eM-Plant* (Power Point Presentation to BMW): Tecnomatrix.

Davidrajuh, R. (2000). *A Petri Net Approach for Performance Measurement of Supply Chain in Agile Virtual Enterprise*: Narvik Institute of Technology, Narvik, Norway.

Davies, R., & O'Keefe, R. (1989). *Simulation Modelling With Pascal*: Prentice Hall.

DoD/US. (2002). *DoD Modeling and Simulation (M&S) Verification, Validation and Accreditation (VV&A)* (Directive No. 5000.61): Department of Defense of USA.

Edelstein, H. (2001). Pan For Gold In The Clickstream. *InformationWeek.com*.

Edelstein, H. (2003). Using Data Mining to Find Terrorists. *Data Mining Review*.

Eilon, S., Wantson-Gandy, C. D. T., et al. (1971). *Distribution Management: Mathematical modeling and practical analysis*: Griffin-London.

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (1999). *Visual C++ Software for Warehouse Simulation (an overview).* Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2004). *On Measuring the Supply Chain Flexibility.* Paper presented at the 2004 European Simulation and Modelling Conference, UNESCO, Paris, France.

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2005). *Using Simulation to Analyse the Procurement of Additives for Lubricants in the Oil Refinery of Porto, Portugal.* Paper presented at the Industrial Simulation Conference 2005, Berlin, Germany.

Feliz-Teixeira, J. M., Brito, A. E. S. C., et al. (2004). *Distributed Application for Supply Chain Management Training.* Paper presented at the Industrial Simulation Conference 2004, Malaga, Spain.

Fox, M. S., Chionglo, J. F., et al. (1993). *The Integrated Supply Chain Management System*: Dep. of Industrial Engineering, University of Toronto, Canada.

Gan, B. P., Liu, L., et al. (2000). *Distributed Supply Chain Simulation Across Enterprise Boundaries.* Paper presented at the 2000 Winter Simulation Conference.

Gonçalves, J. F. (1999). *Gestão de Aprovisionamentos* (revista ed.). Porto, Portugal: PUBLINDÚSTRIA.

Guedes, A. P. (1994). *An Integrated Approach to Logistics Strategy Planning Usig Visual Interactive Modelling and Decision Support.* Unpublished Ph. D., University of Cranfield, U. K.

Heizer, J., & Render, B. (2001). *Operations Management*: Prentice Hall.

Hill, D. R. C. (2004). *Grid Computing and Stochastic Distributed Simulation.* Paper presented at the 2004 European Simulation and Modelling Conference, Paris, France.

Hitchings, G. G. (1969). Analogue Techniques for Optimal Location of a Main Facility in Relation to Ancillary Facilities. *International Journal of Production Research, 7* (3), 189-197.

Hopp, W. J., & Spearman, M. L. (2001). *Factory Physics* (second ed.): Irwin McGraw-Hill.

Kaplan, R. S., & Norton, D. (1992, (January-February 1992)). The Balanced Scorecard: Measures that Drive Performance. *Harvard Business Review, 70,* 71-79.

Kim, K., Jr., B. C. P., et al. (2000). *Agent-based Electronic Markets for Project Supply Chain Coordination*: Department of CEE and Networking Research Center, Stanford University.

Kuehn, W. (2005). *Blended Learning Applying Interactive Discrete Event Simulation in a Web-based Learning Management System.* Paper presented at the Industrial Simulation Conference 2005, Berlin, Germany.

Kulick, B. C., & Sawyer, J. T. (2000). *The Use of Simulation Modeling for Intermodal Capacity Assessment.* Paper presented at the 2000 Winter Simulation Conference, USA.

Lambert, D. M., & Pohlen, T. L. (2001). Supply Chain Metrics. *The International Journal of Logistics Management, 12* (1), 1-19.

Landeghem, H. V., & Bobeanu, C.-V. (2003). *Using Meta-Modelling to Process Petri Nets models of Supply Chains.* Paper presented at the The 2003 European Simulation and Modelling Conference, Naples, Italy.

Lau, J. S. K., Huang, G. Q., et al. (2002). Web-based simulation portal for investigating impacts of sharing production information on supply chain dynamics from the perspective of inventory allocation. *Integrated Manufacturing Systems, 13* (5), 345-358.

Law, A. M., & Kelton, W. D. (1991). *Simulation Modeling & Analysis* (second ed.): McGraw-Hill International Editions.

Lee, Y. T., LcLean, C., et al. (2001). *A Preliminary Information Model for Supply Chain Simulation*: National Institute of Standards and Technology, Gaithersburg, USA, and Musashi University, Nerima-ku, Tokyo, Japan.

Mason-Jones, R., Naim, M. M., et al. (1997). The Impact of Pipeline Control on Supply Chain Dynamics. *The International Journal of Logistics Management, 8* (2), 47-61.

Microsoft. (2003). *The Little Chip That Will Change Your Supply Chain Forever*: Microsoft Business Solutions.

Minegishi, S., & Thiel, D. (2000). System dynamics modeling and simulation of a particular food supply chain. *Simulation Practice and Theory* (8), 321-339.

Nikolic, D. M., Panic, B., et al. (2004). *Bullwhip Effect and Supply Chain Modelling and Analysis using CPN Tools.* Paper presented at the The Fifth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Department of Computer Science, University of Aarhus.

Nwana, H. S. (1996). Software Agents: An Overview. *Knowledge Engineering Review, 11,* 1-40.

Nyquist, H. (1928). Certain topics in telegraph transmission theory. *Trans. AIEE, 47*, 617-644.

O.B. (2005). *What does SKU mean?*: OnlineBusiness.com. Retrieved March 2005, from http://onlinebusiness.about.com/od/faqa/f/SKU.htm

Orsoni, A., Bruzzone, A. G., et al. (2003). Framwork Development for Web-based Simulation Applied to Supply Chain Management. *International Journal of Simulation, 4* (1).

Pardoe, D., & Stone, P. (2004). *Agent-Based Supply Chain Management: Bidding for Customer Orders.* Paper presented at the AAMAS'04, July 19-23, New York, USA.

Ping, G. B., Turner, S. J., et al. (2001). *Distributed Parallel Simulation of Supply Chain Models* (SIMTech Technical Report No. MIT/01/029/MAPS): Singapore Institute of Manufacturing Technology.

Porras, J., & Ikonen, J. (1999). *Approaches to the Analysis of Distributed Simulation.* Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.

PRTM. (2000). U.S. Technology Industry Hits 5.4 Inventory Turns Per Year. *Management Consultants*.

Saw, R. (2002). *Cranfield Blocks Game*: Centre for Logistics and Supply Chain Management (CSCM), University of Cranfield, UK.

Schrage, M. (2000). *Serious Play: How the World's Best Companies Simulate to Innovate*. Boston, USA: Harvard Business School Press.

Schuelke, C. (2001). Data Warehousing Horizons: Best Practice Approaches to Operational, Tactical and Strategic Reporting. *DM Review Magazine*.

Sethi, A. K., & Sethi, S. P. (1990). Flexibility in Manufacturing: A Survey. *International Journal of Flexible Manufacturing Systems, 2* (4), 289-328.

Slack, N. (1983). Flexibility as a Manufacturing Objective. *International Journal of Operations and Production Management, 3* (3), 4-13.

Small, R. D., & Edelstein, H. A. (1997). *Scalable Data Mining*. Two Crows Corp. Retrieved June 2005, from http://www.twocrows.com/whitep.htm

Soderquist, C., & Harris, B. (2001). *Can We Prevent the Next Round of Layoffs?*: At Any Rate. Retrieved July 2005, from http://www.pegasuscom.com/AAR/model2.html

Swaminathan, J. M., Smith, S. F., et al. (1998). Modeling Supply Chain Dynamics: A Multiagent Approach. *Decision Sciences, 29* (3).

Taylor, S. J. E., Fujimoto, R., et al. (2002). *Distributed Simulation and Industry: Potentials and Pitfalls.* Paper presented at the 2002 Winter Simulation Conference.

Taylor, S. J. E., Sudra, R., et al. (2002). GRIDS-SCF: An Infrastructure for Distributed Supply Chain Simulation. *SIMULATION, 78* (5), 312-320.

Tranouez, P., Lerebourg, S., et al. (2003). *Changing the Level of Description in Ecosystem Models: an Overview.* Paper presented at the The 2003 European Simulation and Modelling Conference, Naples, Italy.

Turner, S. J. (2001). *A Generic Architecture for Large-Scale Distributed Simulations* (PowerPoint presentation). Singapore: UKSim.

Veith, T. L., Kobza, J. E., et al. (1998). World Wide Web-based Simulation. *Int. J. Engng Ed., 14* (5), 316-321.

Vijayan, J. (2001, MAY 07, 2001). *Inventory Turns*. COMPUTERWORLD. 2005, from http://www.computerworld.com/industrytopics/retail/story/0,10801,60182,00.html

Weisstein, E. W. (1999). *ErrorPropagation*. From MathWorld - A Wolfram Web Resource. Retrieved 2004, from http://mathworld.wolfram.com/ErrorPropagation.html

Zeigler, B. P., Kim, D., et al. (1999). *Distributed Supply Chain Simulation in a DEVS/CORBA Execution*

Environment. Paper presented at the Winter Simulation Conference, USA.

CHAPTER 3 101
THE FLEXIBLE SUPPLY CHAIN APPROACH 101
3.1 INTRODUCTION 101
3.2 THE BASIC IDEA 102
3.3 A GENERIC NETWORK CONCEPT 103
3.3.1 The customer-supplier-unit 105
3.3.2 Connecting the CSUs 106
3.3.3 More about delivery 107
3.4 MODELLING THE CSU COSTS 108
3.4.1 Prices and costs 109
3.4.2 Purchasing costs 109
3.4.3 Stocking costs 110
3.4.3.1 Deeper into the holding costs 111
3.4.4 Manufacturing costs 112
3.4.5 Delivery costs 113
3.4.5.1 Another method for transport costs 113
3.4.6 Management costs 114
3.4.7 Total and global costs 114
3.4.8 The base price of the products 115
3.5 INFORMATION AND MONEY FLOWS 115
3.6 PRODUCTS AND PRODUCT-BOXES 116
3.7 MODELLING THE CSU RESOURCES 117
3.7.1 The stock 117
3.7.1.1 General information 117
3.7.1.2 Ordering policy 118
3.7.1.3 Technology 120
3.7.2 The fleet 121
3.7.3 The production section 121
3.7.3.1 Manufacturing policy 122
3.7.4 The management section 123
3.8 MODELLING THE CSU ACTIVITY 124
3.8.1 Process of purchase (or ordering) 125
3.8.2 Process of stocking 125
3.8.3 Process of delivery 127
3.8.4 Process of manufacturing 128
3.9 PRODUCTS, ORDERS AND VEHICLES 128
3.9.1 About the system products 129
3.9.2 The material-order structure 129
3.9.3 Modelling the vehicles 130
3.10 ABOUT THE FLEXIBILITY (MATRIX) 131
3.10.1 The rigidity concept 131
3.10.2 Matrix representation 133
REFERENCES: 135

Chapter 3

The Flexible Supply Chain Approach

An object oriented modelling design

3.1 Introduction

As we have seen in the previous chapter, developing a model commonly implies the built of a representation of the system with the help of general purpose primitives, like the basic blocks of logic used in *Petri nets*, *System Dynamics*, UML, etc., or even the wide spread *state transitions diagrams*, which many commercial tools of simulation use. Building a model implies therefore the appropriate ability to construct based on such blocks. Of course, today it is easy to agglomerate a certain number of these blocks and assign them the "self" of an "entity", using the properties of inheritance of the modern object oriented languages, but even so the modeller must have some knowledge about the basis behind those approaches, since it is frequently necessary to readjust such "entities" to the specific contours of a project.

To conceive building a model based on basic multi-purpose blocks seems to be also interesting to the developers of commercial simulation tools, as a higher number of customers can be served and the idea of being flexible enough to answer to different types of modelling requests is promoted. The multi-purpose simulator had always been an attractive approach for selling, as it is a generalist solution with which a wide range of issues can be addressed, in the various areas of knowledge.

In the reality of industry, however, a substantial number of companies purchase these kinds of generalist tools with the intent of addressing specific problems and only later realise that it would be also necessary to contract a modeller specialist, since the system to model reveals itself to be considerably more complex than expected. The usual destiny for such modelling systems is therefore the wastebasket. Exceptions exist, of course, among companies with a high capacity for management and contracting, as is the case of BMW, for instance, which headquarters I had the privilege to visit some years ago in Munich, and where a group of engineers is specifically dedicated to model and simulate the manufacturing processes. This emphasis on the simulation as a matter for specialists seems an interesting policy, and for sure the results emerge accordingly. Besides, the company was using *eM-Plant*, a tool for modelling and simulation which already allows the usage of some pre-defined "real" elements with which the engineer can construct the model; somehow similar to what is proposed in the approach described herein.

The approach presented in this chapter also aims to dissolve the need for special skills to build a Supply Chain model, while at the same time maintaining the need for reasonable skills in managing the simulation process. In this sense, the entities of the Supply Chain are represented as much as possible as they appear in the "reality". Thus, no special knowledge is needed to construct a model other than that of creating a replica of the system based on "scaled elements of the reality". This can probably turn out to be very interesting for those Supply Chain specialists who have few skills in modelling.

In fact, each object is intended to be seen by the user as an imitation of an object that exists in reality, with a certain functionality encapsulated, and not exposed to the modeller. Therefore, the level of abstraction required in the process of modelling comes closer to that of a planner than to that of a specialist on modelling. At this level, one does not need to define or represent the dynamics of the internal processes of the Supply Chain elements, since they are pre-defined in the respective objects of the application. The user only has to configure or tailor these generalised objects in order to represent the particular Supply Chain case with the accuracy needed. This approach can also be considered a "*data-driven*" approach, since there is no need for programming in order to model or to simulate the system.

Finally, the present proposal also reflects the tendency of providing the industry with simulators for specific areas of knowledge, imitating the natural specialisation observed in the business world at the enterprise level. As there are companies specializing in airport projecting, urban traffic design, nuclear central building, etc., our point of view is that it would be advantageous to reproduce the same tendency in industrial simulation tools, instead of systematically developing the models from the basis. This would provide the modelling community with an extensive and interesting "data base" of specific area simulators available to any specialist planner. A huge data base of highly reliable specific area simulators is what we modestly intend to contribute to.

3.2 The basic idea

The concepts used in this approach appear not only from the object perspective taken from the Supply Chain system, but also from the intention to synthesise the behaviour of such systems in a more general way, in particular in what concerns the processes and the events that represent the business activity of suppliers, factories, warehouses, retailers, and, at the end of the chain, last customers. Those events are what trigger the flow of materials and information in the chain.

Unlike some static approaches normally used in simulation for strategic planning, where parameters like the lead time or the average rate of material flow are held as inputs to the system, the ideas described here consider a representation of the Supply Chain in which those parameters result from some dynamically and detailed simulation process. It allows, therefore, the ability for modelling starting from the less abstract processes till the more abstract point of view. The result is a more flexible method for building the model and a more realistic picture of the dynamics involved in the system.

Due to this fact, this approach could probably be seen as being more directed to tactical managers than to strategists, but the intention is that the models constructed with it can be used either to simulate short or long periods of time, therefore revealing also their value for strategic purposes.

The basic idea behind this modelling proposal is to consider the flow of products, of information and of money between any elements in the chain as a general form of *customer-supplier* exchange activity, and also to treat each of these elements as inheriting from a single element which includes the basic behaviour (and resources) of a generalised node of the Supply Chain, that is, simultaneously of a retailer, a warehouse, a factory, a supplier and even a last customer. A description of such a generic element will soon be made, showing its structure as well as the costs associated with its various processes. As we will notice, some parameters usually considered inputs by other modelling techniques will appear here as outputs, providing the analyst with more interesting data for evaluating the Supply Chain performance.

3.3 A generic network concept

During the last decades, most of the approaches used to model the Supply Chain have been mainly devoted to study the problem of dimensioning and geographically positioning the components of the chain in a way that the overall expected costs could be calculated and, maybe, optimised. Such approaches were commonly dealing with average values computed over long periods of time. In a certain way, the intent of those models was to make the results independent of the time variable, and so, what one could retrieve from them was also a kind of static expectation. Although these methods are still very important in the first step of establishing the structure of the Supply Chain on the ground (strategic), they could not produce enough reliable results in order to help the tactical or the operational manager. That is, they hardly survived the need for describing the internal dynamics of the network, or even the weekly need for optimising the processes involved in it. To address issues of these kinds the Supply Chain begun also to be modelled in a dynamic way. The present approach must also be considered one more perspective to enrich such trials.

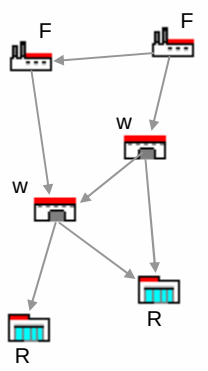

Fig. 3.1 The common Supply Chain structure

As figure 3.1 suggests, the Supply Chain is normally considered by most Supply Chain simulators a kind of network connecting factories (F), warehouses (W) and retailers (R), and each of these elements is treated as exhibiting different internal structures and different behaviours. In the factory, for example, there is expected to run a process of transformation which gives the raw materials the aspect of products; in a warehouse there are only storage and handling processes; and in the retailer the

main activity goes to the action of sales. Then, products are what flow through the network transported by vehicles of some sort.

Here, however, we consider the elements like factories, warehouses, retailers, etc., inheriting from a single generic element that encapsulates a generic *customer-supplier* exchange activity. That is, a basic element that can be transformed into any particular element of the Supply Chain by simply configuring its internal structure and parameters. We decided to name this element *customer-supplier-unit* (CSU), to reflect this idea. Based on this CSU, it will even be possible to model the figures of the primer supplier and the last customer, as we will see later on, usually not considered in simulators dedicated to the Supply Chain.

The overall Supply Chain structure will therefore be transformed into a group of CSU nodes connected together by a network of transports, where there can also be exchange of information (Fig. 3.2).

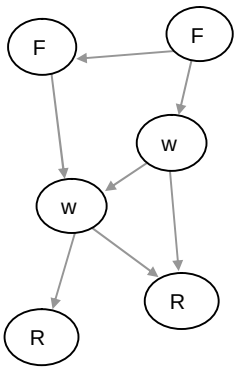

Fig. 3.2 CSU equivalent of a generic Supply Chain

Now, what will distinguish a factory from a warehouse or from a retailer, or from any other nodal element, will only be how that particular element uses the resources and the processes of its CSU and the kind of materials it handles as input and output.

We can now think of each CSU as an element living from a constant exchange with its neighbours, on one side receiving from them materials and information and returning money to them, on the other delivering to them material and information and receiving money from them. This is, obviously, the perspective of a market. And each CSU already represents the model of a complex differential equation.

In order to be able to inspect how the economy develops among the elements of the Supply Chain, a reference to its own economic account will be included in the basic structure of the CSU. One must not forget that most management decisions are taken not only based on technical performances but also on how money spreads along the systems. This account, of course, will include costs and incomes, in a way that by the results of the simulation one can better predict how each of these elements is contributing to the overall economical performance of the chain.

Notice that in the primary point of view of this flexible[1] proposal, the Supply Chain is already established on the ground, that is, the network structure is predefined. The sort of problems to be studied or analysed are therefore related to the management of the physical processes actually running within that structure. These processes are the dynamic processes of purchasing, stocking, manufacturing, delivering and managing. Each of these processes is

[1] Here the classification "flexible" is related both to the ability for representing a high variety of Supply Chain cases and the ability for measuring the Supply Chain flexibility in certain cases, as we will see later.

treated in terms of measures of technical performances but also in terms of economical indexes, whenever possible. This aspect appears very useful to help the manager in making practical decisions, in particular when the Supply Chain under study contains elements belonging to different companies, thus expressing a network of concurrent interests. By simply representing the flow of money in the simulation, for instance, one can easily visualize the importance of each element in the network, or even compute the costs and the incomes for a particular group of elements, an interesting feature for analysing complex horizontal multi-company examples.

3.3.1 The customer-supplier-unit

The basic *customer-supplier-unit* (CSU) is then a conceptual element with which it is possible to model any sort of Supply Chain facility. This element "owns" a group of customers, to whom it delivers material and from whom it receives money, and a group of suppliers, from whom it receives material and to whom it returns money. In order to be able to handle this basic functionality, the CSU must also include in its structure some other general resources. Basically, these resources are a stock, an economical account, an input queue where the supplier vehicles will wait to deliver the materials, and an output queue where other vehicles will line up to receive the material that must be delivered to the customers.

In order to be able to manage these resources, the CSU must as well include in its structure a list of suppliers, a list of customers, a list of products and, finally, a list of vehicles representing its own fleet. Notice that in this approach we establish as a principle that the fleet always belongs to the one who delivers, what seems a reasonable consideration for most practical cases, even when the fleet is hired, as we will see later.

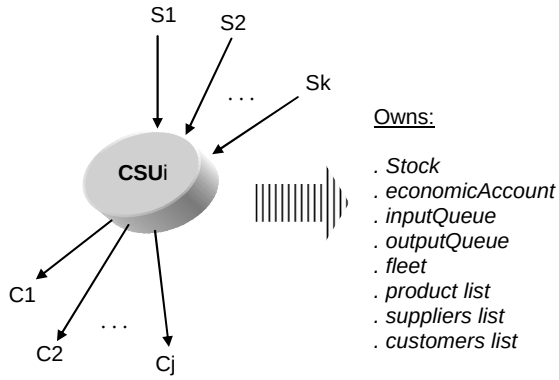

Fig. 3.3 Generalised concept of a CSU with *k* suppliers and *j* customers

Considering the sort of structure depicted in figure 3.3, the CSU becomes a general element ready to be linked to other elements of its kind, thus making possible the overall Supply Chain representation. Of course, different CSUs can be made to handle different amounts of stock, different numbers of customers and (or) suppliers, different products, etc., depending on the role they play in the Supply Chain. For instance, ultimate customers can be thought as CSUs having null stocks, null fleets and a null customer list, being therefore reduced to generators of the demand of products and the sources of money for the Supply Chain. On the other hand, a supplier source can be seen as a CSU having infinite stock and a null supplier list. Notice also that raw materials can be treated as simple products, even if later it will probably be necessary to adapt the unity of transport to the proper case. Thus, this approach also opens the Supply Chain

simulator to the representation of ultimate customers and primer suppliers, which in most simulators are simply treated as statistical parameters.

It is important to indicate, however, that the final CSU structure is substantially more complex that the one described here, since it includes some other features related with production and many others variables and methods with which its internal dynamics are represented.

3.3.2 Connecting the CSUs

In order to model the Supply Chain dynamics it is now necessary to ensure the connection of each CSU to its effective neighbours. This is ensured by a network of "physical" paths in which several kinds of vehicles can move. These paths were designed to represent normal roads, highways, railways, boat circuits, airlines, and even pipelines. Apart from these, an information path was also considered and implemented in the simulator, with which materials can be ordered directly to any facility of the Supply Chain, somewhat like what happens in the order-by-catalogue and order-by-Internet. For the purpose of this section, however, we will assume that general-purpose ground vehicles will be responsible for moving the materials, as suggested in figure 3.4.

Notice that each path can represent different costs to the process of delivery. For instance, vehicles can first run on a highway and then change to normal roads. This aspect, and the probable existence of road junctions in some practical cases, allow us to consider using also the concept of transport *node* (see Fig. 3.4).

Vehicles will travel from node to node with the objective of executing a certain predefined job. And this also means that each CSU must include the dimensions of a node. Of course, as the vehicle moves from one node to the next node the delivery costs are expected to rise, since they are calculated based on the previously known costs of vehicle utilization per kilometre and driver cost per day, among others.

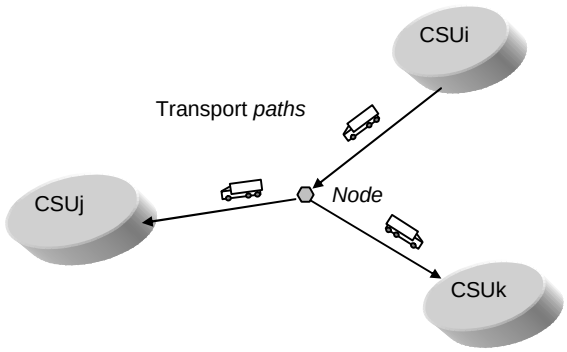

Fig. 3.4 Transport paths connecting supplier CSUi to customers CSUj and CSUk

Notice also that, in principle, the time spent by the vehicles delivering on these paths will not be easily predictable, also because different vehicles from different suppliers can reach the customer around the same time, and then it will be necessary to queue before the effective delivery will take place. Situations like this will be, of course, dependent on the layout of the chain, on the quantity of suppliers serving each customer and on the intensity of the delivery flow in the network, as well as on the delivery policies and on the size and number of vehicles.

Another important notice is that each vehicle is treated as an independent entity, therefore the movement of a vehicle through these paths will be simply dependent on the vehicle itself; that is, it will result from the execution of a certain sequence of events that are the "property"

of the vehicle object. Thus, from a higher-level point of view, there are only two commands (events) each CSU has to send to its vehicles: *startDelivery* and *startReturn*. The first must be sent after attributing a certain job to the vehicle, and the latter when a job must be finished. The vehicle itself must then be capable of delivering the material without any other interference from its CSU, even if a job includes more than one delivery point, as in the case of local delivery, for example.

3.3.3 More about delivery

This concept of modelling also interferes with the speeds of delivery observed in the simulation. Unlike what is expected when using stochastic variables for representing lead times, here the delivery speeds will exhibit a complex behaviour due to the fact that they are also dependent on the operations running in the CSU at a particular moment. In effect, after the material is ordered to a particular CSU, a sequence of actions will begin in that CSU, leading to the processing of the order, then to the packaging of materials and finally to the start of the transportation process. From the perspective of the CSU customer, the speed of the delivery will therefore depend on the time spent on such sequence of actions and even on the criteria used by the CSU supplier to assign delivery jobs to its vehicles. In reality, the effective lead time will result from the summation of a series of delays that in principle are not computable a priori. In addition to this, the time to carry the materials along the paths to the CSU customer must be considered, which of course is dependent not only on the vehicle speed but also on the limit speed of each path along which the vehicle will travel, for example. The figure 3.5, where 3 different paths and 4 nodes connecting supplier CSUi to customer CSUj are represented, helps in imagining the complexity that can affect lead times due to these issues, and mainly in understanding how one can expect the results obtained with this process to be different from those obtained if lead times would simply be modelled by stochastic variables.

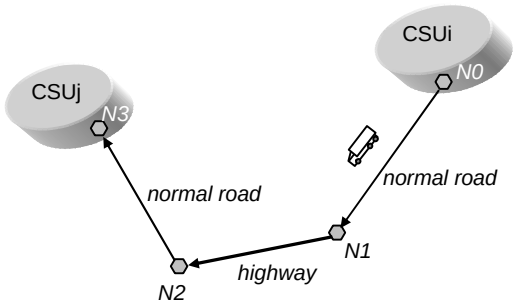

Fig. 3.5 Paths connecting supplier CSUi to customer CSUj including normal roads and highway

In addition to the irregularity introduced in the delivery times by the CSU internal operations, one must consider the source of unpredictability related with the process of transportation itself, mainly when the delivery (or distribution) structure is to include diverse types of transports or diverse paths. In certain practical cases, the materials must be transferred to different vehicles at a certain point on their way to the customer. *Deutche Bahan*, to cite an example, recently pondered the idea of using its own railway structure to help in distributing the spare-parts needed by the manufacturers of locomotive engines, creating points of interface with other kinds of vehicles in certain strategic locations. It is also usual that intercontinental flights are used as links for connecting different and distant Supply Chain structures, as ZARA,

Benetton and other companies practise. Situations like these have made us decide to include in this modelling approach the primitives necessary for representing a more realistic transport model.

We have assigned a limit speed to each transport path, as well as a certain vehicle speed to each vehicle; so, the effective speed of transportation in that particular path can simply be calculated as:

$$transportSpeed = Min(pathSpeed, vehicleSpeed) \quad (3.1)$$

It is therefore considered that each time the vehicle reaches a new node leading to a new path it will use this expression to recalculate its next speed of transportation. This new speed will be the speed to travel along the new path. Notice also that, as it is easily understood, the limit speed assigned to a path can serve not only to model the legal limit speed of that transport connection but also to model the usual conditions of traffic flow in it. Highways crossing the interior of big towns can be modelled with a very low maximum speed, if they are usually congested.

In order to model such a variety of situations associated with the delivery of the materials, the following four types of nodes were considered as elements of intersection for transport paths: **(1) normal node**, if the node just represents a point where vehicles will change to a new path. This is what happens in a passage from a highway to a normal road, for example. Once the node is reached, the vehicle will enter the next path on the way to the current customer. In the case of a node with several exit paths, the vehicle will analyse the situation comparing each of those paths with the paths associated with the actual delivery job; **(2) CSU node**, if the node is coincident with a CSU, that is, with a facility of the Supply Chain. This node has precisely the same function as a normal node, except when the associated CSU is the customer of the present delivery job. In this case, the vehicle enters the queue of vehicles for input (*inputQueue*) of that CSU and waits to be unloaded. Only after this will the vehicle start the return to its CSU or origin; **(3) transfer node**, if the node is used to transfer material from one vehicle to another. This sort of node can be used when there is the need for interfacing two different transport modes, for example, an air-cargo or sea line with the land distributing vehicles (trucks, trains, etc.) serving a region. The transfer process implies costs per unit of product transferred, which are specified in the node properties; and **(4) pause node**, if the node is to represent a certain delay that must be introduced in the delivering process. Usually this node is used to model meal times, sleeping times, etc., with which costs are automatically associated.

In complex Supply Chains structures, these issues can strongly contribute to the unpredictability of the speeds of delivery observed in the simulation.

3.4 Modelling the CSU costs

Before examining in more detail the internal structure of the CSU and the processes related to its dynamics, we decided to expose the ideas behind the representation of the costs associated with this kind of elements. The reason for this is to liberate the discussion as soon as possible

from the well known economical issues, and to move to other interesting aspects more directly concerned with the simulator development.

Since the Supply Chain can now be represented as a dynamic system, the associated costs must also result from certain dynamic processes, that is, costs must be calculated and afforded to the respective CSU by means of certain event executions in time. It is, therefore, important not only to know the amount of a particular cost but also the time when this cost must be afforded to the CSU. This will also allow, for example, a continuous calculation of costs and incomes during the simulation process, offering the Supply Chain manager the important feedback of certain economic indicators.

Based on the idea that each process must have its costs, and again considering that the main processes running in the CSU are purchasing, stocking, manufacturing, delivering and management (see figure 3.6), we have decided to specify costs also following this same taxonomy, as it fits perfectly with the nature of the operations running in the system.

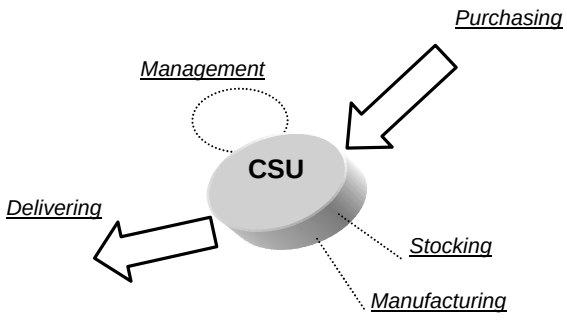

Fig. 3.6 Main processes for assigning costs to the CSU

We will consequently talk of purchasing costs, stocking costs, manufacturing costs, delivery costs and management costs, each of them being thought of as the result of other more specific costs, some assumed fixed and others considered of the variable kind. Yet, it is important to bear in mind that the more the costs are detailed the higher is the number of events that must be added to the model. In this approach we will be only interested in those costs that usually are considered more relevant, and each of them will be handled as being an expression of a fixed and a variable component. The fixed component will express the price to pay for the "existence" of the process, while the variable will represent the price to pay for "operating" it, or for maintaining its "activity".

3.4.1 Prices and costs

There is sometimes some confusion about the terminology used to specify the *value of the product* when it moves along the Supply Chain. To establish the nomenclature from now on, it is assumed that the *product cost* is the value that must be paid by the CSU customer to the CSU supplier for the product, while the *product price* is the value by which the CSU supplier sells the product to the CSU customer. As we see, the two values are in fact the same, but the terms *price* and *cost* must not be confused.

3.4.2 Purchasing costs

These costs must be afforded to the CSU at the end of each purchasing cycle, that is, each time the products (or materials) are received from (and paid to) the supplier. The amount of these costs depends on the price the supplier asks for the products and on the administration involved in the purchasing process, which is mainly related with the costs of ordering the materials.

This aspect is modelled as a cost per purchasing cycle, and as we can see it includes the idea of the "setup cost" mentioned in the inventory management literature. So, the purchasing costs will occur at the end of each purchasing cycle and its amount will be given by:

$$PurchasingCost(t) = \underline{ProductsPrice(t)} + \underline{administration(t)} \quad (3.2)$$

Where administration is associated to invoice preparation, papers, phone-calls, etc., and also to costs with personnel. Notice that the purchasing costs are here considered only as variable costs. Their fixed component will be included in the fixed costs of maintaining the stock structure. This was viewed as a reasonable approach and even appropriate, since in practice the existence of a purchasing process only makes sense in the case of the existence of a stock structure.

3.4.3 Stocking costs

The process of stocking involves diverse aspects, and so, diverse kinds of sources for costs can be assigned to it. Some of these sources commonly referred in the literature (Sussams, 1992, pp. 55) are the operations, which include the receiving, the order picking, the inventory replenishment, etc.; the administration, which includes the processing of the orders, the managing of accounts, personnel, etc.; the occupancy, related to rents, services, insurance, depreciation, maintenance, etc.; and the inventory (or holding) costs, which are due to the existence of the products in stock, usually 1% above the bank base lending rate. The overall stocking costs can then be expressed as:

$$StockingCost = [\underline{operations(t)} + \underline{administration(t)}\ ] + \\ + \underline{occupancy} + \underline{holding\ cost(t)} \quad (3.3)$$

To simplify the representation of these costs in the simulation we will call stock processing costs the operations and the administration costs together, which are variable costs computed per unit of material handled. Then, the occupancy costs will be affected continuously as the time increases, and will be based on a fixed cost per unit time (a fixed amount per month). Finally, the holding costs will be computed based on the value of the money invested in the products actually in hand, no matters whether they are in house or in movement to the clients. This means that *in-transit* inventories are considered in this approach.

In general, the holding cost for the i^{th} product in the inventory exhibits the following dependence with the time:

$$holdCost(t)_i = cp_i\ h_i\ Q_i\ (t - to) \quad (3.4)$$

where:

cp_i = cost of a unit of the product i^{th}
h_i = bank lending rate per unit of time and product
Q_i = quantity of products i in the inventory
to = time when the product was purchased
t = present time

It is important to notice that the holding costs begin when the CSU pays the product to the supplier and will only end when the CSU receives the payment for delivering it to the customer, hence the importance of considering *in-transit* inventories.

Apart from these costs, it is also usual to

consider the costs associated with the stock breaking (Gonçalves, 1999) each time the demand cannot be satisfied from the stock, a situation known as *stockout*. Although these variable costs are difficult to estimate due to their faulty characteristics, which can even interfere with the subsequent demand, here we will approximate them to the quantity of material that have not been served to the client. In addition to this, a measure of the "satisfaction" of the client is used, which represents the time spent to serve the client compared with the time the client accepts to wait for the materials to arrive. This leads to the representation of the "customer satisfaction index" referred to in the previous chapter (see equation 2.7).

3.4.3.1 *Deeper into the holding costs*

Since the holding costs are a special case of expenses, we will dedicate a little more time to them. In reality, these costs are obviously interrelating suppliers and customers, since everybody wants to get rid of the materials as soon as possible to minimise the risk of the investment. It is here that the materials can be compared to hot potatoes or cartoon time-bombs, as mentioned in the previous chapter.

The holding costs are made to begin the moment the CSU issues the payment to its supplier for the materials received[2], and will vanish the moment the materials delivered are paid back by the customer. That is, only after the process of delivery is finished can the holding costs fall.

The event of receiving material implies therefore the actualization of the holding costs both in the *receptor* (CSU_r) and in the *sender* (CSU_s). In the first, the holding costs are increased, while in the second the holding costs are lowered. If the quantity of product transferred is *dQ*, the holding costs per unit of time will increase by the amount $k_r.dQ$ in the CSU_r and will decrease of $k_s.dQ$ in the CSU_s, where k_r and k_s are the actual holding costs per unit of time and unit of product related with the product in cause, given by:

$$k_r = cp_r \,.\, h_r \qquad (3.5)$$

$$k_s = cp_s \,.\, h_s \qquad (3.6)$$

Notice that the actual costs (*cp*) of the product can obviously be different in the diverse facilities, and in certain cases even be dependent on the time. Figure 3.7 shows an example of holding costs calculation along the time.

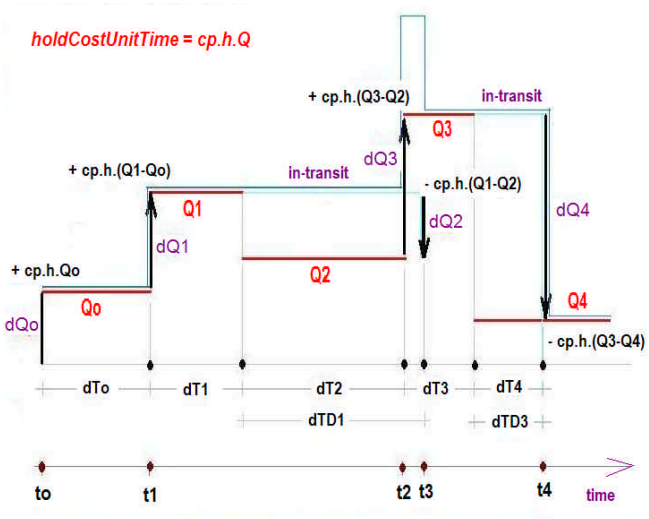

Fig. 3.7 Dynamic calculation of the holding costs

Such an example helps in calculating dynamically the holding costs of a product in each event of receiving material, taking in account the time spent in the delivery (*in-transit* inventory). From this figure one can conclude that the holding costs in the

[2] For simplicity, we consider that the moment of issuing the payment coincides with the moment of receiving the materials into the inventory.

instant t_j can easily be calculated from the expression:

$$holdCost(t_j) = holdCostUnitTime(t_{j-1}) \times (t_j - t_{j-1}) \quad (3.7)$$

where

$$holdCostUnitTime(t_{j-1}) = cp \times h \times Q_{j-1} \quad (3.8)$$

And, with a little more manipulation, deduce that:

$$holdCostUnitTime(t_j) = cp \times h \times (Q_{j-1} \pm dQ_j) \quad (3.9)$$

The sign (+) in this equation will be used in the CSU receptor, while the sign (-) will be used in the CSU sender. In conclusion, the holding costs related to a product can be estimated dynamically in the simulation process as long as one keeps track of the previous inventory level and the actual amount of the product that is exchanged between the CSU sender and the CSU receiver. This was the method chosen in the present simulation approach, since no fractionary delivery was considered.

3.4.4 Manufacturing costs

In our perspective of modelling the CSU, we consider the factory a facility similar to the warehouse, with the differences that the factory will handle three kinds of products (R for raw materials, P for parts and P+ for secondary products) in its inventory and run a process responsible for the transformation of R and P products into P+ products. This process is the *manufacturing*. Obviously, P+ products are related with R and P products by means of a MRP tree. So, the factory can be seen as a warehouse with a frequent rearrangement of products in its inventory. Such a rearrangement is due to the *manufacturing process*[3], represented in the diagram of the next figure (Fig. 3.8).

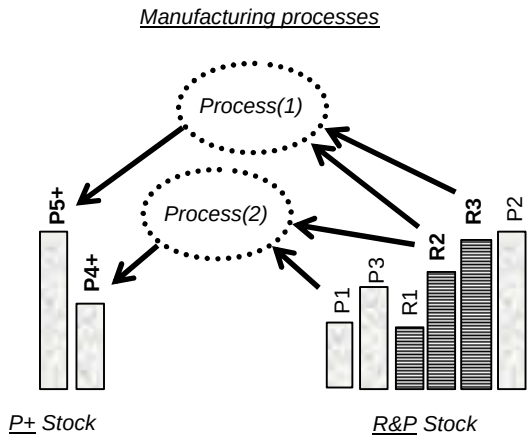

Fig. 3.8 Equivalence of a manufacturing process and a rearrangement of inventory

As it can be deduced from this figure, the variable costs associated with each of these processes will essentially be the costs of transforming R and P products into P+ products. On the other hand, those costs associated with labour and with the space occupancy of the production area can be considered of the fixed type. Notice that the costs related with the stocking process that supports the manufacturing activity are precisely the same referred to in the previous section, since the inventory of the CSU is here considered a common resource for all the processes running in the facility.

The transforming costs must be specified for each product P+ produced in the factory and given per batch of production (lot). These costs must be afforded to the CSU at the end of each production cycle. Thus, after a certain period of time *t*, the contribution of the manufacturing process for the overall variable costs of the CSU will

[3] This can also be called *production process*.

be given by:

$$ManufactCost(t) = \sum [\underline{transforming}(t)]_i \quad (3.10)$$

where i refers to the i^{th} product P+ manufactured during this period of time.

3.4.5 Delivery costs

We consider the delivery costs a direct consequence of the existence of a fleet of vehicles attributed to the CSU. Once again, these costs are the result of the following contributions: the occupancy, which is related with what is needed to maintain the fleet, like garages, machinery, technical personnel, etc.; and transport, which includes the costs of drivers, fuel and lubricants for the vehicles, road charges, meals, delays, material handling, etc.

Similar to what was considered in the stocking cost, the occupancy cost is seen as a fixed cost, and therefore given per unit of time (usually monthly). The transport costs, however, as a sort of variable cost, will be computed during the simulation process each time a vehicle reaches a new node in the transport path network. These costs will then be assigned to the respective CSU when the material reaches its destiny. We therefore have:

$$DeliveryCosts = \underline{occupancy} + \underline{transport(t)} \quad (3.11)$$

The transport component includes aspects like fuel, lubricants, maintenance and roads, as costs depending on the distance travelled (*ds*); drivers, depending on the time travelled (*dt*); and meals and delays, as constant costs. As an example, if a vehicle is going from node *A* to node *B* with a speed *v*, the following transport costs will be added to its CSU when the node *B* is reached:

$$Fuel(ds) = (VehicleFuelConsumption/PricePerLitre)*ds$$
$$Roads(ds) = (PriceOfRoadPerKilometre)*ds$$
$$Driver(dt) = (DriverPricePerUnitTime)*(ds/v) \quad (3.12)$$

Where lubricants and maintenance are considered included in the cost of the fuel. Notice that when the node corresponds to a node where the driver will rest or have a meal, the constant costs of "resting" or of "meal" must be added as well.

Notice also that hiring the services of an external fleet can simply be modelled by keeping the costs of occupancy null, if those services are contracted as variable, or by specifying those costs as fixed and included in a sort of occupancy, if they result from a contract of a fixed amount per month, for example.

3.4.5.1 Another method for transport costs

The previous method for assigning the transport costs to the CSU implies a detailed knowledge about the vehicles used in the delivery process, as well as the itineraries for the customers. Although it is not so hard in practice to obtain this type of information when modelling a case of *distribution*, that is, when the model is built from the point of view of the *sender*, the same is not true when modelling from the perspective of the *receiver*, that is, in a typical case of *procurement*. The reason for this is that in the first case the transport resources belong to the system to be simulated, while in the second case they usually do not, and this kind of information is difficult to obtain from external companies (the suppliers). In such cases,

the transport costs are in fact usually "unknown" for the customer, and merely reflected in the unitary cost of the product purchased. The price of a unit of product decreases when ordering a higher quantity of product units, and vice-versa. Of course, the transport costs are indirectly reflected in the cost of the product for the CSU client, but no detailed data is available as in the latter case.

To model these cases we have decided to attribute to each vehicle a table of prices relating the space occupied by the materials ordered and the overall space available in the vehicle. In reality, this is how things usually work in practice, and so this will definitely be useful in several situations.

So, each vehicle belonging to the fleet of the CSU supplier will have a list of *echelons* with the respective prices for transportation to a certain destiny, as depicted in figure 3.9.

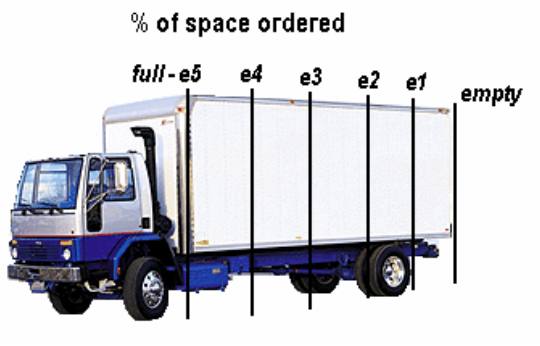

Fig. 3.9 Echelons of transport costs to the customer

As we can see from this figure, we have decided to distinguish 5 different echelons between vehicle empty (0%) and vehicle full (100%), leaving the user free to specify for each of them, both the percentage and the price per unit of product transported. The transport costs are by this way transferred to the prices of the products.

Notice that any data about road charges, drivers, meals, fuel consumption, etc., is now unnecessary. Models can be created even without considering such specific information. This is part of the flexibility of the present modelling approach, in what concerns the representation of Supply Chain systems in diverse situations.

3.4.6 Management costs

Management costs, which can include the costs of managers, analysts, secretaries, consultants, marketing, etc., will be here considered a kind of *fixed costs*, and, in this first approach, will be integrated in the fixed costs of the main resource of the CSU, that is, the inventory. These costs will also be afforded to the CSU as a continuously growing cost.

3.4.7 Total and global costs

Each of the previous costs are recorded during the simulation as instantaneous values, thus, the calculation of their totals will automatically result from the simple summation of those values along the time domain. For instance, considering the *variable transport costs* the time series of values $T(t_i)$ computed at the instants t_i, which can also have the form of a data record put out by the simulation, the *total variable transport* costs will simply be given by the accumulation of these values along the time, that is:

$$Total = \sum_i T(t_i) \qquad (3.13)$$

This is used in this modelling approach as a simple and efficient method of recording the relevant variables along the simulation run, and allows the simplification of many

future calculations. For instance, it makes the calculation of *global costs* extremely simple, not only in a particular CSU but even in the entire network of the Supply Chain. In effect, it holds for any sort *k* of costs in the CSU:

$$Global_{csu} = \sum_k Total_{csu,k} \quad (3.14)$$

and:

$$Global_{supplyChain} = \sum_{csu} Global_{csu} \quad (3.15)$$

Since each facility in the Supply Chain model has this structure of costs integrated in its CSU object, the computation of such quantities becomes extremely easy in any moment of the simulation process.

3.4.8 The base price of the products

Costs would never be useful without a previous definition of the *value* assigned to the products. On the other hand, in reality different members of the Supply Chain can establish different values (prices) for their products, based on indexes like demand and concurrency, for example. Therefore, the price of a product in the modelling process would have to be dependent on the CSU, meaning that different prices would have to be inserted in each CSU when creating the model of a Supply Chain. If the Supply Chain would have *N* facilities and *P* different products, the number of initial product configurations needed would simply be *N* x *P*. Taking *N* = 10 and *P* = 100, for example, this would lead to 1000 different product configurations, and a lot of time needed for inserting the initial conditions into the models.

To surpass this problem, we decided to choose a simpler perspective: (1) establish a base price[4] for each product available in the Supply Chain; and (2) assign to each CSU a typical business margin[5] for its product exchange activities (by default 10%). This way, the number of product configurations needed is practically reduced to *P*, the number of products in the system, if most of the CSU margins are kept the default. This also introduces a kind of *chain of value* for the materials moving in the direction of the last customers, another interesting aspect that can be a subject for future studies.

3.5 Information and money flows

Other things that flow in the Supply Chain are *information* and *money*, which in the present approach we consider associated to quite simple operations. The information will be mainly related with *communications*, either between CSUs during the ordering of materials, or inside the CSU in any process that implies the transfer of data, for instance when filling the time gap between the arrival of a new order and the beginning of its processing. This is modelled by means of a *typical information delay* associated to the CSU, a stochastic variable. Although we were expecting to include in this simulator some additional features about information sharing, in particular regarding the transfer of the demand upstream in the network, a procedure claimed to improve the Supply Chain performance (Mason-Jones & Towill, 1999), the fact is that no effective analytic

[4] This base price is the cost of the product at the facility that makes it enter the Supply Chain system. After that, the product price follow a *chain of value* as the product travels in the direction of the last customer.

[5] Defined as 100x(price-cost)/cost of a product, and assumed a constant of the CSU in the present version of the simulator.

processes were found in the present *Supply Chain Management* (SCM) that could be used to implement such ideas. It is, therefore, an issue that remains open for future research.

On the other hand, the *flow of money* will be modelled as an expression of the process of payment to the supplier for the materials received. Considering that money always flows in the opposite direction of the materials flow, each time one receives materials one has to pay back its price to the supplier. It is based on this elemental process of exchange (Fig. 3.10) that the Supply Chain maintains its activity through the time. The payment is considered to be instantaneous or, at maximum, affected by the *typical information delays* of the CSUs.

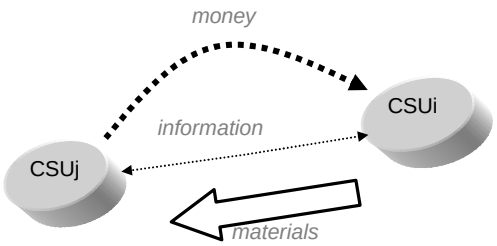

Fig. 3.10 Basic exchanging processes between supplier CSUi and customer CSUj

As we have noticed earlier, introducing this economic aspect is essential to allow the analyst to study the progress of the Supply Chain not only on a technical base but also taking into account economical aspects, as in fact happens in reality.

The money flow will be triggered each time a customer receives material from a supplier. As the chain continues its activity, such flow will spread to all its facilities, allowing the instantaneous calculation of variable costs, incomes, total costs, etc., as well as any other economical indexes, like *cash flow* or *turnover*, for example.

3.6 Products and product-boxes

It is now necessary to describe how the products will be handled in the present perspective of modelling. The properties of the products could in principle be specified the moment these were to be inserted in the stock of each facility, but this would turn to a redundancy since different facilities may use the same product in their inventories. To obviate this problem and make simpler the process of configuring the model, we have decided that the products belong to the entire Supply Chain system, and that each resource of the *stock* type consists of a collection of *boxes* where products are maintained, each *box* being associated to a single product. We call this kind of box a *product-box* (Fig. 3.11).

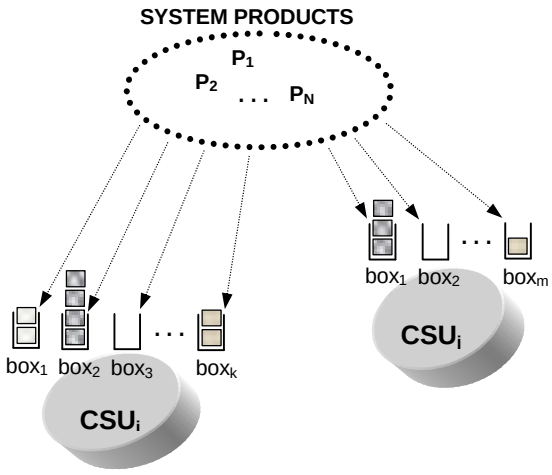

Fig. 3.11 System products and the CSU product-boxes

Thus, any stock resource is primarily seen as a collection of *product-boxes* where system *products* may later be added to or subtracted from. This led to the following two-step procedure concerning the configuring of products in the Supply Chain:

(1) products are first created in the system and their characteristics specified independently of any CSUs; (2) then the appropriate products are linked to the each CSU by the creation of an empty *product-box* for each product.

Notice that these *boxes* will later be filled in with the required quantity of product (initial level of stock) before the simulation starts.

3.7 Modelling the CSU resources

The internal dynamics of the CSU are the complex activity resultant of the response of the facility to external and internal requests, a response normally distributed by five dynamic processes: *purchase*, *stocking*, *manufacturing*, *delivery* and *managing*. As we have seen, this sort of organization is typical and implies the usage of certain typical resources, which mainly are: a *stock*, a *fleet*, a *production* section and a section of *management*. The general CSU will make use of all these resources, and the tailoring to a particular configuration will enable or disable the appropriate ones. The purpose of this section is to describe in more detail how theses resources are modelled in this simulation approach.

We will consider physical resources those that in principle require a physical space in the CSU to operate, which are the *stock*, the *production* section and the *fleet*, as illustrated in figure 3.12. The *management* does not necessary imply such a space, so it will be represented on a different level. As depicted in this figure, each of these physical resources has an associated block of management which specifically analyses the problems and search for solutions in that particular area.

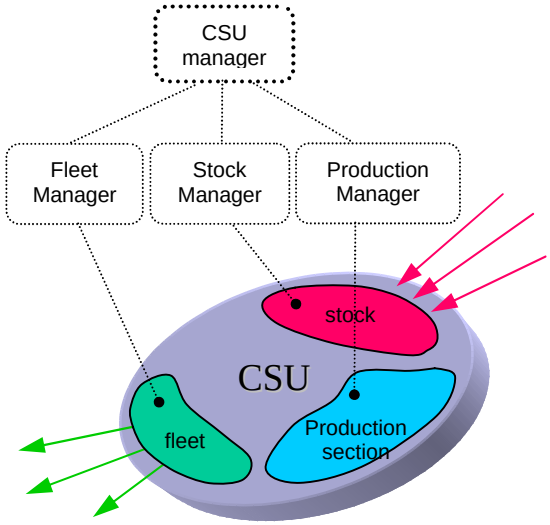

Fig. 3.12 Resources and management in the CSU

Notice that these blocks of management form part of a more general process of administration named *CSU-manager*.

3.7.1 The stock

As previously said, the stock of the CSU is a collection of *product-boxes* created and linked by the user to the products formerly launched in the system. Related to this resource are four levels of data needed to be filled in so that the *stocking* and the *purchase* processes, both related with the stock, can express their entire functionality: (1) general information, (2) ordering policy, (3) technology and (4) stocking costs. This last aspect was already discussed in the section 4 of this chapter.

3.7.1.1 General information

This is the first type of data needed to configure the stock, and is basically related to the creation and the linkage of *product-boxes* to the products of interest, that is, to those products with which the CSU will operate. Then, each *product-box* is

configured with its maximum allowable inventory level as well as with the inventory level at the start of the simulation (initial condition).

3.7.1.2 Ordering policy

The ordering policy assigned to a certain *product-box* will characterize the method by which its product will be reordered from the supplier (or suppliers). To characterize the inventory state and define the *moment* and the *quantity* of product to order we have considered the general parameters shown in figure 3.13, where t_j are the moments when the product arrives from the supplier.

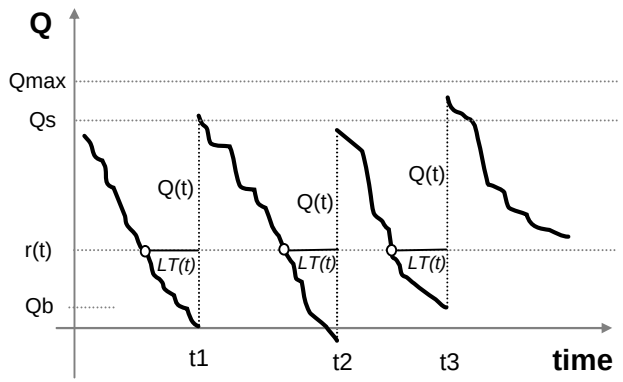

Fig. 3.13 General behaviour of a time dependent (r, Q) inventory system with r = r(t) and Q = Q(t), showing a little *stockout* around t = t2

Notice that both the *reorder level*[6] (*r*), the *quantity to order* (*Q*) and the supplier *lead time* (LT) are given the chance to be dependent on the time. This lets us count with rather more degrees of freedom in the problem than usual. Other parameters are the *base order quantity* (*Qb*), representing the base ordering unit, thus any order must be a multiple of this quantity; the *fill level of inventory* (*Qs*), as the level to which the inventory must be replenished in certain cases; and the *maximum level of inventory* (*Qmax*), which imposes the upper limit of allowable space in the stock for a particular product. Other important parameters are the *safety stock* (*ss*), used to protect the inventory from demand variations, and the *Economical Order Quantity* (EOQ), which is expected to minimise *Q* considering the demand figure, the costs of ordering and the costs of holding (see equation 2.1). Most of the current methods of reordering can be seen as particular cases of this generalised scheme.

Given that in this simulation approach we propose a continuous calculation and update of *LT(t)*, the supplier *lead time*, as well as of the *accumulated demand between supplies*, *DTL(t)*, parameter that aims to measure the *lead time demand*, both the *reorder level* and the *quantity to order* can also be updated the moment that new material arrives, and so be made dependent on the time. This means that the number of different *(r, Q)* policies that can be selected with this scheme is the combinations resulting from assuming *r* and *Q* either as fixed or as variable. For instance, at least 8 different situations could be chosen taking into account that *r* can be:

- fixed by the user as $r = ROP$
- automatically updated to $r(t) = DTL(t) + ss$

and *Q*:

- fixed by the user generically as $Q = Qo$
- fixed by the user as $Q = EOQ$
- automatically updated to $Q(t) = (Qs - Q)$
- automatically updated to $Q(t) = DTL(t)$

[6] Also referred to as *reorder point* (ROP).

Notice that if *ss = 0* the option *(r = DTL + ss, Q = DTL*) can be made to represent the common *two-bins* inventory control system, which is also the base of the *Kanban* system.

The previous considerations were valid to those inventory control systems in which the moment of ordering is triggered by the level of inventory, and therefore we call these "*byLevel*" policies. On the other hand, systems that trigger the ordering of the materials by regular periods of time, hence using what we will call "*byCycle*" policies, will be modelled considering the parameters shown in the general inventory scheme represented in figure 3.14.

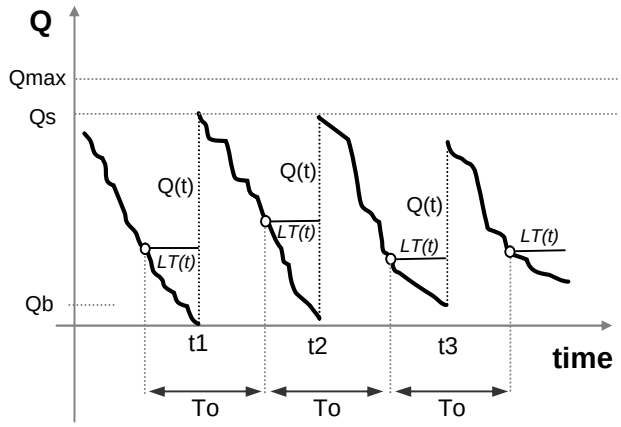

Fig. 3.14 General (R, Q) inventory system with R = To and Q = Q(t)

Here, *To* is the regular period of time between consecutive orders to the supplier, while the other parameters have the same meaning as in the last case.

Since *To* is assumed fixed and previously defined by the analyst, the number of different inventory control methods that can be modelled with this type of policies is solely dependent on the degrees of freedom of *Q*, that is, on how *Q* will be computed. In fact, different stocking dynamics will result from assuming Q as:

- fixed by the user generically as $Q = Qo$
- fixed by the user as $Q = EOQ$
- automatically updated to $Q(t) = (Qs - Q)$
- automatically updated to $Q(t) = DTL(t)$

For obvious reasons, this type of policy is also useful for inserting the demand related to *last customers* into the simulator, that is, to establish the input demand to the Supply Chain. As such demand is frequently given as a *Normal* distribution, the simulator will let the user enter data not only for *Q* but also for the standard deviation associated with it (s_Q), another form of ordering to add to those previously shown. In reality, the possibility of associating a *Normal* behaviour and a *initial phase shift* to the period of ordering (*To*) was included later as well, by adding s_{To} and t_{start} respectively, thus allowing that the variability can also be introduced along the time axis, as suggested by figure 3.15.

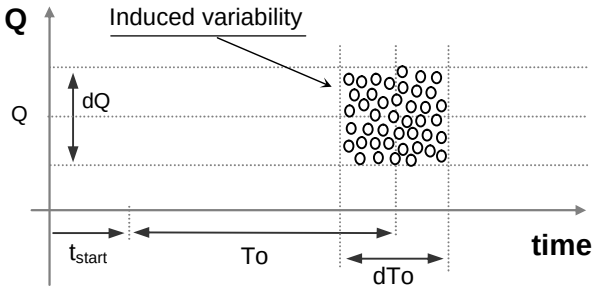

Fig. 3.15 Possibility of inducing variability in volume, and phase shift and variability in the time scale, with a "byCycle" ordering policy

The *last customer* is modelled as a CSU with no stock, no production section and no fleet, and solely represents a generator of demand by means of this type of ordering policy. Notice, however, that in any of

these ordering policies the events can also be generated by historical data, if available, and retrieved form a file in the appropriate format. In that case, the events of ordering the product will be previously scheduled in the simulation for the proper times, and these ordering policies will, naturally, be disabled.

Apart from this, there are some other important questions associated with the ordering policy. The first is "how to order?", which in this approach results in the option *immediate*, if the order must be placed to the supplier in the precise moment that the need is detected; and in the option *management*, if the order must first wait for that a single order can be placed to the supplier in the end of the day, for example.

The second question is "from whom to order?", and it implies that the user links each *product-box* to the respective CSU supplier (or various suppliers).

Finally, the simulator also uses the parameters *one-order-cost* (or setup cost), to represent the cost of placing an order to the supplier; and the "*time for full satisfaction*[7]", related with equation 2.7, which specifies the maximum expected supplier lead time for the product in question.

3.7.1.3 Technology

Technology is another category of data associated with the stock operation in this simulation approach. It is basically related with the technical specifications that are indirect expressions of the actual stocking technology in use in the CSU. Through such data the user specifies both for the *process of inputting* materials into the CSU and the inverse *process of outputting*, the number of bays for loading/unloading the vehicles, the serving speed associated to the stock operational performance, and some other indexes, most of them given in terms of time delays. In the *process of inputting* it is generally considered the utilization of the following basic sectors: input bays for the vehicles to queue and unload; marshalling zone where the orders are disassembled and the respective products prepared to be transferred to the stock; and a stocking system, that is, a sector structured and prepared to receive the products and keep them perfectly ordered and monitored, as represented in figure 3.16.

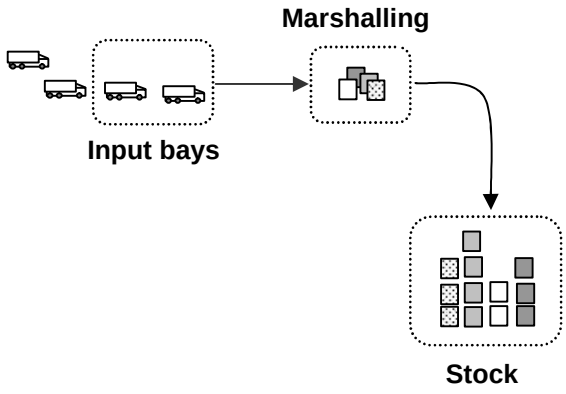

Fig. 3.16 Basic sectors associated with the process of inputting material into the stock

Notice that there will be a zone of marshalling both for the process of inputting and for the process of outputting. The parameters particularly related with the *input process* are the following:

***Number of input bays**: the number of unloading bays available to receive vehicles.*

***Preparing bay time** (min/bay): the average time and standard deviation needed to prepare a free bay to receive the next vehicle, measured in minutes.*

[7] This parameter is presented in the simulator application by the term "*cool-time*", as it is a more compact expression.

Unload time *(min/unit): the average time and standard deviation for unloading the vehicle a stock keeping unit (SKU) of material, measured in minutes.*

Marshalling time *(min/unit): the average time and standard deviation in minutes needed to prepare a SKU to be moved into the stock.*

Max stock input rate *(unit/hour): the maximum rate and standard deviation at which the stock can be operated, reflecting its internal technology.*

Stock input function *(min/unit): the average time and standard deviation in minutes needed to effectively input a SKU to the stocking structure.*

A similar list of parameters is used to characterize the *process of outputting*. All theses parameters are suggested to the user with default values, in a strategy to help in reducing the time spent on configuring the model.

3.7.2 The fleet

The fleet is viewed as a list of vehicles dedicated to the delivery and linked to the CSU. The physical space occupied by the fleet in the CSU is only indirectly known through the *fixed costs* associated to this resource. In principle, if these costs are null, it implies that the tasks associated with the delivery are hired and will be only expressed in terms of *variable costs*.

Each type of vehicle belonging to the fleet is configured in accordance to the three following categories: (1) costs, where information about the driver costs and distance dependent costs are recorded; (2) speed, where the speed of the type of vehicle is characterized; and (3) handling, where data concerning the transport capacity of the vehicle and the typical times for handling materials while loading and unloading are specified.

As previously stated, the vehicle types that can be chosen to compose the fleet include *road vehicles*, *trains*, *boats* and *airplanes*, as we will soon see in the chapter dedicated to the simulator application.

3.7.3 The production section

As in the previous case, the space occupied in the CSU by the *production section* only indirectly is known by the fixed costs of occupancy associated with it. The *production section*, where *manufacturing processes* take place, is modelled as a resource linked to the stock facility, and interfacing with it precisely in the same way the external clients do, that is, by means of orders to output or input material (Fig. 3.17). These orders must wait for the usual processing resources to become available.

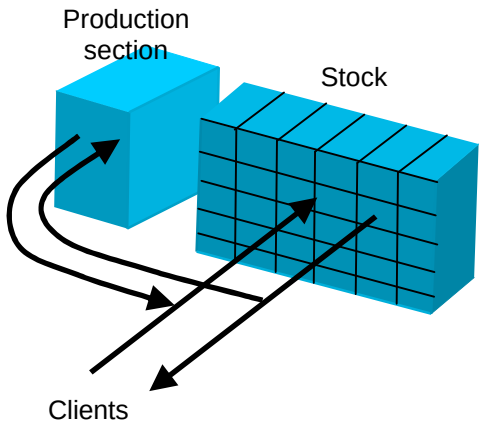

Fig. 3.17 Production section as a regular client of the stock facility

In the same sense that the *products* are handled in the inventory by means of *product-boxes*, each *manufacturing process* (refer again to figure 3.8, if necessary) assigned to the *production section* is here represented by a *product-plan* in which the information needed to produce the

respective product P+ is specified. Notice that only products of P+ type can be linked to a *production section*. A *product-plan* can then be seen as the specific procedure to execute the manufacture of such type of items. The *product-plan* is linked to the *product-box* of the P+ product, which in turn is linked to the P+ *product*, and this to the respective *sub-products* in the form of a MRP description tree (or bill of materials), as shown in figure 3.18.

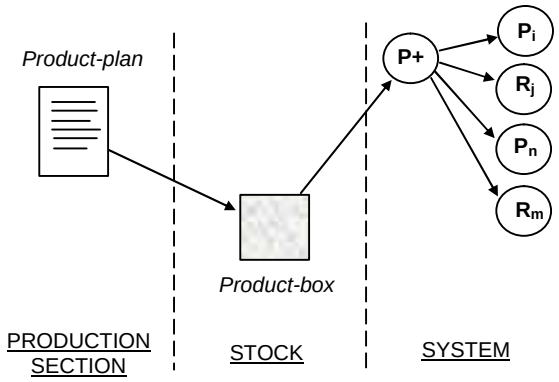

Fig. 3.18 The association between a product-plan, a product-box, a system product and its sub-products

Obviously, a *product-plan* can only be linked to a *product-box* that has previously been associated to a P+ product.

To schedule and to control its internal activities, the *production section* needs to be configured in terms of: (1) general information, which in this case is reduced to simple association of a P+ *product-boxes* to a *product-plan*; (2) manufacturing policy, where information about when to produce, how to produce, etc., is to be recorded; and (3) fixed costs, which in this case include *labour* and *occupancy*, as mentioned in section 3.4.4.

3.7.3.1 Manufacturing policy

Unlike the *ordering policy* of the stock, where data has been separated from the data of "technology", the *manufacturing policy* was made to also incorporate the data related to the "technology" operating at the *production section* and the *variable costs* related to the productive activity. This is clearly due to the fact that while the *stocking process* in principle has the same characteristics to all the products (thus it makes sense to distinguish "technology" from "policy"), the *manufacturing process* is commonly very dependent on the specificities of the P+ product being manufactured. That is, each *product-plan* represents a single and independent *manufacturing process*, therefore it makes sense that data like the related with the moment to start producing, how to produce, etc., is alongside data representing the production rate and the costs of producing, for example. This is our perspective of modelling.

It is now important to present a clear idea about the kind of model we will use to represent the production line, since the required data is obviously dependent on it. Here we consider that the production line can only produce multiples of a certain predefined quantity of product units (which we call the *lot size*) at a time, and also that a *production rate* (*pRate*) characterizes the maximum output rate of the system. The *raw process time* (*rpTime*) is considered the time needed to produce a lot when there is no need for queuing. This approach is equivalent to considering the *manufacturing process* a sort of conveyor with length equal to the *raw process time* and where the lots are allowed to "enter" with a maximum

frequency equal to the *production rate* (Fig. 3.19).

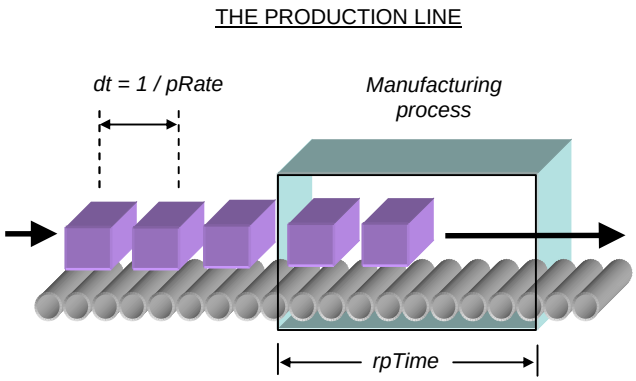

Fig. 3.19 The model used to represent a production line

As it can be deduced from this figure, when producing a series of lots one needs to wait the *raw process time* only for the first lot; the others will follow it, appearing in the output with the frequency *pRate*. The costs of producing those lots will be easily computed once the *cost of producing one lot* is known. This is another parameter of configuration taken into account.

Finally, the information about *when to start* to produce must be appended to this data. Concerning this last aspect the following possibilities (or policies) had been considered:

ByLevel: if the P+ product must start to be produced each time its quantity in the inventory gets smaller than a certain predefined value (equivalent to ROP). When this happens, all the sub-products needed for producing the quantity of one lot are retrieved from the inventory and the new P+ product is produced. At the end of the production the quantity of P+ produced is sent to the inventory.

ByCycle: when the P+ product is to be produced regularly. The model also allows starting the process after a certain initial delay, therefore introducing different shifts in the production of different products. The quantity to be produced is defined by the pre-fixed *lot size* and the frequency is the value set in the *production rate* (*pRate*). This makes the system work in a kind of continuous production mode.

OnDemand: the P+ product is to be produced the moment it is removed from the inventory. In fact, once in this option, the required quantity of P+ is obtained directly from the inventory while the same quantity starts being produced to replace the inventory as close as possible to the previous level. The quantity (q) removed from the stock is dependent on the demand, and so, the number of lots to produce is:

$$nLots = q / lotSize \quad (3.16)$$

Frequently this does not represent an integer number of lots, and a strategy for rounding the value is usually adopted, by excess or by defect.

3.7.4 The management section

This resource is considered a pure abstraction in this approach. The costs associated with it can be appended to the fixed costs of keeping the stock facility in the form of a new item, for example. This was our option. The moment the user fills in the fixed costs associated to the stocking, the user also fills in those associated to the management of the entire CSU facility.

The management structure is, in any case, distributed, the *CSU-manager* being the first responsible for the management in

the entire CSU, composed of sub-blocks dedicated to more specific areas, such as the *Stock-manager*, the *Fleet-manager* and the *Production-manager*. Each of these "expert managers" is responsible for executing the policies of its respective section, for example, as well as to resolve the particular situations occurring in any tasks related to its domain, as figure 3.20 suggests.

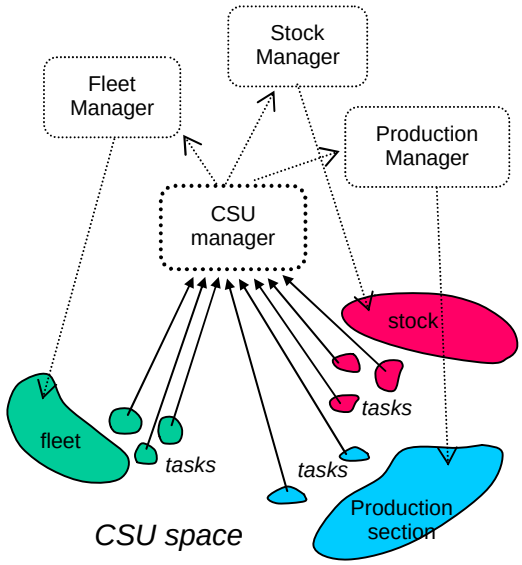

Fig. 3.20 Supervised-distributed management in a CSU

This approach can be considered of the "top-down" kind, with the *CSU-manager* working as a distributor of management jobs. The advantage is that in any problem related to the CSU communications are always established with only one and the same "entity", that is, the "supervisor" *CSU-manager*, which then will analyse the situation and distribute a job to the most appropriate "manager". As we will see later, when a CSU event needs to make certain decisions, the control is sent to the *CSU-manager*, and the process waits till it receives some answer.

3.8 Modelling the CSU activity

Till now we have discussed the static part of the Supply Chain modelling, that is, those issues and elements that are important to the definition of the aspect, the structure, or, in one line, the general topology of the system. It is obvious that this structure strongly interferes with the dynamics that will develop in the simulation when the model will be made to run, since in reality it represents the physical constraints of the problem. Even so, what in fact drives the advance of the simulation in time is the sequence of events that represents each of the processes running in the CSU and the operational rules they contain. The present section is dedicated to this topic.

As we know from the last sections, in each CSU there are some processes running and interfacing with one another, that way creating the complex dynamics of the unit. These processes are, as we know, the *purchase*, the *stocking*, the *manufacturing*, the *delivery* and the *management*. Except for the *management*, which we will consider an instantaneous process coincident with any possible events of the CSU, each of the other processes is set to contribute to the overall CSU's state with a certain number of states (we will call live states those states of which the duration can previously be computed or known; and dead states those states where this cannot be done, as in the case of waiting states or queues, for instance (Brito & Feliz-Teixeira, 2001)). The collection of such states, and the possible connections among them, form the complete state diagram of the CSU, which in fact is the core of the model. So, to simulate the Supply Chain we first have to design and specify the state diagram of the

CSU, and then to convert it into a sequence of events which can finally be translated into a computer programming code. In our case this code will be the C++, a few of which will be presented in the next chapter.

While creating the Supply Chain model in the simulator application, the user will not need to think on these internal CSU states, as in this approach they become endogenous to the CSU object. The Supply Chain representation will therefore appear as a game played with visual CSU objects that must be appropriate interconnected by paths of transportation and of information. As we have noticed earlier, from the point of view of the user, each modelling task will be a kind of "real objects" exercise. ***

3.8.1 Process of purchase (or ordering)

This process is related to the ordering of materials to the supplier, and is due to begin whenever one of the following conditions is observed: (1) the present stock control policy determines it is time to reorder; (2) there is an order from a client that cannot be fulfilled directly from stock and, in the case of accepting *backorders*, new material must be ordered.

After creating and preparing the new order with the list of materials required to the supplier, any of these "conditions" will trigger the *purchase process*, which can simply be described by the two major events: (1) transmission of the new order to the respective supplier, which we will name *e_order()*; and (2) receiving the material that arrived from the supplier, which we will name *e_arrive()*. Between these events the *purchase process* will fall into a dead state, where it waits for the materials to arrive from the supplier, as represented in figure 3.21.

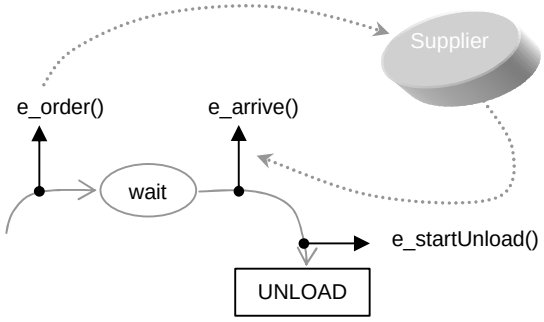

Fig. 3.21 State diagram of the purchase process

Notice that the supplier *lead times* are initially unknown, since they result from the internal dynamics of the CSU supplier and depend on the transport speed and availability, for example. Also notice that the event *e_arrive()* can be thought the beginning of a subsequent live state which in fact represents the UNLOAD activity. This will let us later substitute the event *e_arrive()* for the event *e_startUnload()*.

As we have seen, the event of ordering *e_order()* is strongly linked to the *ordering policy* previously assigned to the product in cause, and can be generated anywhere in the simulation. Anyhow, when the policy is one of the "ByCycle" type, this event is basically generated by a circular *ordering process* that will maintain the activity by itself along time. This process will be running independently for each product configured that way. This is the method used to generate de demand at the ultimate customer level, that is, the leading demand to the Supply Chain.

3.8.2 Process of stocking

For reasons of simplicity, we will regard the *stocking process* as the meeting of two other distinct processes: the *input process* and the *output process*. As the names

suggest, the *input process* is responsible for inputting into the stock the materials arriving to the CSU, while the *output process* will ensure the outputting of materials from the stock of the CSU. Figure 3.22 gives an idea of the *events* and *activities* involved in the *output process*:

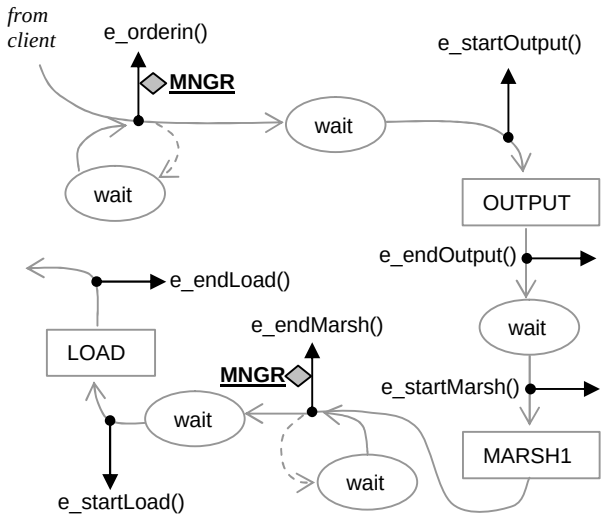

Fig. 3.22 State diagram (or activity diagram) related with the stocking output process

Basically, this process starts in response to an *e_order()* event sent by a CSU client, that is, an order to output material. Such event is received in this CSU in the form of an *e_orderin()* event, so that the two events can be distinguished. This event will then ask the permission of the *CSU-manager* to satisfy such an order for outputting materials. The *CSU-manager*, by means of its *Stock-manager*, will decide if the order is to be immediately served, rejected as a *stockout*, or made to wait to be satisfied later when new material will arrive from the supplier. As long as the order is ready to be served to the client, the process enters the phase of outputting the materials from the inventory. Then the material will be prepared and packed at the MARSH1 activity, and the order gets ready to be sent to the client. Anyhow, the *CSU-manager* (now directed to the *Fleet-manager*) once again called by the *e_endMarsh()* event will decide whether or not the order will wait for later LOAD and transportation, keeping it in a different waiting place depending on that decision. Once the transport to the client is available and an output bay is free for loading, the LOAD activity can start. At the end of it, the event *e_endLoad()* will trigger the delivering process.

Concerning the *input process*, that is, the inverse of this process, we can find some similarities with what was described before. As figure 3.23 shows, the diagram of this process can, in fact, be seen as a part of the diagram of the previous process with some new activities replacing the old ones.

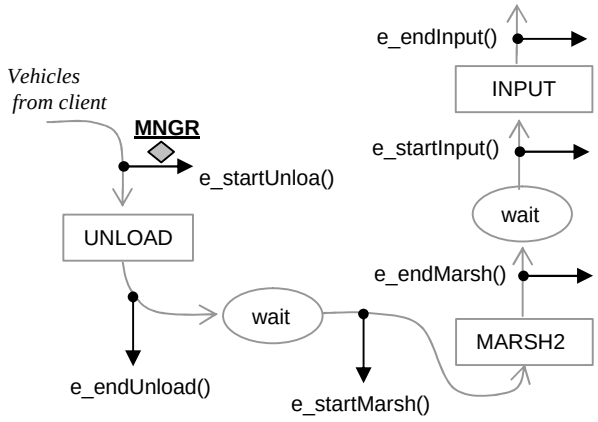

Fig. 3.23 State diagram of the input process

When a vehicle arrives from the supplier, the *e_startUnload()* event is responsible for calling the *CSU-manager* for stock management. Then, the activity of UNLOAD replaces the activity of LOAD, the marshalling works in the opposite direction and the activity of INPUT substitutes that of OUTPUT.

Despite these similarities, in this first version of this modelling approach the *input process* has suffered some simplifications which practically reduced it to a modified version of an UNLOAD activity. This was equivalent to consider that the material will be moved into the stock precisely in the activity of UNLOAD, greatly simplifying the modelling in terms of the need for synchronization between the *input* and the *output* activities while accessing the stock of the CSU. Although, since not-yet-fulfilled orders in the *output process* will have to queue while waiting for the OUTPUT activity, this simplification is expected not to lead to significant discrepancies in the results, as it is as if the INPUT activity would, together with the marshalling, run in parallel with the OUTPUT activity and, on average, the material would already be in the inventory when it must be taken out from there. On the other hand, to output from inventory is usually significantly slower than to input, and so it is expected that this simplification does not induce grave distortions in the model. Of course, in this provisional version certain times and mainly costs are still computed as if the activities of MARSH2 and INPUT would be present, but the issue is to be resolved in a next version.

3.8.3 Process of delivery

In this process, transportation vehicles will be carrying materials to the clients. As it was earlier mentioned, the transport of those materials can be done in many different ways, from using normal roads to airlines, and therefore the efficiency of this process will be strongly dependent on the kind of transportation chosen. Although the delivery process is conceptually very simple, as shown in figure 3.24, it can also become significantly complex, principally when nodes of transfer have to be used for modelling connections between different types of vehicles, or even when the delivery network is in itself complex. In any case, the vehicle entities will themselves be responsible to handle the materials while they move along the network, making them practically "transparent" to the CSU *delivery process*, as the same figure suggests.

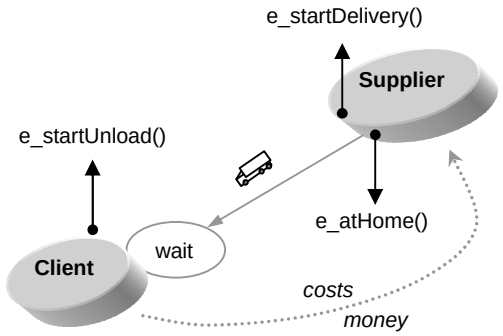

Fig. 3.24 State diagram of the delivery process

So, we consider that this process begins with the event *e_startDelivery()*, which is equivalent to *e_endLoad()*, and will end the moment the vehicle is finally received by the CSU client, that is, at an event *e_startUnload()*. In the case of more than one vehicle delivering to the same client, they probably need to wait for a free *inputBay*, as proposed in the same figure. With this model, only after the end of simulation can the lead times of the system be estimated. So, lead times are treated as outputs, and not as simulation inputs.

Notice that the true final part of the *delivery process* is the return of the vehicle to its CSU home, which will terminate with the event *e_atHome()*. For reasons of simplicity, however, the time for returning home will be estimated as 75% of the time

spent to reach the client, so, this event will be in fact automatically scheduled when the vehicle reaches the CSU client, at the "end" of the *delivery process*. At that same moment also the costs and other parameters related with the vehicle are computed and the respective payment is made to the right suppliers.

3.8.4 Process of manufacturing

The *manufacturing process* is modelled here assuming no product mixing and only a line of production for each product P+ manufactured. This, of course, can be made more complex in future, but for the moment it will be treated in this simplified form, as our major concern is to focus attention on the overall Supply Chain behaviour and not particularly on production[8]. In a wide range of real cases, however, this model is expected to be useful without significantly affecting the consistency of the results.

The states diagram of such a model is very simple and is represented in the next figure (Fig. 3.25). There is a waiting state (queue) for the orders waiting to enter the production, an event *e_startProduction()* responsible for the beginning of the activity, and an event *e_endProduction()* in charge of the actions that must be executed when the PRODUCTION activity is finished. As indicated in the section 3.7.3.1, this model uses the traditional concepts of production rate, which establishes the maximum frequency at which the orders can enter the PRODUCTION activity, and the raw cycle time, defining the time that each *lot* must spend in that same activity.

[8] "Manufacturing" and "production" are terms used with equivalence in this context. We will use "production" more during the implementation phase, since it is a shorter word.

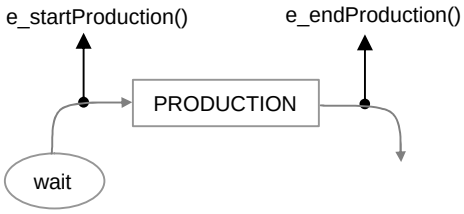

Fig. 3.25 The simplified manufacturing process

As previously said, the start of the *manufacturing process* will be decided either by the stock level of the product P+ to be produced, or by using a production cycle, as in the continuous production, or directly triggered by the orders of the clients if operating under a on demand policy. This will be established by the user. At the proper time, the necessary products and the respective amounts are removed from the stock (using the *output process*) and the new product P+ starts to be produced. At the end, the new product will be added to the stock of the CSU and the associated costs computed.

3.9 Products, orders and vehicles

It is important at this moment to make also some considerations concerning other elements of this modelling approach which do not specifically belong to the CSU structure, despite remaining basic for the simulation. These are related to the principal flows in the Supply Chain and therefore they may be seen as playing the roles of messengers between CSUs. They are the *system products*, the *material-orders* and the *vehicles* for transportation. *System products* are in fact passed from CSU to CSU by *vehicles* in response to *material-orders*. This section is reserved to describe each of them in more detail.

3.9.1 About the system products

The idea of *system products* was used to resolve certain conflicts related to the attributes of products in different CSUs. For example, as was previously noticed, if the products would be considered property of the CSU, the stock of each facility would only use its very own items, transforming such representation redundant and little flexible, since in reality diverse facilities use the same product in their inventories. Another aspect was the need for relating different ordering policies to the same product if it would belong to different CSUs, for example. So, we obviously decided that the *ordering policy* would not be linked directly to the *product* but to something in the stock that would contain the product, that is, the *product-box*. *System product* is therefore a term aimed to avoid confusion whenever necessary, but in fact represents the *product*. This *product* is the same item to all the Supply Chain *system*.

Some parameters have been ascribed to the *system product* with the intent to reference it in the Supply Chain model and to define its attributes. These are:

***Reference**: a unique identification of the product in the system. It is automatically attributed by the system when the product is created and can be used in future searches.*

***Type**: the type of the product. R for raw materials, P for parts and P+ for secondary products. Notice that only products P+ will be allowed to be linked to a manufacturing process (through a product-plan).*

***Name**: a name to easily identify the product.*

***Units per container**: the number of product units (SKUs) that can be transported in a normal container. All the measures in space occupancy are later made based on this value.*

***Base price**: the cost of a unit of product (SKU) when it enters the Supply Chain for the first time. Most of the products enter through the primer suppliers, but it does not have to be the case. The price of the product in the next CSUs will be dependent on the chain of value.*

***Typical load/unload time**: an estimation of the time needed to load or unload a product unit (SKU). This is basically an index that later can be used in a special situation, if necessary.*

Other parameters like the *life-cycle*, intended to represent the period of time that the product is considered valuable, which strongly interferes with its price, as well as the *next-generation*, by which the new versions of the product would be generated in the simulation, have not been implemented in the present modelling version, although they have been planed since the beginning.

3.9.2 The material-order structure

The *material-order* (or simply *order*) is the element that triggers the flow of the materials in the Supply Chain. In this version, the *order* is primarily composed by a list of *materials*. A *material* is seen as a registry containing attributes related to the product to order, the required quantity, the reference, the product name, etc., so that the supplier can clearly link to it the correct item of its inventory (Fig. 3.26).

material	ref	name	quantity	units/container
MAT1	p(1)	butter	100	1000
MAT2	p(23)	milk	300	500
MAT3	p(12)	water	50	400

Fig. 3.26 Example of a list of materials

In the previous figure, a *list of materials* (a part of the *order*), is represented in the form of a table.

Since in the simulation the order is always made to return to the CSU of origin (when not refused by the supplier), some other attributes were also added to the *material*, like a link to the *product-box* of origin, the cost of the product meanwhile served by the supplier, and the average time that it was waiting before being effectively delivered to the client. These attributes are mainly used to help in simplifying further calculations and to ensure the coherence of the flow of materials between the facilities.

Then, the complete *order*, as considered in this approach, is an element containing the information of a list of *materials* and, in addition to that, some other attributes to help identify the *supplier* and the *client*, the moment when the order was placed, the vehicle currently transporting it, the total number of SKUs, etc. We like to abstractly represent the order as a kind of another "box" which is at first sent empty to the supplier and then will return with different types of materials to its CSU of origin. This is what we intend to represent in figure 3.27.

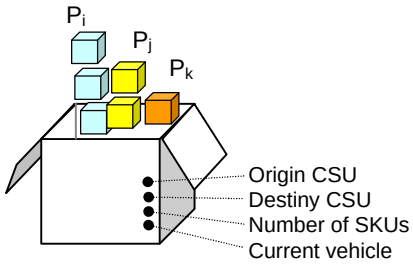

Fig. 3.27 Basic model of a material-order

Notice that this is a basic and illustrative model. The one effectively implemented in the simulator includes a *list of materials*, the *delivery path* to the destiny, 14 other attributes, and also some methods for executing certain operations.

3.9.3 Modelling the vehicles

All vehicles are modelled here in the same basic form, that is, they are based on the same abstract mould (or class) which we have named *Veiculo* (vehicle in Portuguese). This element has, as usual, diverse attributes, but also a *transport path* which is associated with it before the start of each new delivery job. Basically, this is how the vehicle gets to know the way to its clients, and a form to be independent of the CSU, from then on. Although the *Veiculo* class implemented in the simulator has almost 40 attributes (it would make no sense to present them in this context), here we will only talk of those which are essential to understanding how the vehicle dynamics have been modelled. With that purpose, figure 3.28 shows a visual representation of the concept adopted.

Basically, before starting the delivery, the vehicle inspects the first order it carries and assumes as its job the transportation of such an order to the respective client, by following the *delivery path* previously assigned to the order by the *Fleet-manager*. Notice that each order includes its own *delivery path*. When the current order is delivered, the vehicle inspects the next order and will repeat the previous procedures till there are no more orders to delivery. At that moment, the vehicle returns home.

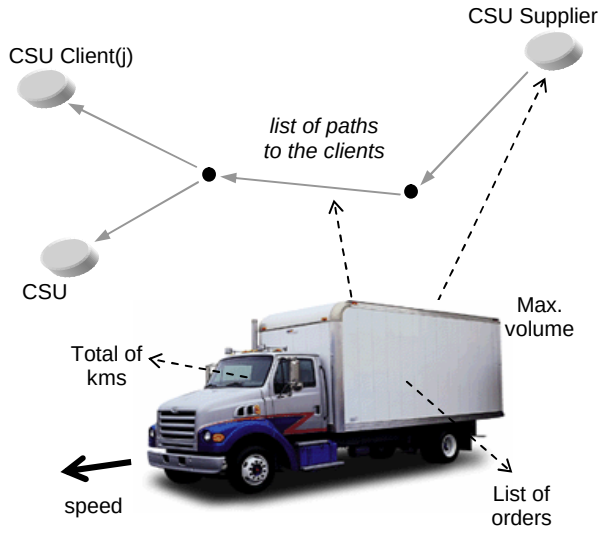

Fig. 3.28 The vehicle and some of its attributes

The vehicle movement along the delivery paths is basically driven by two events: *e_start()*, which corresponds to the moment the vehicle enters a new *path*; and *e_end()*, which is executed at the end of each path, when a new *node* is reached. In this case, some actions must take place, depending on the type of the node:

Normal node: the vehicle compares the next path of the current order with each of the paths going from the node. The vehicle will follow the path that will lead to the current client.

Transfer node: the vehicle transfers to the new vehicle all the materials it is carrying and all the information needed for the later calculation of costs, delivery times, etc. Then, it returns home.

Pause node: the vehicle simply stops and schedules the next event of *e_start()* for the proper time, obviously including the delay of the pause. The costs of the pause are also added to the costs of transportation.

CSU node: when reaching this type of node the vehicle will act as at a *normal node* if this node is not the CSU client to where the current order is to be delivered. On the contrary, if the node corresponds to the CSU destiny, the vehicle enters that CSU and transfers the responsibility of the next actions to the CSU. In the proper time the CSU will set it free, either to execute the next delivery job or to return home.

3.10 About the flexibility (matrix)

In the previous chapter some references were made to the concept of flexibility and the way it is currently seen by Supply Chain managers. The present modelling approach is also claimed to introduce the possibility of measuring the flexibility associated to a certain Supply Chain structure by means of simulation. This new feature is based on finding a matrix related to the amounts of inventory in imbalance (rigidity) when steps of demand are injected in the Supply Chain. The concept of rigidity is seen as the inverse of flexibility, and it will be used in order to simplify the representation of the matrix. In this section we will present in more detail such a proposal for measuring the Supply Chain flexibility, and explain the way it was integrated in the simulator. More detailed information about this theory and the method of building the rigidity matrixes, as well as some practical results, can be found in Feliz-Teixeira & Brito (2004).

3.10.1 The rigidity concept

As was referred in the previous chapter, in the section devoted to *facility related metrics*, the concept of *rigidity* is a way of representing the energy spent on imbalance while the company is reacting to a sudden

step in the demand (Fig. 3.29). In the case where the attention is focused in the inventory performance, the rigidity is measured in terms of SKU/day in imbalance. In figure 3.29, this corresponds to the excess and deficit in the quantity of product in inventory along the dashed area of the *Equilibrium* function, divided by the time of imbalance *Tq*.

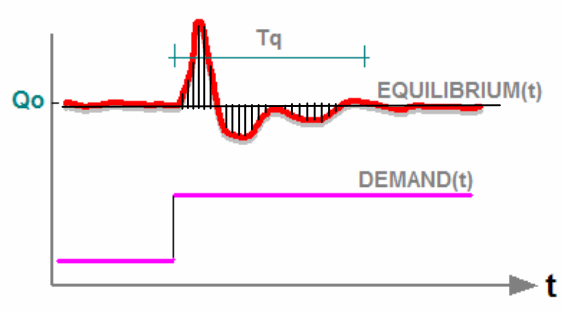

Fig. 3.29 Concept of rigidity as an imbalance ratio

This means that the more unstable the response to a sudden change in the demand, the more rigid the system. That is, the less it is able to absorb shocks.

This is a general view of rigidity, which can be applied not only to the stock reaction but also to any other class of responses, like to the costs, for example (in that case the rigidity could be measured in €/day). In any case, it is essential to write the equations governing this process in order to obtain quantitative results from such a metric. This is what we will do right now, for the example of the inventory.

We first suppose that the inventory will be in equilibrium when the ratio of products going in, equals the ratio of products going out. That means that the stock manager is in fact able to ensure the rotation of the materials, or the flow of products, without incurring excessive costs of holding or costs due to *stockouts*.

So, considering Q_{out} the number of SKUs leaving the inventory during the time interval dT_{out}, and Q_{in} the number of SKUs entering the inventory in the time interval dT_{in}, such a condition of equilibrium can be simply expressed as:

$$Q_{out} / dT_{out} = Q_{in} / dT_{in} \tag{3.17}$$

Supposing that we make the observations during the same time interval, that is, if we consider $dT_{out} = dT_{in}$, this leads to:

$$(Q_{out} - Q_{in}) / dT_{in} = 0 \tag{3.18}$$

which is another way to represent the *Equilibrium* function for the inventory, and can be thought an "*imbalance ratio*", measured in inventory units (SKU) per unit of time. This already corresponds to a certain "energy" spent per unit of time, and thus gives us an idea of a "power spent on imbalance", which in fact can be seen as the inverse of flexibility. By this definition, the system is more flexible if the rigidity gets lower, and a simple way to measure it is to integrate the *Equilibrium* function over the time and then divide this value by the total time of imbalance. So, if for a certain period j of time $dT_{in}[j]$ it holds that there was $Q_{in}[j]$ units of material going into the node and $Q_{out}[j]$ units of material going out, the rigidity at the end of the observations can be computed as:

$$RIGIDITY = \frac{\sum_j |Qin[j] - Qout[j]|}{\sum_j dTin[j]} \tag{3.19}$$

And, from it, retrieve the flexibility:

$$FLEXIBILITY = \frac{1}{RIGIDITY} \qquad (3.20)$$

It is based on these expressions that the matrix of the rigidity is computed in the Supply Chain simulator.

3.10.2 Matrix representation

The calculation of the rigidity is easily achieved by counting the number of product units leaving and entering each facility between successive arrivals of materials from the suppliers. This is in accordance with the equation (3.19) and it is easily implemented in practice.

We must notice, however, that till now we considered the rigidity of a single CSU (by its inventory), which can be represented by a scalar. On the other hand, the rigidity of a Supply Chain emerges as a property of a collection of CSUs, therefore resulting in a more complex mathematical element, as is the matrix. That is, the rigidity of a Supply Chain will have the form of a matrix R_{ij} where the rigidities of all the facilities are exposed under certain criteria. This matrix will characterize the Supply Chain only for the specific conditions in which it was created, being obviously dependent on the ordering policies adopted at each facility, as well as on the delivery policies, the amplitude of the demand shocks, etc. It is, however, a good tool to form an idea about how the overall system will be able to answer to such a demand change. It is, in fact, the multi-node *step response* of the system, which contains extremely useful information about the system itself.

In order to clarify this subject we use again the example described in the final comments section of the last chapter, that is, the didactic Supply Chain of the "*Cranfield Blocks Game*" (Saw, 2002), which meanwhile was analysed with the present modelling concept (Feliz-Teixeira & Brito, 2004):

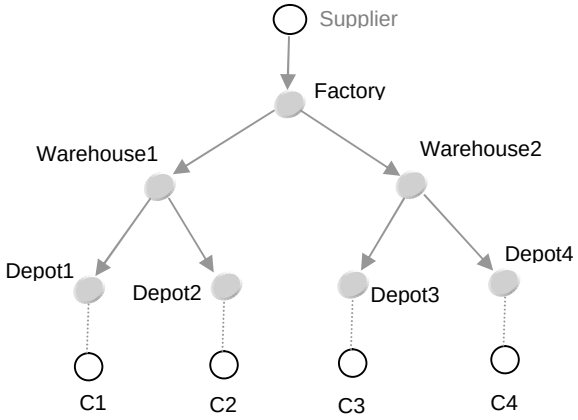

Fig. 3.30 The *Cranfield Blocks Game* Supply Chain

This is a seven node Supply Chain (from *Depot1* to the *Factory*) where the direct demand (*inputs*) from *customers* (*Cj*) is injected at the *depots* level, and the *Supplier* is considered an infinite source of materials. It is a three level Supply Chain where, for simplicity, only one product is considered.

This system has four (4) *inputs* driving certain *output* functions at each of its seven (7) nodes. Thus, by means of injecting at each input a step of demand one will be in principle able to observe how any of those output functions (as inventory, costs, etc.) respond to such conditions, obtain some measures and then represent the rigidity matrix of the overall system. A simple way to write down this matrix is to plot the individual rigidities in columns ordered by increasing levels of the facilities, as suggested in the next figure. This let us fast acquire an idea about how the chain is susceptible to such input "shocks".

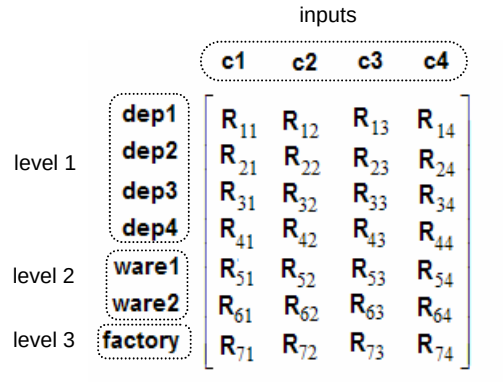

Fig. 3.31 General rigidity matrix for the Supply Chain of the *Cranfield Blocks Game*

From this representation one can also realize that the rigidity of the chain will be the rigidity of each of its "arms" connecting the *last customers* to the *primer suppliers*. It is therefore unsurprising that a particular Supply Chain may show a high flexibility in serving certain customers while exhibiting a disastrous flexibility in serving others. A nice result from these considerations is that "*the flexibility of an arm will be extremely dependent on the flexibility of its slowest node*", since the speed in a chain is the speed of the slowest element. So, it seems that *cooperation* is a need that naturally emerges from the idea of flexibility applied to the Supply Chain.

In practice, the procedure leading to the rigidity matrix, by using simulation, is as follows: (1) simulate the system for a certain time with constant demand in all its inputs; (2) at a certain moment, provoke an instantaneous increment of demand in the first depot, keeping the others working as previously; (3) at the end of the simulation, measure the rigidities in all the facilities of the Supply Chain, using the formula (3.19), and record them in the form of a column; (4) repeat the simulation and the previous process, applying the step of demand to the next depot, till all the depots have been "shocked".

In the chapter dedicated to the results more about this example will be presented and discussed.

References:

Brito, A. E. S. C., & Feliz-Teixeira, J. M. (2001). *Simulação por Computador*. Porto, Portugal: PUBLINDÚSTRIA.

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2004). *On Measuring the Supply Chain Flexibility.* Paper presented at the 2004 European Simulation and Modelling Conference, UNESCO, Paris, France.

Gonçalves, J. F. (1999). *Gestão de Aprovisionamentos* (revista ed.). Porto, Portugal: PUBLINDÚSTRIA.

Mason-Jones, R., & Towill, D. R. (1999). Using the Information Decoupling Point to Improve Supply Chain Performance. *Journal of Logistics Management, 10* (2), 13-24.

Saw, R. (2002). *Cranfield Blocks Game*: Centre for Logistics and Supply Chain Management (CSCM), University of Cranfield, UK.

Sussams, J. E. (1992). *Logistics Modelling*. London, UK: Pitman Publishing.

CHAPTER 4 137
IMPLEMENTATION AND OVERVIEW 137
4.1 INTRODUCTION 137
4.2 THE SIMULATOR STRUCTURE 137
4.2.1 The classic proposal 138
4.2.2 The OOP tendency 139
4.3 DISCRETE EVENT MODELLING 139
4.3.1 System states and time advance 140
4.3.2 Events versus activity cycles 140
4.3.3 The SimEvent object 141
4.4 THE BASIC ENTITY 141
4.4.1 The SimEntity object 142
4.5 LISTING THE EVENT IDS 143
4.6 CODING THE STATES (ACTIVITIES) 143
4.7 THE SIMULATOR OBJECT 144
4.7.1 The schedule method 145
4.7.2 Random Normal generator 146
4.7.3 The executive loop 146
4.7.4 The SimDlg loop controller 147
4.8 ENTITY MOTION (SIMMOVIE) 149
4.9 A DAILY MARK (SIMTIMEMARK) 150
4.10 SOME UTILITIES 151
4.10.1 The SPECTRUM class 151
4.10.2 The graph window (GraphDlg) 152
4.10.3 Visual modelling (Area) 153
4.11 SUPPLY CHAIN ENTITIES 154
4.11.1 Paths 155
4.11.2 Nodes 156
4.11.3 Vehicles 159
4.11.4 Material-order (Encomenda) 161
4.11.5 The product 162
4.11.6 Product-box 163
4.11.7 Product-plan 165
4.11.8 The CSU entity 166
4.11.8.1 Configuring the facility 167
4.11.8.2 Icon 167
4.11.8.3 States (activities) 167
4.11.8.4 Event handlers 168
4.11.8.5 Other methods 169
4.11.8.6 Main resources 169
4.11.8.7 Suppliers 169
4.11.8.8 Fleet 170
4.11.8.9 Graphs 170
4.11.8.10 Final metrics 170
4.12 THE SIMULATOR APPLICATION 171
REFERENCES: 175

Chapter 4

Implementation and Overview

Coding the simulator in C++ and an overview on the final application

4.1 Introduction

This chapter will be organized in two major parts. At first, we will present and discuss the basis for implementing the simulator in a general purpose programming language, which in this case will be focused in the C++. Then, an overview on the final computer application for Supply Chain modelling and simulation meanwhile developed will be offered, giving the reader an idea about its particularities and potentials.

Since documenting in detail each of these subjects would imply the writing of a large amount of information, we decided to reduce such exposition to a level considered essential to the good understanding of the issues without the loss of any important aspect. Principally concerning the C++ development, it is important to remind that the source code is distributed over 55 different object classes, several of which having quite more than 10 attributes and various methods in their structure. In particular the CSU class makes use of almost 180 attributes of several kinds and 40 methods, of which 20 are specifically dedicated to simulation events. A detailed description of such information would be unreasonable in the present context. We will try, therefore, to focus the attention on the most relevant issues, integrating them in such a way that the overall exposition can come about positive and useful.

We would also like to note that while some C++ code will be presented in the first part of the chapter, constant attention will be given to the representation of the same ideas in the form of diagrams, whenever possible, in order to also allow non-C++ users to benefit from such information. Besides, whenever we consider it to be useful, we will also include some brief explanations about any relevant aspects related to *discrete simulation*.

Concerning the second part of the chapter, we decided to organise it as a sort of guided “tour” around the computer application which meanwhile was built with this modelling approach.

4.2 The simulator structure

The present simulator was designed and implemented founded on the paradigm of *sequential objects*, that is, using an object oriented ambience where each object has access to the others through a mechanism of shared memory, within a single application. Thus, the access to most of the *variables* and *methods* of the application can be made by means of *pointers*, or references. Although this is just the classical *Object*

Oriented Programming (OOP) scheme, it is still the method with which the fastest executions of *time-dependent* models are achieved, as the basic information between objects is naturally passed in "parallel". That is, there is no need for any *protocol of communication* to transfer information between objects. Instead, the objects live in a "sea of memory", even if direct access to certain parts of that memory can be restricted. Such is the groundwork supporting our *discrete event* modelling approach.

4.2.1 The classic proposal

The structure usually recommended for the implementation of a discrete simulator largely reflects the idea of dividing the programme code into the three main blocks shown in the figure 4.1, each having different responsibilities: (1) the **executive**, where the source code of the "engine" of the simulator resides; (2) the **operations block**, where all the code related with the implementation of the *activity cycle diagrams* of the model is, therefore where the system operations are described; and (3) the **utilities block**, containing all the types of resources, functions and methods used for supporting the simulation process, like random number generators, histograms, graphical tools, *dialog-boxes*, etc.

While the *executive* mainly controls the time-advance in the simulation, that is, ensures the rolling of the simulation loop, and the *block of utilities* can simply be seen as a repository of useful tools, it is in the *operations block* that the core of the model is described. Simulation languages, for example, frequently hide from the user the code of the *executive* and of the *utilities block*, as they are considered a constant for the simulation of any model.

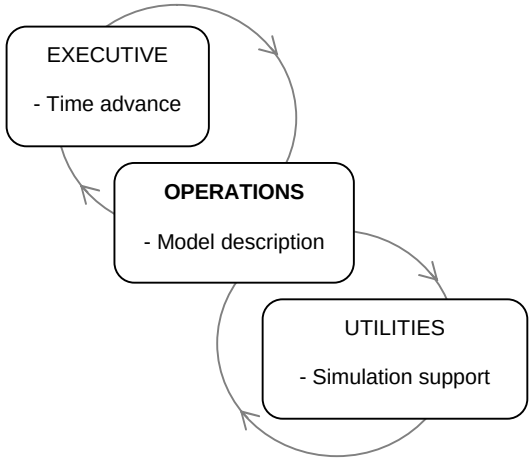

Fig. 4.1 The three basic blocks of a discrete simulator

Although this structure continues to be very useful for didactical purposes, the fact is that it does not fit completely adequate to the development of certain simulators using the OOP paradigm. In effect, it can be seen as the result of an obvious strategy for compiling the source code of an application in three separated blocks. That way, most of the times the user only had to compile the *operations block*, as the others rarely would need to be updated.

On the other hand, the OOP languages often structure the process of compilation object by object, as the *methods* (internal functions) associated with a particular object are usually confined to the same source code file. The code associated with the model dynamics therefore has the tendency to spread through some objects of the application, naturally betraying the concept of a single *operations block*. The approach described herein, being an OOP proposal, is one of those cases.

4.2.2 The OOP tendency

Under an OOP description, the model behaviour is dispersed through the diverse objects that represent the active entities of the system. So, the structure of the simulator naturally gets more flexible for adapting to each programmer style, without loosing the coherence of the previous case. In a certain way, the simulator structure becomes horizontal. In our case, it was likely to follow the object structure of the *Microsoft Visual* C++, whilst the idea of the two step procedure was maintained: first model, then simulate.

The model is created by means of identifying and describing the appropriate *entities* of the system, which are seen as elements linked to their very own states (activities), and frequently containing their very own "*operations block*" too. The description of the dynamics of each *entity* eventually implies access to some resources of support (those previously described in the *utilities block*), but most of these resources are now objects (as the *dialog-boxes* used for configuring and ensuring the human interface with the entities, for example), or even methods belonging to other more complex objects, as to the object <*Simulator*>, which in this approach is also the "owner" of the *executive*, as we will see[1].

This vision of entities owning their individual states, in effect contrasts with the old tendency of viewing the states as belonging to the system, which was a typical perspective of straight programming. Here, to the contrary, the states of the system are seen to result from the collection of the states belonging to the entities. Entities give states to the system, so, a system without entities is a system without states. Figure 4.2 represents this new type of structure.

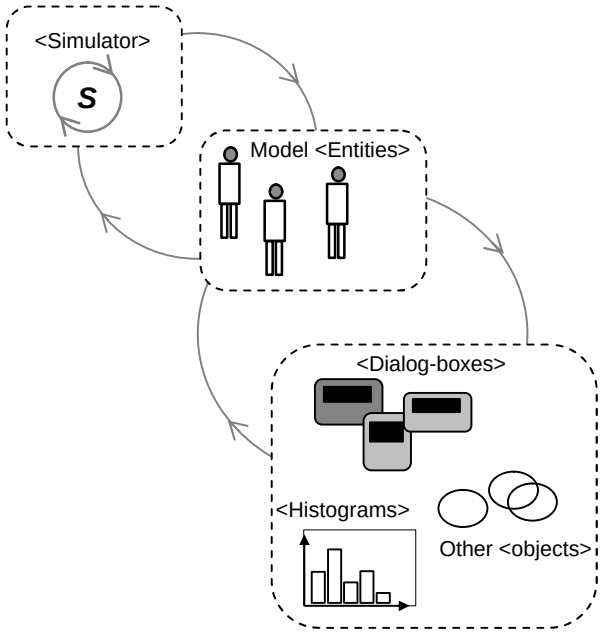

Fig. 4.2 The simulator OOP general structure

As one can see, it is still possible to find similarities between this structure and the classic proposal, if the elements are separated by means of the three previous criteria, but under a superior level of abstraction. Also notice that now, for updating the model, it is only necessary to update the model <*entities*>, that is, the active objects in the simulation, those that contribute with their states to the state of the system.

4.3 Discrete event modelling

The method of simulation used here is the one usually known in literature as *discrete event*, or *event-scheduling*. At least to clarify how this method "looks" to the

[1] To avoid confusion with similar words along this text, we will enclose objects between brackets, like in <object>.

system states and how it handles the advance of the time, since these are critical issues in any dynamic simulation, we suggest focusing the attention a little more on its basis.

4.3.1 System states and time advance

We can begin by noting that a system is classified as *discrete* when its states are represented in a discrete way, as illustrated in the next figure (Fig. 4.3).

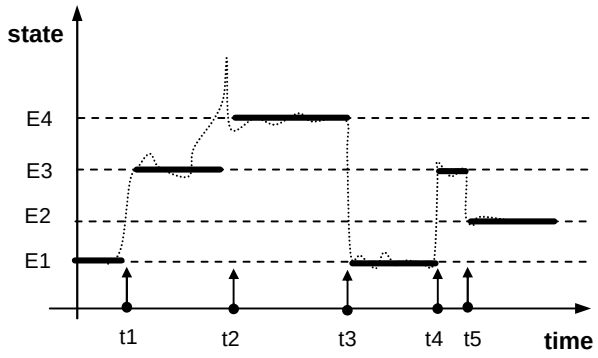

Fig. 4.3 Discrete representation of system states

The system states result, as previously noticed, from the contribution of the states associated with each entity. So, if a system would have *N* entities and each entity *M* states, this would mean that the number of system states would be M^N, that is, the possible combinations of the entities states. Supposing that a single facility of the Supply Chain has 10 states and a vehicle has 2 states, a Supply Chain of 10 facilities and 10 vehicles would mean $10^{10} + 2^{10} = 10^{10} + 1024$ states, approximately $10^{10} = 10 \times 10^9$, the equivalent of a 10 GByte computer disc of states. From this, one sees why discrete simulation is so powerful.

As one may realize from the same figure, the time in these kinds of systems can simply be made to advance solely when a new event will take place (to a state transition), that is, to the *next event*. As we will see, these events are scheduled on the "time axis" of the simulation by any of the entities during their operations, using the method *Schedule()* belonging to the <*Simulator*> object. This object will later be responsible for the maintenance of the coherence of the time, and in the appropriate moment order the current entity to execute the current event. As we know, the correct sequencing of events is usually ensured by the method *Schedule()*, which always keeps the *event-list*[2] of the simulation in chronological order.

4.3.2 Events versus activity cycles

The model dynamics are described here in terms of *state transitions diagrams*, or *activity cycle diagrams*, a collection of *live* and *dead* states defining how the entity jumps to consecutive states. In the *event approach*, however, events are used to represent the states, since in fact between consecutive events nothing happens in the system, that is, the system remains in the same state (see example of Fig. 4.4).

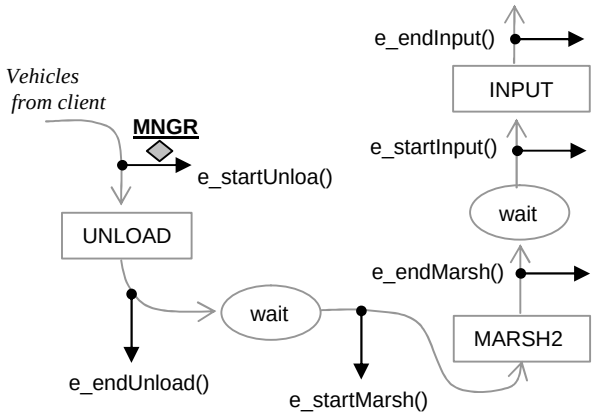

Fig. 4.4 Activities and events in the input stock process

[2] This list contains events to be executed in the future, in chronological order. It is seen as the time axis of the system.

As one can easily infer from this figure, the transformation of any *activity cycle diagram* to a *sequence of events* is easily achieved by using an *event* at the beginning and another at the end of each *live activity* in the diagram. Each event will then be made to correspond to a method.

4.3.3 The SimEvent object

The object <*SimEvent*> is the element in the simulation that unequivocally identifies the time of execution of a certain event and, as events are now property of entities, which is the entity to execute it. The <*SimEvent*> has the following C++ structure:

```
class SimEvent : public CObject
{
public:
float          m_time; //event time
UINT           m_event; //event ID
SimEntity*     m_pEntity; //entity
void*          m_ppp; //generic pointer
};
```

Notice that the attribute *m_time* will carry the time when the event must be executed in the simulation; the attribute *m_event* represents a unique *identifier* (ID) for the event type to be executed, and previously defined in the system; *m_pEntity* is a pointer linked directly to the entity that executes the event; and *m_ppp* is a general purpose pointer which have been added to pass to the entity any sort of information, or even other entities. This pointer, as a new proposal (commonly the <*SimEvent*> is reduced to the other 3 attributes), plays an important role in the interface between different entities and in passing information on to different processes.

4.4 The basic entity

Any entity in the simulation will have in its structure, apart from the traditional attributes and methods typical of any class of objects, also methods corresponding to the events associated with it and specific to its behaviour. In general, as figure 4.5 suggests, an entity is described by attributes and a collection of states. These states represent the different situations in which the entity can be found in the simulation.

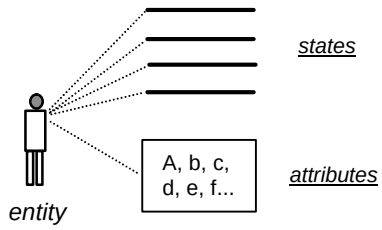

Fig. 4.5 Attributes and states of a generic entity

In this *next-event* approach, such states are linked to the entity by their equivalent events, which are implemented as *event methods*. So, the entity must be able to execute the respective *event method* each time it receives the ID of one of its events. This is ensured by an additional method (a *rooter*) which can be called from "outside" the entity. We have named this method *Execute()*, as shown in the next figure.

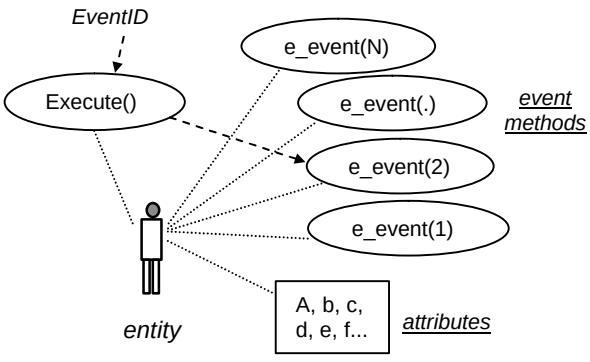

Fig. 4.6 Attributes and methods of a generic entity

In the figure, the *event ID* is connected by the *Execute()* method to the *event method* named *e_event(2)*, as an example. By norm, we will add the prefix "*e_*" to the name of any *event method*, so that it becomes easily distinguishable from other methods.

4.4.1 The SimEntity object

The entities will be modelled as elements deriving from a primer object class containing the basis for the description of any entity. We have therefore built a class named <*SimEntity*> for containing all the properties that will be common to any future entity. This generic entity includes two event methods named *e_SimShow()* and *e_SimNewDay()*. The first is responsible for the representation of the entity in the display window of the application; the second is related to the time and gives the entity a sense of the days passing. This last aspect is used by some entities to take decisions. A simplified version of the <*SimEntity*> is shown in the figure 4.7, while the complete structure of the class is presented afterwards.

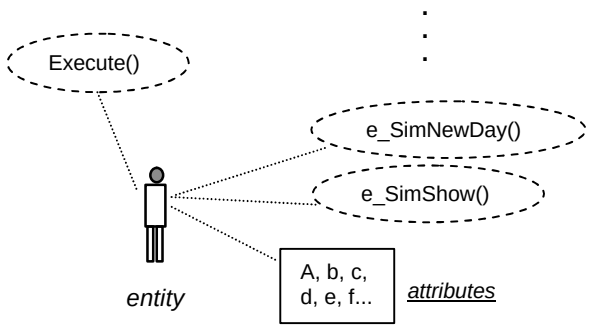

Fig. 4.7 The generic SimEntity element

The methods of this basic entity are shown in dashed ellipses to remind that they are *virtual* methods, and therefore they do not necessarily need to be implemented in the upper level entities. Even so, they can be accessed by direct call from any entities, as default methods.

Here is the entire <*SimEntity*> prototype:

```
class SimEntity : public CObject
{
public:
BOOL    m_livre; //logic variable for general use
CString m_text; //auxiliary text
CString m_nome; //name of the entity
CString m_ref; //reference of the entity
float   m_time; //time of the entity (simulation)
float   m_initdayTime; //first day
UINT    m_estado; //state of the entity
Simulator* m_pSim; //pointer to the <Simulator>
UINT    m_tipo; //entity type
float   m_x; //entity x position in the theatre
float   m_y; //entity y position in the theatre
int     m_nivel; //entity level

SimEntity(CString nome); //public constructor
virtual void Serialize(CArchive& ar); //serializer

protected:
SimEntity(); //private constructor

//Methods related with the simulation:
public:
virtual BOOL Execute(UINT event, void* ppp=NULL);//rooter
virtual void e_SimShow(void* ppp=NULL); //show
virtual void e_SimNewDay(void* ppp=NULL); //day time

DECLARE_SERIAL(SimEntity)
};
```

As previously said, analysing all the characteristics of each object is not the intent of this text, so we will simply make reference to the most relevant ones.

It is in particular interesting to notice that a pointer to the <*Simulator*> is kept in the attribute *m_pSim*, so that the entity can associate itself with the instance of the simulator it belongs to. This way the entity will have access to certain important methods of the <*Simulator*>, for instance, to the *Schedule()* method.

Other important methods listed in this class are the *Serialize()*, responsible for saving (and opening) the object data to (and from) the computer disk; and the two *SimEntity()* methods, which are constructors of the objects of the class and can be used to implement a default configuration. These methods, however, are part of the *Microsoft Visual* C++ structure (please see Microsoft, 2000), therefore we will avoid commenting on them in detail.

4.5 Listing the event IDs

Each event that may probably occur in the simulation must previously be declared in the application by introducing a unique ID in a section of *#define* statements. This was made in the file "*events.h*". For example, the next lines of code show a little part of this file where some event IDs related with the CSU entity are declared:

```
//.....
#define CSU_ORDER_ARRIVE    234 //client order arrives
#define CSU_ORDER_IN        232 //initial inspection
#define CSU_ORDER_NOW       230 //order to supplier
#define CSU_START_OUTPUT    226 //start output activity
#define CSU_END_OUTPUT      224 //end output activity
//.....
```

4.6 Coding the states (activities)

As the entities are objects, and the states of the system are seen as places where the entities may pass, or sometimes even agglomerate, these are implemented in the form of *lists of objects*. For this purpose we use the C++ type known as *CPtrList*, which implicitly already has all the functionality needed for manipulating a list of objects, as the appropriate methods for adding elements, removing elements, etc. Please refer to Microsoft (2000) for further details concerning this type. The system will move from state to state as the entities will be moved from list to list. The next lines of code represent some examples of states implemented as object lists.

```
//some examples of dead activities (queues).....
CPtrList        m_inorderList; //orders to the CSU
CPtrList        m_waitOutputList; //wait for output
CPtrList        m_waitMarshList; //waiting marshaling
CPtrList        m_waitLoadList; //waiting load
CPtrList        m_waitProdList; //wait production
CPtrList        m_inputBaysList; //wait input bay
CPtrList        m_outputBaysList; //wait output bay

//some examples of live states (live activities)...
CPtrList        m_onOutput; //in output
CPtrList        m_onMarsh; //in marshaling
CPtrList        m_onLoad; //in loading
CPtrList        m_onUnload; //in unload
CPtrList        m_onProduction; //in production
```

Notice that even if the *activities* belong to the entities, those clearly must be physically located outside the entities. The entities may transport the rules for jumping from *activity* to *activity* (in the form of event methods), but their *activities* are located at the object(s) with which they interface.

4.7 The simulator object

This is the object that has the complete responsibility of the simulation process. Once the system is properly modelled, the <*Simulator*> may be launched in the application whenever the user decides to start the simulation process. This object is created in run-time as a window which automatically links itself to the model previously created and configured by the user. Its first task is to build an "image" of the model by means of pointers to all the model objects. Then, it executes the routine *init()* where the consistency of the model is verified and the initialization of the simulation takes place. If no errors are detected, the model starts to run by entering the *Loop()* method[3]. Figure 4.8 represents, in a simplistic way, the basic structure of this object.

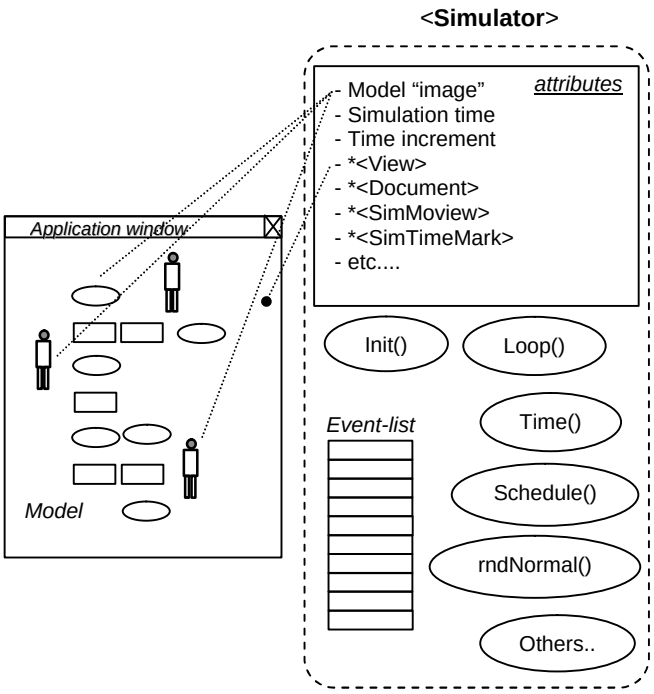

Fig. 4.8 The basic structure of the <Simulator>

[3] This method is the core of the *Executive* of this simulator, responsible for the time advance in the simulation.

Since the <*Simulator*> is the "container" of the simulation process, it is also the element that holds the methods of general use, like the *Schedule()*, by means of which a new event can be inserted into the *event-list*; the *rndNormal()*, to generate a random number from a *Normal* distribution; the *Time()*, by which the entities can retrieve the current simulation time; and some others. It also owns the *event-list*, in order to keep the correct track of the simulation time, a list holding objects <*SimEvent*> and implemented by means of a *CPtrList*.

In its attributes section, the <*Simulator*> also includes a pointer to the application's <*View*>, the window where the objects are graphically represented and where the movements associated with the simulation take place, and a pointer to the respective <*Document*>, the standard "data base" of the application, which in fact holds the entire model. The model is saved to and opened from the computer hard disc directly to the <*Document*>, by means of *Serialize()* methods.

Without any additional discussion, here we show the entire C++ prototype of this object, with some commentaries:

```
class Simulator : public CWnd
{
public:
        Simulator(View* pView); //constructor
float   Time() {return m_time;} //time method
BOOL    Init(); //initialization
BOOL    Loop(); //simulation Executive
void    Schedule(float time, UINT event, SimEntity* pEnt,
void* ppp=NULL);
float   rndNormal(float med, float dp);//qualquer intervalo
void    OutputReportToFile(); //final report to file
void    FinalCalculations(); //Supply Chain metrics
void    CleanAll(); //clean all objects for the next replication
```

```
void      LogText(CString text); //print to log file

//Attributes
CPtrList *m_veiculoList; //image of the vehicles
CTypedPtrList<CObList, Nodo*> *m_nodoList; // nodes
protected:
void      LancaNodosVias(BOOL preview); //init nodes
void      LigaNodos(); //link nodes
void      LancaFrotas(); //link fleets

//physical objects in the theatre:
CTypedPtrList<CObList, Area*>     *m_areaList; //areas
CTypedPtrList<CObList, Via*>      *m_viaList; //paths

//private to the simulator
float     m_time; //simulation time
CPtrList  m_eventList; //event-list

public:
CTypedPtrList<CObList, Product*> *m_productList;
CPtrList  m_objMovingList; //objects moving list
CPtrList  m_ordersList; //orders launched in the simulation

float     m_stepTime; //time increment
float     m_endTime; //end time for the simulation
float     m_maxCostsIndex; //for fixed costs
View      *m_view; //application's view
int       m_seed; //random generator seed
int       m_nReplications; //number of replications
FILE*     m_fileLog; //file for simulation log

protected:
Document        *m_doc; //application's document
SimMovie        *m_simShow; //visual effects
SimTimeMark     *m_simTimeMark; //day marks

public:
virtual ~Simulator(); //destructor
DECLARE_MESSAGE_MAP()
};
```

4.7.1 The schedule method

The method used by any active element of the simulation to schedule events for the future in the *event-list* is the *Schedule()*. The entire code of this method is relatively long, so we will "reduce" it here to the essential and some comments.

```
void Simulator::Schedule(float time, UINT event, SimEntity* pEnt,
void* ppp)
{
    SimEvent      *curEvent = new SimEvent(); //current event
    SimEvent      *timeCell; //event of the event-list

    //configure the current event with the method's data
    curEvent->m_time = time;
    curEvent->m_event = event;
    curEvent->m_pEntity = pEnt;
    curEvent->m_ppp = ppp;

    POSITION pos2;
    POSITION pos = m_eventList.GetHeadPosition(); //head of list

    while(pos != NULL) //while not the event-list end
        {
        pos2 = pos;
        timeCell = (SimEvent*)m_eventList.GetAt(pos);
        UINT cellEvent = timeCell->m_event;
        ////////////////////////////////////////////////////////////////////
        //insert the new event in the appropriate place of
        //the event-list in order that the entire list
        //will be maintained in a chronologic order,
        //comparing "time" with "timeCell->m_time"
        //and the priority of the even what the times
        //are coincident, by means of the inferior "eventID"
        //.....(Code not shown here).....
        ////////////////////////////////////////////////////////////////////
        m_eventList.GetNext(pos);
        }
    //event-list empty or at the end:
    if(pos == NULL) m_eventList.AddTail(curEvent);
}
```

4.7.2 Random *Normal* generator

This being a stochastic approach, some model variables must be configured taking into account the associated uncertainty. In this version of the simulator this is achieved by means of a *Normal* random number generator inspired by Everett (2001) and following the procedure described in the common Box-Muller method (Box & Muller, 1958). No other statistical distributions had been implemented in this version, as the main intent was to test the concepts and the viability of the Supply Chain modelling proposal. So, any inputted variability is at the moment expressed in terms of a *Normal* distribution of values, represented by an *average* and a *standard deviation*. The code of the generator is the following:

```
float Simulator::rndNormal(float avrg, float dp)
{
    //returns the average if simulator set to deterministic
    if(m_view->m_useRandom==FALSE) return avrg;

    float a1, a2, w, rnd1, rnd2;

    //Theory:////////////////////////
    float pi=3.1415926535f;
    float a1= (float) rand()/RAND_MAX; //between [0,1]
    float a2= (float) rand()/RAND_MAX; //between [0,1]
    float loga1=(float) log(a1);
    float cosa2=(float) cos(2.0f*pi*a2);
    //Normal random number, between [0,1]:
    float rnd1= (float) sqrt(-2.0f*loga1)*cosa2;
    ////////////////////////////////////////////////////////////
    //Fast algorithm due to Box&Muller:
    do    {
          a1 = -1.0f + 2.0f * rand()/RAND_MAX;
          a2 = -1.0f + 2.0f * rand()/RAND_MAX;
          w = a1 * a1 + a2 * a2;
          } while( w >= 1.0f );
    w = (float) sqrt( (-2.0f * log( w )) / w );
    rnd1 = a1 * w; //first variate [0,1]
    rnd2 = a2 * w; //second variate [0,1]

    //.....................................................
    //To compute a Normal number with average "avrg"
    //and standard deviation "dp" we simply have to scale
    //one of these [0,1] Normal random numbers and
    //add it to the average value: (z=avrg+rnd*dp)

    return(max(0, avrg+rn1*dp)); //return the Normal number
}
```

Notice that there must be some caution in using this method for numbers near 0, since for convenience we have filtered it in order not to return negative values. As frequently event times are computed by means of a call to this function, the generation of negative numbers would imply scheduling events for the "past", which must be forbidden due to the instability it would provoke in the simulator.

4.7.3 The executive loop

As previously said, the method *Loop()* is the core of the *Executive* of the simulator. This method is continuously called during the simulation run and it is responsible for retrieving the *current event* from the top of the *event-list*, and directing it to the respective entity. The entity will then execute it by calling the appropriate *event method* (or *event handler*).

Apart from this, this method is also in charge of detecting the event associated with the end of simulation (SIM_END), which will stop the current simulation process, leading the application to reinitiate for the next replication, if any is yet to come. The simulation process is complete when all the replications previously defined by the user

are complete. Here we give the code of this method:

```
BOOL Simulator::Loop()
{
 SimEvent* event;
 UINT eventID;
 SimEntity *entity;
 BOOL end=FALSE;

 if(!m_eventList.IsEmpty()) //if event-list not empty
    {
    //Remove next event from the top of the event-list
    event = (SimEvent*) m_eventList.RemoveHead();

    m_time = event->m_time; //simulation time = event time
    eventID = event->m_event;//get the ID of the event
    entity = (SimEntity*) event->m_pEntity; //event's entity

    m_view->m_actSimTime = m_time; //<view> time

    if(eventID == SIM_END) //if it is end of simulation
          {
          m_nReplications--; //replication count
          FinalCalculations(); //final calculations
          end=TRUE;
          }
    else //if it is a normal event
          {
           //link the entity to this simulator
           entity->m_pSim = this;
          if(m_view->m_simDebug) LogText(box);//debug
           //pass the event execution to the entity
           entity->Execute(eventID, event->m_ppp);
          }
    delete event; //free the memory
    }
 return end;
}
```

Even if we have named this method *Loop()*, any attentive reader would have noticed that in fact it does not represent a loop of code. The reason for this is that the *Executive* loop have been implemented here in a non-traditional manner, in order to guarantee the access to the model elements even while the simulation is running. Usually, the *Loop()* method is based on a *while* statement, for example, but that way the process associated with it would use too many resources of the processor, inhibiting the operating system to dispatch the windows messages from the keyboard or the mouse. This means that the user would have to wait for the end of simulation to be able again to change the configuration of the model elements. An interesting feature, however, would be to enable the user to make those changes at any moment, something that was perfectly achieved with the present proposal, by means of calling the *Loop()* method from an object external to the <*Simulator*>. Such a scheme will be explained in the next section.

4.7.4 The SimDlg loop controller

The complete *Executive* loop is slightly more complex in this simulator than the traditional one, since a new element is introduced in the loop (a loop controller). This element, the <*SimDlg*> dialog window, implemented in the form of an overlapping *dialog-box*, is inserted between the old *Executive* and the actions of the entities. The next figure (Fig. 4.9) represents the entire *Executive* loop and the way it is linked to the grand simulation cycle. Notice that the <*SimDlg*> object acts as being a stand-alone application linked to the <*Simulator*> by means of a pointer. So, the method *Simulator::Loop()* can be called directly from there whenever the "step" button is pushed, for example. This is a very

flexible scheme, and it is also a simple method to implement the manual advance of the simulation, step by step.

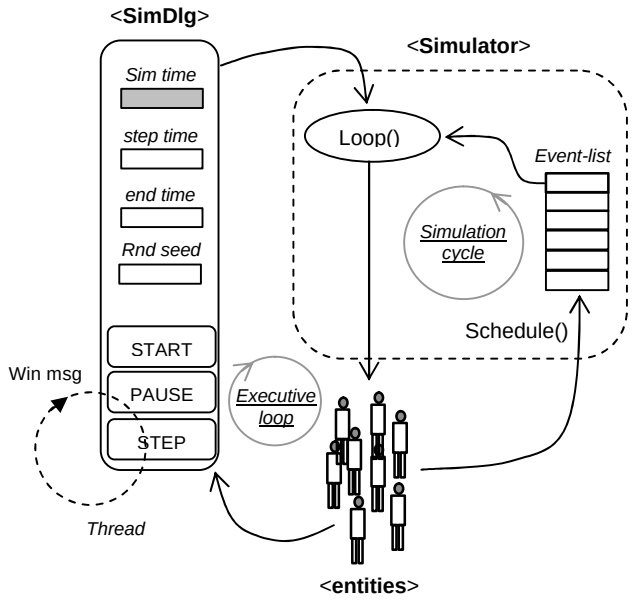

Fig. 4.9 New executive loop and the simulation cycle

The secret of this scheme is the standing alone process (*thread*) that continuously sends a *Windows* message to the <*SimDlg*> object, simulating the pushing of the button "STEP", whenever the dialog is not in the "PAUSE" state. In response to this message the <*SimDlg*> directly calls the *Loop()* method of the <*Simulator*>. Such a "switch" of processes liberates the application for enough time so that it can process the keyboard and mouse events, allowing the user to interfere with the model entities even during the simulation run.

The implementation of this *thread* was based on the *CWinThread* class, offered by the C++ package, and the respective thread function is simply written as:

```
UINT myThreadFunc(LPVOID pParam)
{
    SimDlg* simdlg = (SimDlg*)pParam; //link to <SimDlg>

    while(!simdlg->m_paused)
    simdlg->SendMessage(WM_COMMAND, ID_STEP, 0);

    return 1;
}
```

This is a standing alone loop that will be started by the button "START" or at the beginning of each replication, and will only finish at the end of a replication. Apart from that, it will be sending ID_STEP messages to the <*SimDlg*> object, which simulate the push of the button "STEP".

For those interested in knowing how the C++ code of a *dialog-box* looks like, here we show the example of the <*SimDlg*> object[4]:

```
class SimDlg : public CDialog
{
//Construction
public:
    SimDlg(CWnd* pParent); //standard constructor

//Dialog Data
    //{{AFX_DATA(SimDlg)
    enum { IDD = IDD_SIM_INIT };
    float  m_time; //simulation time
    int    m_seed; //random generator seed
    float  m_endTime; //time to end the simulation
    float  m_stepTime; //step on the simulation advance
    float  m_timeDays; //time in days
    //}}AFX_DATA
    View*          m_view; //associated view of the model
    Simulator*     m_pSim; //associated simulator
```

[4] Every *dialog-box* code used in this simulator has been automatic generated by the C++ package and is considered basic for a C++ programmer. Thus, we will avoid showing such type of code from now on, unless strictly necessary.

```
    CWinThread* mySimLoop; //thread for "step" messages
    BOOL        m_paused; //pause state

//Overrides
    //ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(SimDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL

//Implementation
protected:
    void StartThread();

//Generated message map functions
    //{{AFX_MSG(SimDlg)
    afx_msg void OnRun(); //button START handler
    virtual void OnCancel(); //Cancel handler
    afx_msg void OnStep(); //button STEP handler
    afx_msg void OnPause(); //button PAUSE handler
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
};
```

4.8 Entity motion (SimMovie)

The movement of certain entities in the simulation theatre, that is, in the window of the simulation application, is ensured by another special object launched in the beginning of the simulation. This object, which we have named <*SimMovie*>, is also a standing alone entity which responds to the event ID = SIM_SHOW. As an entity, this object inherits from the class <*SimEntity*>, earlier defined. The other entities will move in the display window during the simulation process as a result of the continuous action of the <*SimMovie*>, which, in response to the event SIM_SHOW, forces each of the current entities in motion to redraw itself in the simulator window. This process auto-repeats in intervals of time *dt* = *stepTime*, previously set by the user. The scheme is represented in the next figure (Fig. 4.10).

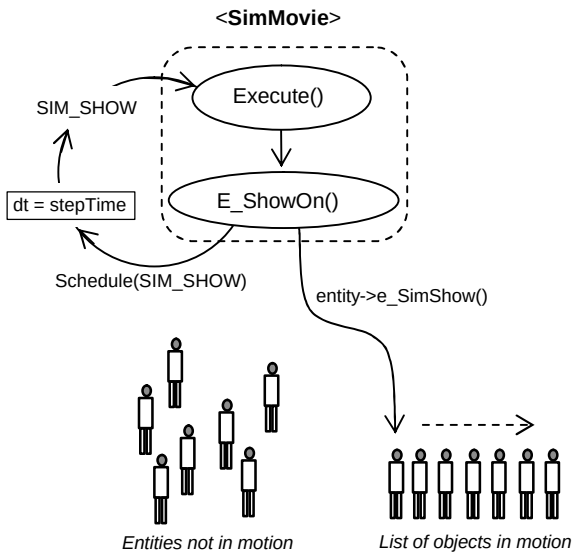

Fig. 4.10 Process for moving entities in the display

Basically, the <*SimMovie*> will be running by scheduling the event SIM_SHOW to itself, after scanning the list of objects in motion and calling the method *e_SimShow()* of each entity referenced in that list. This method will update the position of the entity in the simulator window, giving the impression of movement. Notice that the method *e_SimShow()* is also an *event handler*, so, it can be accessed through the event scheduling mechanism, instead of by direct call. Each time an entity is stopped or absent from the list of objects in motion it will be unaffected by this process, therefore appearing static in the simulator window.

The <*SimMovie*> C++ prototype has the following structure:

```
class SimMovie : public SimEntity
{
public:
    SimMovie(Simulator* pSim); //constructor

//event rooter
    virtual BOOL Execute(UINT event, void* ppp=NULL);

protected:
    BOOL e_ShowOn(); //SIM_SHOW event handler
};
```

Notice that the event SIM_SHOW is received from the "exterior" by the *rooter Execute()* and then processed by the event handler *e_ShowOn()*.

We will use this opportunity to present the code of a simple *Execute()* method, the *event rooter* of the entity:

```
BOOL SimMovie::Execute(UINT event, void* ppp)
{
    BOOL ok = FALSE;
    if(event == SIM_SHOW) ok = e_ShowOn();
    return ok;
}
```

And finally, the code of the event handler *e_ShowOn()*, which processes the SIM_SHOW event. Notice the re-scheduling of the same event at the end of the method, after the interaction with the entities in the list of objects in motion:

```
BOOL SimMovie::e_ShowOn()
{
SimEntity* pobject;

m_time = m_pSim->Time(); //present simulation time

//.....
//inspect the list of objects moving in the simulation
//and order each of its objects to draw itself in the
//simulator <view> (display window)

POSITION pos;
pos = m_pSim->m_objMovingList.GetHeadPosition();
while(pos!=NULL) //all the list
    {
     pobject = m_pSim->m_objMovingList.GetNext(pos);
     if(pobject->m_estado!=STOPPED)
          pobject->e_SimShow(); //draw it
    }

//Schedule the next event SIM_SHOW
float dT = m_pSim->m_stepTime; //use the step time
m_pSim->Schedule(m_time + dT, SIM_SHOW, this);

return TRUE;
}
```

4.9 A daily mark (SimTimeMark)

There is another auto-repeating process which is launched at the beginning of the simulation process with the intent of regularly sending to each moving entity the time mark of a new day. This is done by the entity <*SimTimeMark*>. The process is very similar to that described in the previous section for the <*SimMovie*> object, but now it deals with the SIM_NEWDAY event ID, which is re-scheduled 24 in 24 hours. This event is re-scheduled at the end of the *e_SimNewDay()* method, its *handler*, after the *e_SimNewDay()* of each entity in the list of objects in motion is called.

We refrain from presenting here the code related to this entity, due to its similarity with the previous case. We observe, however, that the CSU object uses this time mark to analyse the orders waiting for being placed to the suppliers and to possibly

rearrange them the best possible way. The CSU method *e_SimNewDay()* has the following aspect:

```
void CSU::e_SimNewDay(void* ppp)
{
    //time of the beginning of the day
    m_initdayTime = m_time = m_pSim->Time();//sim time

    if(!m_toOrderList.IsEmpty())
      CSUManager(m_time, CSU_ORDER_NOW, NULL);
}
```

The basic idea is to inspect the *material orders* accumulated during the day in the list *m_toOrderList*, and call the *CSU-manager* with a request to "order-now" that material. The logic and criteria associated with this process will belong to the *CSU-manager*.

4.10 Some utilities

Apart from the normal utilities for *password access* to the application or to *add text* commentaries to the simulator window while the model is being developed, at least two other important utilities deserve to be mentioned. These are the <*SPECTRUM*> and the <*GraphDlg*> classes. The first is extensively used to record large amounts of data during the simulation process; the second is a type of window where such data can dynamically be presented to the user.

4.10.1 The SPECTRUM class

This name comes from several years ago when I was working with light transmission spectra in a Physics laboratory. The code was conceived at that time and meanwhile I begun to use it in almost all the software needing graphs visualisation. So, the term stayed. Basically, it is a structure of a chart as a general collection of data points (Y_j) of the form $Y_j(x)$, where x is considered any sort of continuous variable:

```
class SPECTRUM
{
 public:
    float  LMin; //minimum value in X axis
    float  LMax; //maximum value in X axis
    float  dL;  //step in the X axis
    float  Ymin; //minimum value in Y axis
    float  Ymax; //maximum value in X axis
    CString NomeX; //title of X axis
    CString NomeY; //title of Y axis
    CString NomeGraf; //title of the graph
    CArray<FPoint, FPoint> Data; //array of data points
    short okDone; //control
    float  mean; //average value of the Y data points
    float  stdev; //standard deviation of Y data points
    float  total; //total summation of Y values (integral)
    float  ymin; //minimum value of the Y data
    float  ymax; //maximum value of the Y data
    COLORREF color;
    void* window; //window to which this object is connected

    SPECTRUM(); //constructor
    void Calculate(); //compute average, stdev, etc.
    void Clean(); //clean all data
};
```

Notice that to compute the average, the standard deviation, minimum and maximum and the integral of the Y_j values, recorded in the *Data* array, it is simply needed to call the method *Calculate()*, implemented by the following code:

```
void SPECTRUM::Calculate()
{
int np;
int N = this->Data.GetSize(); //number of data points
if (N <= 0) return; //abandon if there are no points
```

```
float minimo = 99999.9f;
float maximo = -99999.9f;
float sum = 0.0f;

//compute average, minimum, maximum and integral
for(np=0; np<N; np++)
    {
    if(this->Data[np].y < minimo) minimo = this->Data[np].y;
    if(this->Data[np].y > maximo) maximo = this->Data[np].y;
    sum += this->Data[np].y; //accumulate (integral)
    }
this->mean = sum/N; //record the average
this->total = sum; //record the integral
this->ymin = minimo; //record the minimum
this->ymax = maximo;//record the maximum

//compute standard deviation
float ds;
double sumds = 0.0f;
for(np=0; np<N; np++)
    {
    ds = (this->Data[np].y - this->mean);
    sumds += ds*(ds/(N-1));//accumulate squared deviations
    }
this->stdev = (float)sqrt(sumds/N); //compute, record stdev
this->okDone = 1; //computation done
}
```

4.10.2 The graph window (GraphDlg)

Using collections of data registered in the form of <*SPECTRUM*> objects, <*GraphDlg*> is an object prepared to show this data in an interactive window either in the form of a standard *Y(x)* graph or in the form of a histogram. This is a very useful tool for representing graphical results, as an object of this type can be linked to any kind of object in the application. So, entities and other objects can simply record their data in <*SPECTRUM*> objects and then make it visible by means of a <*GraphDlg*>. The examples shown in figures 4.11 and 4.12 are two of the several <*GraphDlg*> associated to the CSU object. The first represents the variation of time delays in the processes of delivering and purchasing, and the second several Supply Chain measures after three simulation replications.

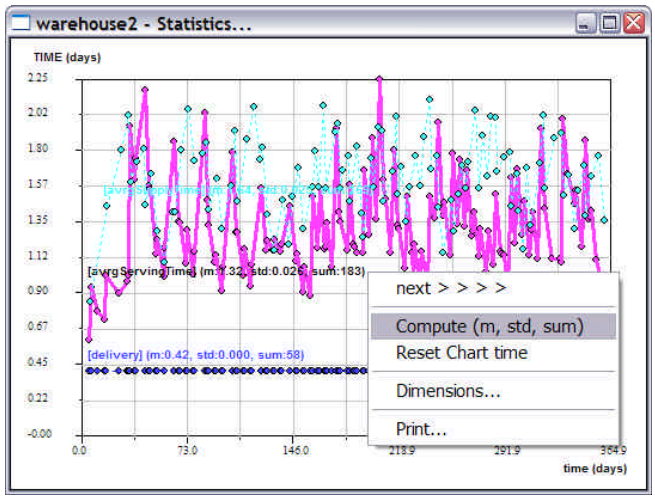

Fig. 4.11 Interactive <GraphDlg> used with time delays

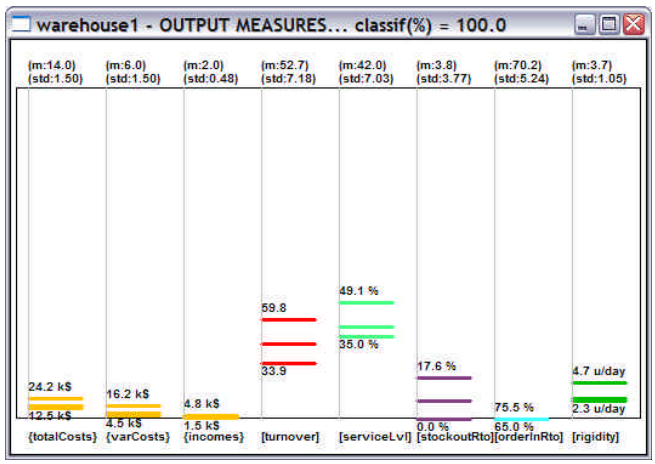

Fig. 4.12 Histogram type chart with a <GraphDlg>

In any case, the interactivity with these windows is made by means of a right-mouse-click floating-menu, which gives the user the possibility of redoing the calculations or of inspecting a part of the data in more detail, by adjusting the dimensions of the window.

The prototype of the object <*GraphDlg*> is presented in the next lines. Please observe the *Windows* event handlers related to the mouse operation are shown in the last section of the class.

```
class GraphDlg : public CDialog
{
//Construction
public:
    GraphDlg(CWnd* pParent = NULL);//constructor
    void Histogram(CDC* pdc, CRect* prect);
    void Graph(CDC* pdc, CRect* prect);
    SPECTRUM*  m_spectrum; //the active sectrum
    CPtrList       m_spectrumList; //list of spectrums
    SimEntity*     m_owner; //the entity who owns this
    int            m_graphType; //Normal or Histogram
    CPoint         m_po;
    CPoint         m_pf;
    float          m_xpixel;
    float          m_ypixel;
    int            m_bold;

//Dialog Data
    //{{AFX_DATA(GraphDlg)
    enum { IDD = IDD_GRAPH1};
    //}}AFX_DATA

//Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(GraphDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);
    //}}AFX_VIRTUAL

//Implementation
protected:
//Generated message map functions
//{{AFX_MSG(GraphDlg)
afx_msg void OnPaint();
afx_msg void OnSize(UINT nType, int cx, int cy);
afx_msg void OnClose();
afx_msg void OnMenuMore();
afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
afx_msg void OnMenuGraphDimxy();
afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
afx_msg void OnMouseMove(UINT nFlags, CPoint point);
afx_msg void OnMenuGraphBold();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};
```

4.10.3 Visual modelling (*Area*)

In addition to these utilities, it is important to mention that the creation of the Supply Chain model is based on the manipulation of objects which represent a type of "scaled" elements of reality. Obviously, a scaled model of a warehouse building is not relevant to the Supply Chain simulation, but on the other hand the itineraries for transportation and distances can be interestingly represented by scaled factors. When talking about scaled elements of reality we therefore are mainly talking about an iconic representation of those elements in the operations theatre, as usually happens in a game.

The operations theatre is defined as a sort of rectangle with dimensions measured in kilometres squared, where the facilities are freely located by the user. Distances and paths can in effect be made scaled to the reality, but the rest of the entities are represented by icons or small drawings. In any case, as previously noticed, each entity has the ability of drawing itself in that application's window. This can be done by calling its method *e_SimShow()*, but in fact this method will directly or indirectly call the more general method *Area::Draw()*, belonging to the class <*Area*>, a class from which any object must inherit if needing

visual representation (see Fig. 4.13). Thus, each of these objects may have its own associated visual symbol.

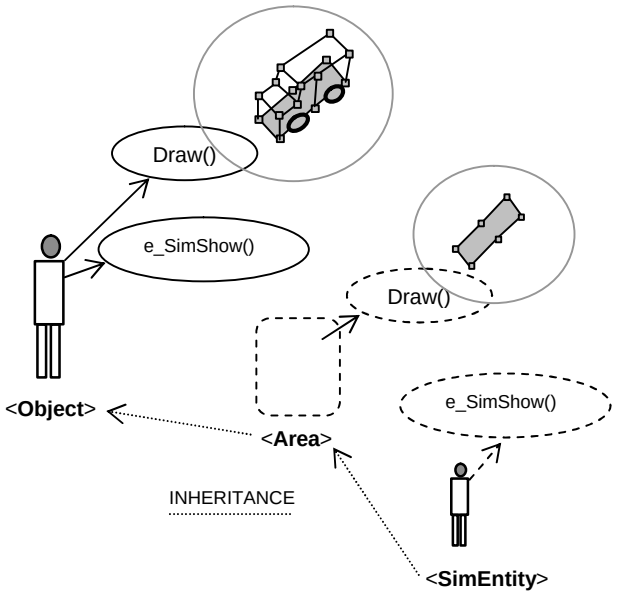

Fig. 4.13 The basic representation of a visual <Object>

The <*Object*> to be represented inherits form the class <*Area*>. This class has a basic *Draw()* method, normally used to represent the object in its basic form, as a collection of points and lines defined when the object is created in the application. On the other hand, the method *Object::Draw()* can be made responsible for a more elaborated visual representation. In the case of the CSU, for example, this method draws a suggestive icon in the coordinates where the facility is located. The class <*Area*> has the following code structure:

```
class Area : public SimEntity
{
public: //Attributes
    COLORREF   m_color; //base line colour
    UINT       m_nPenWidth; //line width
    FPoint     m_pfCentro; //area centre (xo, yo)
    SHORT      m_selectON; //selected
    CRect      m_rectBound; //rectangle container
    CArray<FPoint, FPoint> m_fPoint; //array of points

public: //Methods
        Area(); //constructor
void    Bound(); //create bounding rectangle
void    Set(FPoint pti, FPoint ptf);
void    Move(View* pView, FPoint pti, FPoint ptf);
virtual BOOL Draw( CDC* pdc ); //the Draw method
virtual void MoveTo(View* pView, FPoint ptf);
virtual void RotateAlfa(float teta,float sinal,float dx,float dy);
virtual void Translation(float dx, float dy);
virtual void CopyTo(Area* pElem );
virtual void Serialize( CArchive& ar );

DECLARE_SERIAL(Area)
};
```

As we can see, this class can offer some more utilities than just the *Draw()* method. It also implements methods for executing spatial transformations in the object, as well as for copying it to other objects of its kind, and to *Serialize* it in the computer disk. The <*View*> of the application is the object owning the specific resources to perform the job of representing, locating, moving and configure all the objects of the model, as well as the facility of *zooming*, as we will see soon.

4.11 Supply Chain entities

Once the basics of the simulator are understood, the moment to pay attention to the implementation of the Supply Chain entities finally arrives. These elements are in reality the core of the models with which the results presented in the next chapter have been obtained. Therefore, the presentation of such objects will be slightly more detailed than in the previous cases. In any case, we will maintain the practice of

presenting the information also based in general diagrams, when appropriate.

4.11.1 Paths

The concept of *path* is implemented by the class <*Via*>, which means "*path*" in Portuguese. The *path* is a sequence of line segments that will terminate in a *node*. This concept includes not only the idea of a path for transporting materials but as well the idea of a path of information. In effect, even if most of the times the *path* is used for transporting materials by means of vehicles, a certain type of path may be used to model the *process of ordering* to any facility in the Supply Chain, similar to the link of information established while ordering by telephone, catalogue or even *Internet*. Figure 4.14 aims to represent with some detail the concept of a *path*.

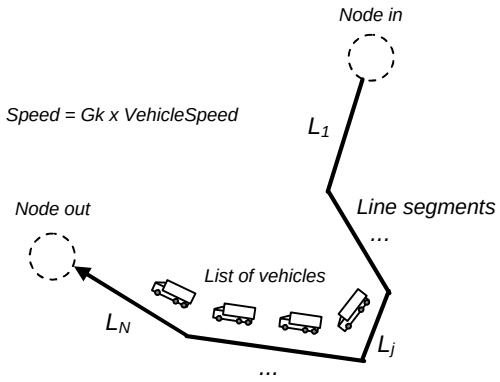

Fig. 4.14 The concept of a path (object <*Via*>)

Notice that a *list of vehicles*, a node of input and a node of output are connected to this element. Each path must be defined with a specific type: ROAD, RAIL, AIR, SEA, INFORMATION, or XTUNNEL. This last type (named *xtunnel* while we were imagining a non-stop tunnel of running products) is used to simulate the transport process of a kind of a pipeline, where the materials ordered always have a transport resource available, that is, there is no need for waiting for vehicle availability.

The total path length is the summation of the lengths of the individual segments; it is automatically computed at the moment of the object creation and then ascribed to the path as an attribute. This, of course, helps to speed up further calculations.

Besides the attributes inherited from the class <*Area*>, this element also has attributes related to costs of utilization, typical delay for the flow of information, allowed maximum transportation speed, etc., as we will see in the code of its class.

A special attribute, however, is the geometric factor (*Gk)* used to adapt the speed of the vehicles when the path is forced to a certain predefined total length. As this usually destroys the correct geometric scale of the path, compared with the scale of the drawing, and as travel times must not be affected by such operations, a geometric corrective factor must be introduced in the speed of the vehicle. This factor is simply given by:

$$Gk = S_0 / S_1 \qquad (4.1)$$

Where, relative to the scale of the drawing, S_0 represents the original path length and S_1 the current path length. The new speed of the vehicle must therefore be computed as:

$$Speed_1 = Speed_0 / Gk \qquad (4.2)$$

This avoids the need for using geographic coordinates to model long distances. At any time, we can simply force the path to have

a certain total length. Then, each time a vehicle enters the path it regulates its speed by equation 4.2.

Here is the definition of the class <*Via*>:

```
class Via : public Area
{
 public: //Attributes:
    UINT m_viaTipo; //Type of path
    float  m_chargeValue; //fixed cost for utilization
    BOOL m_viaCharged;//charging path
    BOOL m_chargeDestiny; //charge in the destiny
    float  m_Gk; //constant to multiply the speed
    float  m_maxVel; //maximum speed (km/h)
    float  m_dtime; //auxiliary time delay (xtunnel...)
    float  m_dtimeDp;//and stdev

    Percurso  m_tramoList; //List of line segments
    //NOTICE: This list is not serialized. It is computed
    //in the beginning of the simulation. This is more
    //interesting and versatile, as paths can be copied and
    //then originate others lists of segments.

    CPtrList  m_veiculoList; //List of vehicles in the path
    BOOL m_nodoOK; //connected to a node = yes
    Nodo* m_nodoOut; //output node
    Nodo* m_nodoIn; //input node
    float  m_comp; //total length (km) = sum of segments

 public: //Methods:
    Via(UINT viaTipo); //Construtor
    virtual BOOL Draw(CDC* pdc); //draw the path
    virtual void CopyTo(Via* pVia); //Copier
    virtual void Serialize(CArchive& ar); //Serializer

 protected:
    Via(); //private constructor

 DECLARE_SERIAL(Via)
};
```

Notice that the path is seen as a passive entity, since it does not make use of events or event handlers.

Finally, as an interesting example of a *Draw()* method, we present the *Via::Draw()* method responsible not only for the drawing of the path but also for calling the methods of drawing the respective vehicles:

```
BOOL Via::Draw(CDC* pdc)
{
    Area::Draw(pdc); //draw the path as an <area>

    Veiculo* pveic; //pointer to vehicle
    POSITION pos=m_veiculoList.GetHeadPosition();
    while(pos!=NULL) //scan the list of vehicles
          {
          pveic=(Veiculo*)m_veiculoList.GetNext(pos);
          pveic->Show(); //force the vehicle to draw
          }

 return TRUE;
}
```

4.11.2 Nodes

The concept of a *node* (object <*Nodo*>) is represented in figure 4.15. As noticed in the previous chapter, this element is prepared to operate in the modes NORMAL, PAUSE, TRANSFER and CSU, and it is automatically created by the simulator as the intersection of several *paths*. There is, for this reason, an attribute for the radius of the node (*r*), which defines a circle inside of which the *extreme points* of the *paths* must be located in order that the automatic algorithm can link them to the *node*. The node is therefore prepared to handle a *list of input paths* (paths going into the node) and a *list of output paths* (paths going out of the node).

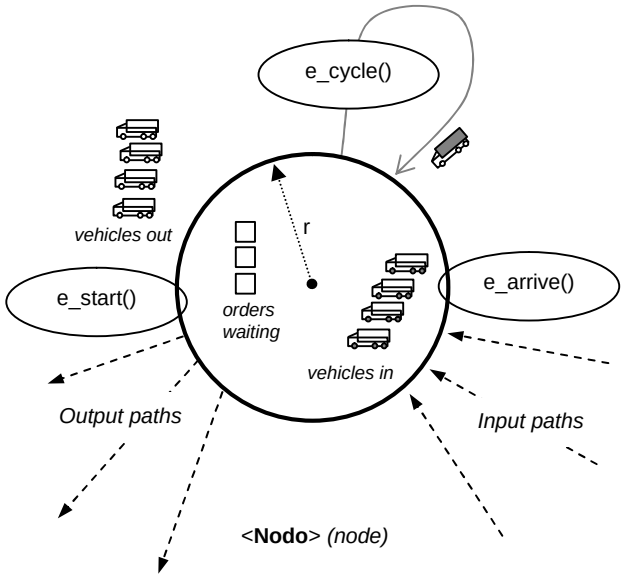

Fig. 4.15 The concept of a node (object <*Nodo*>)

At the same time, the node uses two other lists for vehicles: the *list of vehicles that entered* the node (*m_veiculoList*) and the *list of vehicles prepared to abandon* the node (*m_veicOutList*) to start (or continue) the delivery. Apart from these, a *list of orders waiting* for transportation is also included in this structure, for when the node is operating in the mode of TRANSFER. As we know, once in this mode, the vehicles arriving at the node just unload the materials and leave, transferring to the node also the responsibility for the next phase of the delivery. The transfer process may then be started in two ways: either an appropriate vehicle is automatically created inside the node to transport the materials (by trigger), or the materials will wait so that the next vehicle created cyclically (by cycle) in the node will be available to transport it. Such a scheme allows modelling interfaces with regular transports, like with regular trains, for example, and also with dedicated transports. Notice that the event *e_cycle()* controls the frequency of creating the transfers when working by cycle.

As shown in the figure, the event *e_arrive()* is responsible for "processing" the vehicles arriving at the node, while the event *e_start()* is responsible for the (re)start of the delivery process.

Notice that the other modes of operation are quite simpler: a NORMAL node will automatically transfer the vehicle arriving to the appropriate output path (leading to the customer) without inducing any delay in the process; a PAUSE node will simply do the same but scheduling the output for a certain time later, inducing a time delay in the process (as well as costs); and a CSU node will act either as a NORMAL node or as the receiver of the material.

Here follows the code with comments of the entire class <*Nodo*>:

```
class Nodo : public Area
{
 public: //Attributes
    UINT m_tipoNodo; //node type
    float   m_inputTime; //average time for creating vehicles
    float   m_inputTimeDp; //stdev
    int     m_nServico; //service number (run-time)
    float   m_raio; //radius in metres
    CPtrList   m_veiculoList; //in vehicle list (not serialized)
    CPtrList   m_veicOutList; //out vehicle list (not serialized)
    CPtrList   m_materialList; //list of orders waiting transport
    UINT m_semaforo; //0-GREEN ou 1-RED
    CSU* m_csu; //CSU of the node (if not NULL)
    float   m_unloadTimePal; //typical unload times TRANSFER
    float   m_loadTimePal; //typical load times TRANSFER
    float   m_totalChargePal; //total charge per SKU
    float   m_pauseTime; //pause time in SEMAFORO
    float   m_pauseCharge; //pause charge in SEMAFORO
    BOOL m_useByCycle; //transfer cyclically TRANSFER
    BOOL m_useByTrigger;//transfer trigger TRANSFER
    BOOL m_useManagement; //Manage the transfer process
    float   m_daysCycle; //transfer cycle definition
    float   m_daysCycleDp; //stdev
```

```
    veicData  m_roads; //road vehicles available TRANSFER
    veicData  m_trains; //rail vehicles available TRANSFER
    veicData  m_boats; //boat vehicles available TRANSFER
    veicData  m_airplanes; //flight vehicles available TRANSFER
    CTypedPtrList<CObList, Via*> m_inViaList; //input paths
    CTypedPtrList<CObList, Via*> m_outViaList; //output paths

public: //Methods
    Nodo(UINT tipo, float r); //public construtor
    virtual BOOL Draw(CDC* pdc); //draw node
    virtual void Serialize(CArchive& ar); //Serializer
    virtual void CopyTo(Nodo* pnodoTo); //copier

protected:
    Nodo(); //private construtor

//Methods related with the simulation:
public:
    virtual BOOL Execute(UINT event, void* ppp=NULL); //rooter
    virtual void e_SimShow(void* ppp); //SimShow
    BOOL e_Start(void* ppp); //start vehicle out of the node
    BOOL e_Arrive(void* ppp); //vehicle arrived at the node
    BOOL e_Cycle(void* ppp); //transfer by cycle
    void   GiveViaTo(Via* pvia, Veiculo* pveic); //associate path
    Via*   GetNextRandomVia(); //choose a path randomly

 DECLARE_SERIAL(Nodo)
};
```

Notice that the method *GiveViaTo()* has the task of assigning the appropriate *output path* to the vehicle leaving the node, after detecting the next path of its current delivery job. Since the code of this method and of the other methods are too long to present here, we will show as an illustrative example the code of the event handler *e_cycle()*, which as we know is related with the cyclical creation of transfer vehicles:

```
BOOL Nodo::e_Cycle(void* ppp)
{
m_time = m_pSim->Time(); //simulator time
Veiculo* pveic; //vehicle to be generated
veicData* veicdat; //data of the vehicle
Via* pvia; //pointer for paths

//scan the output paths to create a vehicle to each path:
POSITION pos = m_outViaList.GetHeadPosition();
while(pos!=NULL)
    {
    pvia=(Via*)m_outViaList.GetNext(pos); //*path
    if(pvia->m_viaTipo==VIA_ROAD && m_roads.m_nVeic)
        veicdat=&m_roads;
    else
    if(pvia->m_viaTipo==VIA_RAIL&& m_trains.m_nVeic)
        veicdat=&m_trains;
    else
    if(pvia->m_viaTipo==VIA_SEA && m_boats.m_nVeic)
        veicdat=&m_boats;
    else
    if(pvia->m_viaTipo==VIA_AIR && m_airplanes.m_nVeic)
        veicdat=&m_airplanes;

    pveic = new Veiculo(veicdat); //create the proper vehicle
    GiveViaTo(pvia, pveic); //assign vehicle to the path

    //Nodo::e_Start() will treat of the rest...
    pveic->m_estado=TRANSFER;
    m_veicOutList.AddTail(pveic); //put it in the outlist
    this->e_Start(pveic); //call e_Start()
    }
//.....
//shedule this same event for the next time (cycle):
float avrg = m_daysCycle;
float stdev = m_daysCycleDp;
float dtime = m_pSim->rndNormal(avrg, stdev);
float t = m_time+24.0f*dtime; //time in hours
m_pSim->Schedule(t, NODO_CYCLE, this);

return TRUE;
}
```

4.11.3 Vehicles

All the kinds of transport vehicles are implemented based on the class <*Veiculo*>, which code will be presented in this section. As described in the previous chapter, the *vehicle* is an element that moves along the Supply Chain paths in order to deliver the *material-orders* it carries to the clients. Notice that each order transported by the vehicle in its *container* has associated the *path to the client* in the form of a sequence of nodes. Each time the vehicle reaches the end of a path and arrives at a new node it notifies the node by posting to it an event NODO_ARRIVE. The node is then expected to resolve the situation, in accordance to what was exposed in the previous section.

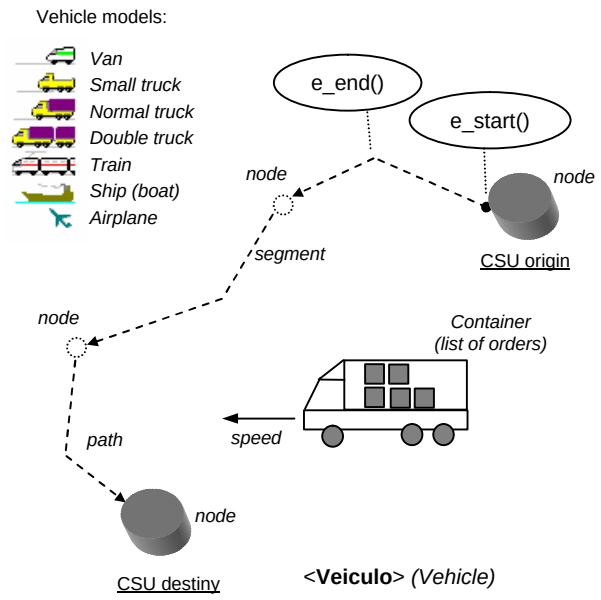

Fig. 4.16 The concept of a transport vehicle (<*Veiculo*>)

As figure 4.16 suggests, in each delivery job the vehicle also knows the *CSU of origin* and the *CSU of destiny* by inspecting the current *material-order*, and so it just has to follow the respective paths till arriving at the destiny.

This movement of the vehicle along each path is based on the events *e_start()* and *e_end()*, which code is listed later in this section. The first method *(re)starts* the vehicle in the beginning of the path, assigning to it the correct direction to go, the correct speed, etc.; the second method is in charge of *interfacing* it with the next segment or with the node in the end of the path, as we will see.

An interesting visual aspect of this class is that one may choose different graphical representations for the vehicle (models), like the form of a truck, of a train, of an airplane, etc. Such a form will then be used to reference the vehicle in any visual effects (like movement, for example). The creation of these shapes is ensured by the method *Model()*.

The code of the class <*Veiculo*> is the following:

```
class Veiculo : public Area
{
public: //Attributes
    UINT        m_sentido; //direction of movement
    UINT        m_tipoVias; //kind of allowed paths
    UINT        m_modelo; //model of the vehicle
    float       m_velMedia; //average speed
    Percurso*   m_actPercurso; //actual path
    Tramo*      m_actTramo; //actual segment
    Tramo*      m_prevTramo; //previous segment
    Tramo*      m_nextTramo; //next segment
    float       m_actLambda; //%of the segment
    Tramo*      m_tramoDestino; //destiny segment
    float       m_lambdaDestino; //% segment destiny
    float       m_fullIndex; //[0,1] how full is the vehicle
    float       m_actVeloc; //actual speed
    float       m_totalTime; //total time of utilization
    float       m_startTime; //initial time (of EndLoad())
    float       m_totalKm; //total of kilometres
    float       m_startKm; //starting kms
```

```
    int             m_actPos; //actual position in the path
    Via*            m_actVia; //actual path
    float           m_startDeliveryTime; //start delivery time
    UINT            m_nextEvent; //the next event
    double          m_accTimeVolume; //time x volume(%)
    float           m_lastTimeVol; //time of last load/unload
    int             m_nTrips; //number of two-way trips
    float           m_lastStart; //last start time
    int             m_nodoIndex; //next node index
    CPtrList        m_nodoList; //list of nodes, way to clients
    veicData*       m_data; //vehicle data
    CPtrList        m_container; //container for the orders
    float           m_tLoadUnloadStart;//simulation (runtime)
    float           m_tLoadUnloadEnd; //idem
    void*           m_csu; //CSU owning the vehicle
    float           m_cashToAsk; //money to ask (xtunnel)

//Methods.....................................................................
    Veiculo(veicData* newdata); //public constructor
//Methods related with the class Area
    virtual void    CopyTo(Veiculo* pVeiculo);
    virtual void    MoveTo(View* pView, FPoint ptf);
    void            Show(); //Also draws the vehicle
//Other methods
    void  Modelo(UINT modelo); //create the vehicle model
    int   CalcSentido(Tramo* tramo); //compute direction
    float CalcTimeToStop(); //compute time to stop
    float CalcTimeToEnd(); //compute time to next segment

protected:
    Veiculo();//construtor default

public: //Methods related with the simulation
    virtual BOOL Execute(UINT event, void* ppp=NULL);
    virtual void e_SimShow(void* ppp=NULL);

private: //event handlers:
    BOOL e_Start(void* ppp=NULL); //begin of a segment
    BOOL e_End(void* ppp=NULL); //end of a segment
};
```

The following code pertains to the event handler *e_Start()*:

```
BOOL Veiculo::e_Start(void* ppp)
{
    m_actLambda=0.0f; //begin of the segment
    m_estado=MOVING; //vehicle is moving
    m_time = m_pSim->Time(); //time=simulation time
    m_lastStart = m_time; //keep it

    //if vehicle is not yet in the list of entities in motion
    if(NULL==m_pSim->m_objMovingList.Find(this))
        m_pSim->m_objMovingList.AddTail(this); //add it

    //locate the next segment
    POSITION pos;
    m_nextTramo=NULL;
    if(m_actTramo!=m_tramoDestino)
        {
        pos = m_actPercurso->FindIndex(m_actPos+1);
        m_nextTramo=m_actPercurso->GetAt(pos);
        }

    //update speed
    m_actVeloc=m_velMedia; //original speed
    m_actVeloc /= m_actVia->m_Gk; //geometric Gk factor

    m_sentido=CalcSentido(m_actTramo); //direction

    //time to reach the end of the segment
    float dL=m_actTramo->m_comp;
    float nexTime=(1.0f - m_actLambda)*dL/(m_actVeloc);

    //Schedule the event VEIC_END
    m_pSim->Schedule(m_time+nexTime,VEIC_END,this);

    return TRUE;
}
```

And finally, the code of the event handler *e_end()*:

```
BOOL Veiculo::e_End(void* ppp)
{
    m_actLambda=1.0f; //end of the segment
    m_time = m_pSim->Time(); //update time

    //update distance travelled (accumulate)
    m_totalKm += m_actTramo->m_comp; //total Kms

    POSITION pos;
    //if not yet in the last segment.....
    if(m_actTramo!=m_tramoDestino) //more segments
          {
           //change to the next segment
          m_prevTramo=m_actTramo; //previous segment
          m_actPos++; //advance in the segment list
          pos = m_actPercurso->FindIndex(m_actPos);
          m_actTramo=m_actPercurso->GetAt(pos);

           //restart in the new segment
           this->e_Start(); //start moving in the new segment
          }

    //already in the last segment.....
    else{
           Nodo* pnodo = m_actVia->m_nodoOut; //node
           pnodo->m_veiculoList.AddTail(this); //add vehicle

           //take out vehicle from the moving list
           pos = m_pSim->m_objMovingList.Find(this);
           m_pSim->m_objMovingList.RemoveAt(pos);

           //Send an event ARRIVE to the new node
          float t=m_time;
           m_pSim->Schedule(t,NODO_ARRIVE, pnodo);
          }
    return TRUE;
}
```

4.11.4 Material-order (*Encomenda*)

Despite the fundamental role played in the Supply Chain by *material-orders*, these are not treated here as entities in the sense of what we have defined as the <*SimEntity*> object. Instead, they are modelled as ordinary objects with no events to handle. The *activities* related to them will be seen as belonging to the <*CSU*> entity. This results from the fact that in this approach *event handlers* are associated with objects playing the role of decision. For instance, the *vehicle* is the centre of decision concerning its movement, the *node* decides about its operations, etc. On the other hand, *material-orders* do not. These are merely static elements which vehicles transport from one place to the other, and, once in the CSU, are internally moved under the decisions of the facility.

Even so, since they are perhaps the most important elements in the simulation, we will present their structure implemented in the class <*Encomenda*> (order):

```
class Encomenda //material-order
{
public:
CSU*     m_csuOrigin; //CSU of origin
CSU*     m_csuDestiny; //CSU of destiny
CPtrList m_materialList; //List of materials
float    m_timeToBeCool; //cool time...
float    m_timeToRefuse; //time to refuse
float    m_timeOrdered; //time when ordered
int      m_totalPal; //total SKUs ordered
float    m_volume; //order volume (in containers)
CPtrList m_nodePath; //nodes till the CSU destiny
int      m_endProdQuant; //runtime (production)
void*    m_pplan; //pointer to a productPlan
void*    m_pveic; //pointer to the vehicle carrying
float    m_varCost; //final cost (set by supplier)
float    m_setupCost; //setup cost for ordering
BOOL     m_wasWaiting; //true, if not served from inventory
BOOL     m_okInspected; //true, if already inspected
```

```
Encomenda() //inline constructor
        {
        m_totalPal=0;
        m_varCost=0.0f;
        m_setupCost=0.0f;
        m_wasWaiting=FALSE;
        }

//try to serve an order from this one:
bool    TryToServe(float time, void* penc);
//remove a material line and update the order parameters:
bool    RemoveMaterial(void* mat);
//add a line of material and update the order parameters:
bool    AddMaterial(void* mat);
//delete order (encomenda):
void    Trash(void* csu);
};
```

As we can see, this class also uses an inline constructor for the initialization of the objects of its kind, a method of which the body is completely included in the class definition.

Profoundly related to the <*Encomenda*> is the <*Material*> object, which represents a *line of material*, the core of the *material-order*. The code of its class is the following:

```
class Material
{
 public:
    CString     m_prodName; //product name
    CString     m_prodRef; //product reference
    int         m_quantity; //how much, product units
    int         m_containerUnits; //units per container
    float       m_averageWait; //average wait
    float       m_prodCost; //set by the supplier
    int         m_stockState; //my stock state

    ProductBox* m_pbox; //my prodBox

Material()  //inline constructor
    {
    m_stockState = MAT_STOCKYES;
    m_prodCost = 0.0f;
    m_averageWait = 0.0f;
    m_quantity = 0;
    }
};
```

4.11.5 The product

The class <*Product*> implements the idea of *system product* described in the previous chapter. Although this class is prepared to be used as an *active entity*, handling the event *e_newGeneration()* with which new generations of the same product would cyclically be injected in the Supply Chain, in the present simulator version such features are disabled. In effect, the associated code was not implemented because the required work would probably not offer any compensation, since the issue is assumed not essential for the demonstration of the viability of the approach, the aim of this version. The <*Product*> will therefore be treated as an ordinary object. The code of its class is:

```
class Product : public SimEntity
{
public: //Attributes
CString   m_name; //name of the product in the supply chain
CString   m_ref; //reference of the product in te supply chain
UINT      m_type; //type of product  (R, P, P+)
COLORREF m_color; //color of the product
float     m_typLoadUnloadTimePal; //typical load/unload time
float     m_typLoadUnloadTimePalDp; //stdev
float     m_To; //time of creation of this generation
float     m_Vo; //initial value of this generation
float     m_VoDp; //stdev
float     m_dTLife; //this generation lifetime
```

```
float       m_dTLifeDp; //stdev
float       m_dTNext; //time for the next generation
float       m_dTNextDp; //stdev
int         m_containerUnits; //number of units in a container
CArray<int,int>     m_subprodQuant; //MRP sub-products tree

public: //Methods
                    Product(); //constructor
virtual void        Serialize(CArchive& ar);
virtual BOOL        Execute(UINT event); //events rooter
BOOL                e_newGeneration(); //event of new generation

DECLARE_SERIAL(Product)
};
```

It is interesting to notice that the idea of injecting new generations of the same product into the system would strongly contribute for revealing the importance of the *Reverse Logistics*, since obsolete products (with very low value) would be circulating in the Supply Chain, using not only the stock and the transport resources but also "depreciating" the demand by influencing negatively the next generation sells. By means of such a process, the Supply Chain would start to be simulated not just as an equation of materials flow in a single direction, but as a system of equations representing flows in opposite directions. The issue of *Reverse Logistics* is nowadays being treated with increasing interest (see Knemeyer et al., 2002; Ritchie et al., 2000; Tibben-Lembke, 2002) and perhaps the inclusion of this feature in this simulator will be part of a future work.

4.11.6 Product-box

As suggested in the previous chapter, the concept of *product-box* is used to represent a kind of "box" in which a specific product is held at the stock resource. Each *product-box* contains units of a specific product. So, it will also be associated with an *ordering policy*, for example. And, since the *stock* of the CSU is modelled as a *list of product-boxes*, each of which with its own inventory characteristics and own ordering policy, the processes running in the CSU will mainly interface with *product-boxes* instead of directly with *products*. In reality, most of the times one handles "boxes", and not units of product, in a stock facility.

The <*Product-box*> object includes data to characterize the stock, like maximum and initial inventory levels, *byLevel* or *byCycle* ordering policies, and whether or not these policies may follow the demand trend, etc. Therefore, it centralises much of the information needed by the *CSU-manager* for taking inventory related decisions. Any operation of input (output) of SKUs into (from) the stock, is the responsibility of the *product-box* object by means of its method *UpdateStock()*. And it is at the moment that such operations are executed that the stock level is also inspected. If appropriate, the *product-box* notifies the CSU of a need for material, the CSU will compute the quantity to order, and again the *product-box* will be responsible to handle the communication with its suppliers and place the order, by calling the method *AskMaterial()*. Notice however, that the *product-box* may also be configured in a way that this method is called cyclically, when modelling the figure of the *last customer* as a demand generator, for example. Such a process is controlled by the event PRODBOX_CYCLE which is handled by the event method *e_Cycle()*.

Another feature of this object is that instead of scheduling the events of ordering throughout the simulation process, all the orders can be scheduled to the appropriate

time and to the respective suppliers at the beginning of the simulation, based on a historical data record previously described in a text file. The method responsible for this is the *OrderByFile()*.

The ability to induce *demand-steps* into the normal generated demand is another interesting aspect included in the *product-box*. Although this is only authorized in *last customer* CSUs, it automatically generates a sudden step in the current demand randomly in time, thus enabling the later computation of the *Rigidity* of the Supply Chain. The amplitude of such a step will be defined by the user, as a multiplying factor for the current demand.

Some other services can be found in the <*Product-box*> class, but one also deserving reference is the data registered in the form of <*SPECTRUM*> objects, like the demand, the inventory, the production, the demand between supplies, the EOQ variation, and the supplier lead time variation. These can later be presented graphically to the user in an interactive window of the <*GraphDlg*> type. For all this, it is worth to inspect in more detail the entire code of the class <*Product-box*>:

```
class ProductBox : public SimEntity
{
public: //Attributes
    int     m_maxStock; //maximum stock (sku)
    int     m_initialStock; //initial stock (sku)
    int     m_actStock; //actual stock (sku)
    BOOL m_useHistory; //use historical data
    CString m_historyFile; //historical data file
    BOOL m_useByLevel; //order byLevel
    int     m_levelToOrder; //level to order (rop)
    int     m_rop; //level to order (rop) runtime
    BOOL m_useByCycle; //order byCycle
    float  m_daysCycle; //ordering cycle (days)
    float  m_daysCycleDp; //stdev
    float  m_daysCycleStart; //order cycle begin
    BOOL m_useKanban;//order by Kanban
    int     m_baseOrderQuantity; //base order quantity
    int     m_actOrderQuantity; //quantity to order (runtime)
    int     m_orderQuantityDp; //stdev
    BOOL m_useDifToInitialStock; //use diff. to initial stock
    BOOL m_useTrackDemandX; //adjust order level
    BOOL m_useTrackDemandY; //adjust quantity
    BOOL m_useEOQ; //adjust quantity to EOQ
    BOOL m_useImediatOrder; //order immediately
    BOOL m_useOrderManagement; //order later
    BOOL m_useDemandSteps; //generate demand steps
    float  m_supplierRespTime; //supplier lead time
    float  m_supplierRespTimeDp; //stdev
    float  m_actPrice; //actual price (to sell)
    float  m_actCost; //actual cost (at the supplier)
    float  m_coolTime;  //time delay for 100% satisfaction
    float  m_coolTimeDp; //stdev
    float  m_avrgSuppDelay; //computed lead time (runtime)
    float  m_avrgDemand; //computed average demand
    float  m_avrgTimeDemand; //time between 2 demands
    float  m_avrgEOQ; //computed average EOQ
    float  m_lastTimeDemand; //last time demanded
    float  m_lastTimeSupply; //last time supplied
    float  m_lastTimeProduce; //last time produced
    float  m_lastTimeStock; //last time handled in stock
    float  m_lastTimeAsked; //last time asked (!=ordered)
    float  m_accDemandSupply; //demand between supplies
    float  m_accDemandProduce; //demand between manuf.
    int     m_acc0; //acumulator 0
    int     m_acc1; //acumulator 1
    float  m_accHoldingCost; //accum. holding cost (to EOQ)
    float  m_accDemand; //accum. demand (to EOQ)
    float  m_oneOrderCost; //order setup cost
    float  m_oneOrderCostDp; //stdev
    int     m_qBase; //base step-demand (only last customer)
    float  m_mxStep; //multiplying factor for step-demand.

    SPECTRUM   run_stock; //stock graph
    SPECTRUM   run_production; //production graph
    SPECTRUM   run_demand; //demand graph
```

```
    SPECTRUM   run_suppdelay; //supplier lead time graph
    SPECTRUM   run_accdemand0; //accum. demand graph
    SPECTRUM   run_EOQ; //EOQ graph

    CSU* m_csu; //CSU owning me (runtime)
    CPtrList m_suppliers; //list of suppliers (CSU*) (runtime)
    CArray<int,int> m_suppliersIndex; //suppliers
    Product* m_product; //related system-product (runtime)
    ProductPlan* m_pplan; //associated product plan

    public: //Methods
    ProductBox(); //constructor
    void  CopyTo(ProductBox* pbox); //copier
    virtual void Serialize(CArchive& ar); //serializer
    int     UpdateStock(float time,int quant,void* ppp=NULL);
    void  OrderByFile(FILE* file, void* psup); //historical
    BOOL AskMaterial(float time, int q); //order the material
    Void  Clean(); //clean the product box

    //Related with simulation events
    virtual BOOL  Execute(UINT  event,  void*  ppp=NULL);
    BOOL e_Cycle(void* ppp=NULL); //ask cyclically
    BOOL e_askMaterial(void* ppp=NULL);//ask historically

    DECLARE_SERIAL(ProductBox)
};
```

Notice that the *product-box* may be linked to a *product-plan*, if the product can also be manufactured in the facility. This enables the modelling of CSUs where the product is produced internally even if at the same time it is ordered from an external supplier.

4.11.7 Product-plan

While the *product-box* is seen as a means of implementing the ordering of materials to external suppliers, the *product-plan* is instead focused on the internal orders for manufacturing the product. The two activities are by default concurrent, and if a product is to be only manufactured and never ordered, we, by convention, signal it by setting *m_baseOrderQuantity*=0 in the associated *product-box*.

The structure of this class is very similar to the previous one, as we can see in the next lines of code. Notice, however, that it is considerably smaller, as some information can be indirectly accessed by the respective *product-box* object. The parameters used are also typical of a manufacturing process:

```
class ProductPlan : public SimEntity
{
public: //Attributes
ProductBox* m_pbox; //connection to a product-box
BOOL    m_useHistory; //use historical data
CString  m_historyFile; //historical data file
BOOL    m_useByLevel; //produce based on inventory level
BOOL    m_useByCycle; //produce cyclically
BOOL    m_useDemand; //produce demand-driven (JIT)
int     m_levelToProduce; //producing level (rop)
float   m_lotsDayRate; //rate of production
float   m_lotsDayRateDp; //stdev
float   m_daysCycleStart; //produce cyclically start (day)
int     m_prodQuantity; //how much needed (runtime)
int     m_lotQuantity; //lot quantity
int     m_lotQuantityDp; //stdev
BOOL    m_useDifToInitialStock; //produce to refill stock
BOOL    m_useImediatProduction; //produce immediately
BOOL    m_useProductionManagement; //if upper-managed
float   m_productionTime; //time for producing one lot
float   m_productionTimeDp; //stdev
float   m_productionCost; //cost of producing one lot
float   m_productionCostDp; //stdev
Material*m_material; //material to be manufactured
CSU*    m_csu; //the CSU which owns me (runtime)

public: //Methods
ProductPlan();
void    CopyTo(ProductPlan* pplan);
void    OrderByFile(FILE* file, void* psup);
```

```
virtual void        Serialize(CArchive& ar);
BOOL    AskToProduce(float time, int q);
void        Clean();

//Methods related with events
virtual BOOL Execute(UINT event, void* ppp=NULL);
BOOL    e_Cycle(void* ppp); //cyclically manufacturing

DECLARE_SERIAL(ProductPlan)
};
```

4.11.8 The CSU entity

Finally, we arrived to the most complex and extensive[5] entity in the simulator. As indicated previously, the <*CSU*> class uses around 180 attributes of several kinds and 40 methods, 20 of which are event handlers. In the present context, the presentation of this code is inappropriate. In any case, as this is an extremely important class, we will present the code of its definition in the appendix 1. In this section, we will look at this object by focusing the attention on some particular aspects of its structure.

In the figure 4.17 the concept of a general <*CSU*> object is portrayed. This is a sort of *configurable* component with which, as we claim, diverse types of facilities can be modelled. Its central element is a *stock* resource, around which "life" runs. There is a process responsible for inputting product SKUs into the stock, and a process for outputting them from there. Vehicles, however, do not usually transport SKUs, but instead *material-orders* (collections of SKUs), therefore a section of *marshalling* is used to ensure both the break down of a *material-order* into its SKUs components or the opposite task. This is a general form of linking the stock with the vehicles loading and unloading at the output and input bays, the "material" interfaces with the external world. On the other hand, the information interface with the *clients* is considered immaterial, that is, clients place orders by means of some sort of information system, and the same is true with regard to the information interface with the *suppliers*. The orders of the clients, however, are received at a single point (*reception*), while the orders to the suppliers can be placed directly from each *product-box* after the agreement of the *CSU-manager*. This means that each supplier may, at least in theory, be treated in a different way. In the figure 4.17, dashed lines represent information, while solid lines are used for materials flow.

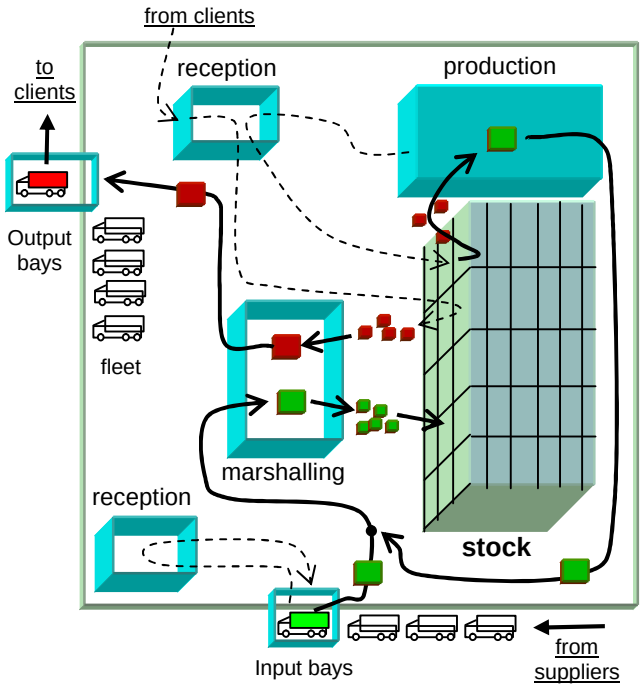

Fig. 4.17 The concept of a general CSU

As we can also see from this figure, the distribution of materials to the clients is

[5] Only the C++ code associated with the implementation of this object represents more than 130 pages of the present format! It cannot even be shown in appendix.

ensured by means of a *fleet* of dedicated vehicles, and the *production section* is the element of the <*CSU*> responsible for the manufacturing process. This element was coded in the form of a list of *product-plan* objects.

4.11.8.1 Configuring the facility

There are two variables in this class with which one can define the type of facility that the <*CSU*> object is to represent, from a *primer supplier* to a *last customer*. These are:

```
BOOL   m_useStock; //stock enabled
BOOL   m_useProduction; //production enabled
```

By setting each of these variables true or false the user may enable or disable the respective resources, and by that way transforming the <*CSU*> into a different type of facility. For instance, in a *warehouse* one may disable the production section, while in a more complex facility this section and the stock must be enabled, as in a *factory*, for example. *Primer suppliers* who do not use production may be implemented as normal warehouses (but with no paths to any suppliers), whilst the *last customer* has neither production section nor stock. The *last customer* has only some *product-boxes*[6] that cyclically order products. The code of the <*CSU*> class is then prepared to handle the situation by means of enabling or disabling the respective processes.

4.11.8.2 Icon

Any <*CSU*> object can be associated with an icon, to display an image of the facility in the simulator window. This is very useful not only for the visual representation of the Supply Chain but also because it allows a fast identification of the facilities in the simulator window. The attribute is:

```
HICON   m_icon; //icon to represent the CSU
```

Notice that this also helps the modeller to faster configure any particular facility, since each icon replies to the *right-mouse-button-click* with a menu for configuring and analysing the properties of the facility. The icons used are shown in the next figure (Fig. 4.18).

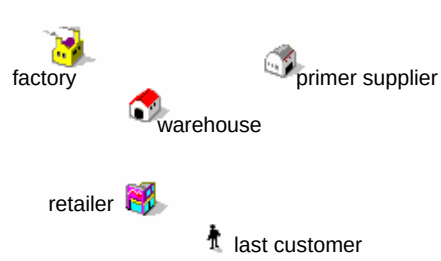

Fig. 4.18 Icons to represent different types of CSUs

4.11.8.3 States (activities)

As previously mentioned, the *states* (*activities*) of the CSU are modelled as *lists of objects*, in this case of pointers for objects (*CPtrList*). The objects themselves are created in the <*Document*> of the application and maintained in other lists which can be *serialized*. The following are the states considered in the present version of the <*CSU*> object:

```
//dead activities (queues).....................
CPtrList  m_inorderList; //orders in waiting list
CPtrList  m_waitOutputList; //orders waiting output
CPtrList  m_waitMarshList; //orders waiting marshalling
CPtrList  m_waitLoadList; //orders waiting to be loaded
```

[6] In reality, the last customer also uses a stock resource, but only to gain access to the utilization of product-boxes.

```
CPtrList m_waitMaterialList; //orders waiting to be supplied
CPtrList m_waitProductionList; //inorders waiting production
CPtrList m_waitProdList; //orders waiting production
CPtrList m_orderToLoadList; //orders to be now loaded
CPtrList m_freeVeicList; //free vehicle list
CPtrList m_workingVeicList; //working vehicle list
CPtrList m_inputBaysList; //input bays list
CPtrList m_outputBaysList; //output bays list
CPtrList m_veicWaitUnloadList; //vehicles waiting unload
CPtrList m_toOrderList; //orders to order with management

//live activities (live states).....
CPtrList m_onOutput; //orders in output
CPtrList m_onEnterOutput; //orders entering output
CPtrList m_onMarsh; //orders iin marshaling
CPtrList m_onLoad; //orders being loaded
CPtrList m_onEnterLoad; //orders entring the load
CPtrList m_onUnload; //orders being unloaded
CPtrList m_onEnterProduction; //orders entering manuf.
CPtrList m_onProduction; //orders being manufactured
```

4.11.8.4 Event handlers

As any other entity derived from the class <*SimEntity*>, the <*CSU*> object includes in its structure the event rooter *Execute()* and the *e_SimShow()* and *e_SimNewDay()* event handlers. But, apart from these, it also includes the handlers related to all the internal activity of the facility; that is, related with the processes of purchase, stocking, manufacturing, management, and the initial part of the delivery. These methods are defined in the next lines of code:

```
BOOL e_OrderNow(void* ppp=NULL); //order to supplier
BOOL e_OrderArrive(void* ppp=NULL); //order arrived
BOOL e_OrderIn(void* ppp=NULL); //order enters
BOOL e_MaterialIn(void* ppp=NULL); //materials arrive
BOOL e_EnterOutput(void* ppp=NULL); //enter output
BOOL e_StartOutput(void* ppp=NULL); //init to output
BOOL e_EndOutput(void* ppp=NULL); //end to output
BOOL e_StartMarsh(void* ppp=NULL); //init marshaling
BOOL e_EndMarsh(void* ppp=NULL); //end marshaling
BOOL e_StartLoad(void* ppp=NULL); //init Loading
BOOL e_EndLoad(void* ppp=NULL);//end Loading
BOOL e_StartUnload(void* ppp=NULL); //init unloading
BOOL e_EndUnload(void* ppp=NULL); //end unloading
BOOL e_VeicAtHome(void* ppp=NULL); //vehicle returned
BOOL e_EnterProduction(void* ppp=NULL); //enter manuf.
BOOL e_StartProduction(void* ppp=NULL); //start manuf,
BOOL e_EndProduction(void* ppp=NULL); //end manuf.
```

Notice that, contrary to what is usually practised, each event handler is able to receive a pointer transmitted through the simulation cycle. This is extremely useful, in certain cases. For example, to assign an unload operation in *this* CSU to a particular *vehicle* at a certain *time* one only has to use the code:

```
m_pSim->Schedule(time, CSU_START_UNLOAD, this, *vehicle);
```

Then, the event will pass by the *event-list* of the simulation, and later, at the execution of the handler *e_StartUnload()*, can again be made to point to the correct *vehicle*, as the next code exemplifies:

```
BOOL e_StartUnload(void* ppp)
{
Veiculo *pveic = (Veiculo*) ppp; //retrieve safe pointer
pveic->Draw(); //draw vehicle...
}
```

This method can also be interesting for implementing complex decision criteria for queue serving, for example, as it allows passing entities directly from a *live state* to another *live state*, ignoring the queue in between. So, only after being analysed, the entity would be sent to the appropriate position in the queue. These types of

criteria, however, are not used in this simulator version.

4.11.8.5 Other methods

Here we list some other methods of the <*CSU*> considered of interest in the present context, briefly commenting them:

int **CSUManager**(float time, UINT event, void* pent, void* ppp=NULL); //CSU-manager, responsible for all the important decisions of the CSU. Notice that it is called in the same form as the method Schedule(), with the information of an <*SimEvent*>.

void **ComputePathToDestiny**(Encomenda* penc); //creates a path to the order from this CSU to de CSU destiny. This path is linked (in terms of nodes) to the order that soon will be delivered to the client.

int **ComputeOrderQuant**(ProductBox* pbox, Material* material=NULL, float maxTime=99999.9f); //calculates the quantity to order to the supplier. This method analyses the current situation of the product inventory and, taking in account the type of ordering policy, establishes the quantity that the *product-box* must order to its supplier.

int **ComputeProductionQuant**(ProductPlan* pplan, int wantQuant=0); //calculates the quantity to produce, similar to what the previous method does.

UINT **FindPath**(Nodo* pnodo, Nodo* pnodofim); //find path between two nodes

Float **ComputePrice**(ProductBox* pbox); //gets the current price of a product (to sell) taking into account the chain of value in the Supply Chain.

bool **InspectEncomenda**(Encomenda* penc); //inspect the order from the client, verifying its consistence in terms of products and quantities, as well as correct destiny, etc. The order can be refused for some reasons, if the products do not make part of the stock, for instance, giving an error.

void **Stockout**(float time, Encomenda* penc, float qstockout, float valstockout); //give stockout of a certain order placed by a client. It updates the costs related to the situation, informs the client, reduces the client satisfaction, etc.

void **DoNewMeasures**(float time); //do some measures concerning the study of the rigidity concept (code not yet stabilized).

4.11.8.6 Main resources

The three main resources of the CSU are also defined as lists of objects. The fleet is created during run-time, therefore needing no *serialization* (uses the *CPtrList*), since the information about its vehicles is part of the CSU data. On the other hand, the stock and the production section are coded as lists allowing operations of *serialization* (*CTypedPtrList*):

```
CPtrList m_fleet; //vehicle fleet of the CSU (runtime)
CTypedPtrList<CObList, ProductBox*> m_stock;
CTypedPtrList<CObList, ProductPlan*> m_prodSection;
```

4.11.8.7 Suppliers

Also the suppliers are linked to the CSU at the beginning of the simulation by means of a method belonging to the <*View*> named *LigaSuppliers()*. Thus, suppliers are linked as pointers. On the contrary, *string* references to the effective suppliers are kept in a *data-array* which can be *serialized*:

```
CPtrList m_suppliersList; //all possible suppliers
CArray<CString, CString> m_suppliersYes; //yes suppliers
```

The performance of the suppliers is monitored during the simulation process using the following two variables, which are updated based on the *exponential*

smoothing moving average method (see Heizer & Render, 2001, pp. 86):

```
float m_avrgSupplyTime; //avrg lead time of the suppliers
float m_avrgTimeArrive; //avrg time between two arrivals
```

4.11.8.8 Fleet

The vehicle assignment and the attributes related to the *fleet* are coded in the CSU by means of several <*veicData*> objects (see appendix 1), where information regarding the operational parameters of the vehicle is defined and, in particular, the *echelons of transport cost* to the clients. Each type of vehicle assigned to the facility (and the number of it) is enabled or disabled from the following potential list:

```
//Type of vehicles and respective attributes of the fleet:
veicDatam_vans; //vans, number=?
veicDatam_smallTrucks; //small trucks, number=?
veicDatam_normalTrucks; //normal trucks, number =?
veicDatam_doubleTrucks; //double trucks, number=?
veicDatam_trains; //trains, number=?
veicDatam_boats; //ships, number=?
veicDatam_airplanes; //airplanes, number=?
veicDatam_xtunnel; //xtunnel, any CSU always has one!
```

4.11.8.9 Graphs

As we have seen, graphs are important tools that use data recorded in <*SPECTRUM*> objects. Similarly to the <*Product-box*>, the <*CSU*> uses this type of support to offer a visual representation of several variables and metrics related to the facility. As:

About inventory:

```
SPECTRUM    run_stock; //local inventory
SPECTRUM    run_stocktransit; //in transit stock
SPECTRUM    run_satisfaction; //satisfaction with supply
```

About costs:

```
SPECTRUM    run_deliveringcosts; //costs of delivery
SPECTRUM    run_stockingcosts; //stocking costs
SPECTRUM    run_producingcosts; //production costs
SPECTRUM    run_buyingcosts; //purchasing costs
SPECTRUM    run_holdingcosts; //holding costs
SPECTRUM    run_stockoutcosts; //stockout costs
```

About accumulated costs:

```
SPECTRUM    run_accDeliveringcosts;
SPECTRUM    run_accStockingcosts;
SPECTRUM    run_accProducingcosts;
SPECTRUM    run_accBuyingcosts;
SPECTRUM    run_accHoldingcosts;
SPECTRUM    run_accStockoutcosts;
```

About materials handling:

```
SPECTRUM    run_stockouts; //stockouts (me)
SPECTRUM    run_notserved; //orders not served (client)
SPECTRUM    run_demand; //demand
SPECTRUM    run_arrived; //arrived from suppliers
SPECTRUM    run_avrgDelivTime; //to the clients
SPECTRUM    run_avrgServingTime; //to the clients
SPECTRUM    run_avrgSupplyTime; //from suppliers
```

Notice that in each <*CSU*> the total inventory is registered (resulting from the summation of all the products used in the facility), but as well there is the possibility of accessing data specific from a particular product, by its *product-boxes*. Therefore, there are two levels for inspecting the CSU operation, the level of the “product” and the level of the “facility".

4.11.8.10 Final metrics

CSU metrics are computed at the end of each simulation run, and then also recorded into <*SPECTRUM*> objects. Some of this information is then accessible by means of a <*GraphDlg*> window, where it is shown in the form of a histogram (example in figure

4.19). For each simulation replication, a parallel line representing the value of the metric will be drawn in the appropriate category. So, after some replications, one has an affective visual representation not only of averages but also of the magnitudes of the respective deviations.

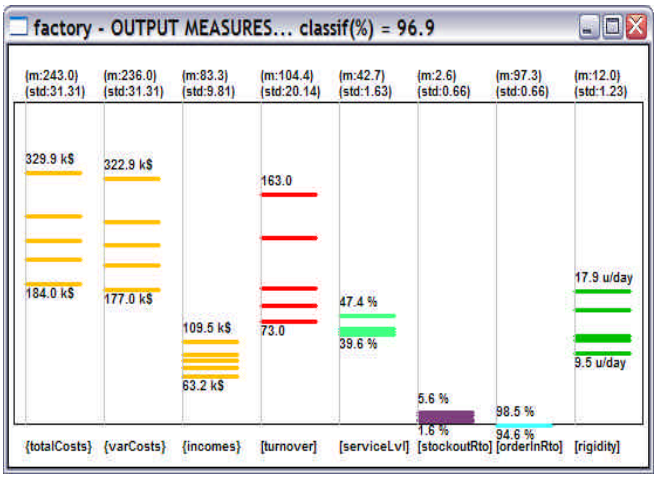

Fig. 4.19 Some metrics after 5 replications (an example)

These are the metrics computed in the present simulator version:

```
//final characterization
SPECTRUM      end_buyingcosts; //total purchase costs
SPECTRUM      end_holdingcosts; //total holding costs
SPECTRUM      end_stockingcosts; //total stocking costs
SPECTRUM      end_deliveringcosts; //total delivery costs
SPECTRUM      end_producingcosts; //total manuf. consts
SPECTRUM      end_stockoutcosts; //total stockout costs

//important facility metrics
SPECTRUM      end_totalcosts; //total costs
SPECTRUM      end_varcosts; //total variable costs
SPECTRUM      end_incomes; //total incomes
SPECTRUM      end_turnover; //turnover
SPECTRUM      end_servlevel; //service level
SPECTRUM      end_stockoutratio; //stockout ratio
SPECTRUM      end_orderinratio; //(1/dTarrive)
SPECTRUM      end_agility;//rigidity?
```

Fleet metrics and the rigidity matrix are also computed at the end of simulation, by the object <*View*>, but only displayed in a text window.

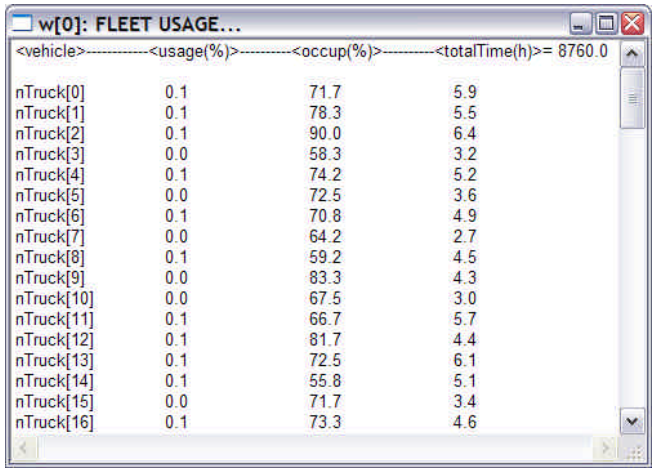

Fig. 4.20 Example of the flee metrics shown

4.12 The simulator application

We have reserved this part of the chapter for presenting the application for modelling and simulation that has resulted of the implementation of the approach described. This is a standard *Windows* application that can run in any ordinary computer, and it is comprised of a single executable file of around 2MByte of size. The majority of the results shown in the next chapter has been achieved by means of this application during the period of its development. Some small changes were meanwhile introduced to the program, but not of great importance and therefore deserving no special mention. We will make note of them when appropriate.

As was already observed, this section will be organized as a short tour around the application, and will be mainly based on the presentation of images and some comments.

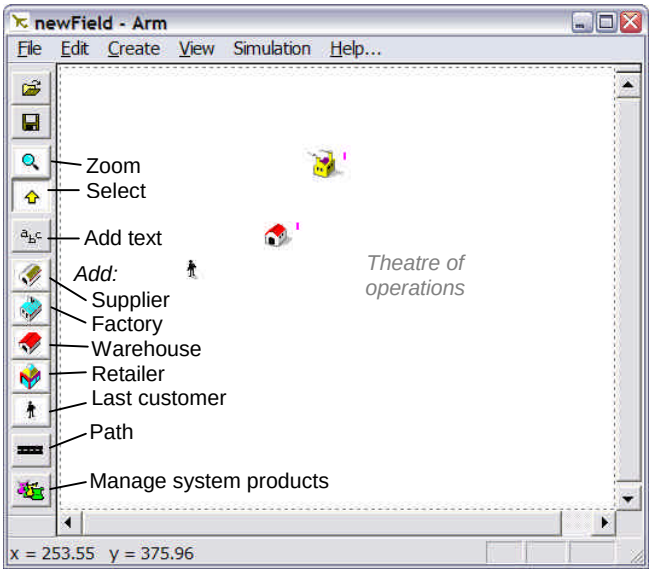

Fig. 4.21 This is a general view of the application, showing the theatre of operations and the drawing toolbar, with which the user can construct the Supply Chain model, adding facilities to the theatre of operations.

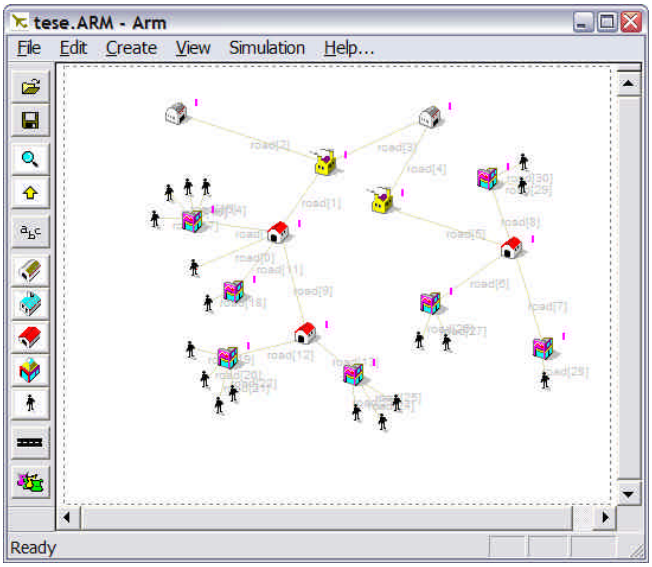

Fig. 4.22 In this figure, a model of a Supply Chain is already represented in the simulator window, with the facilities interconnected by means of transport paths. In this case, the Supply Chain has four levels, from the last customers (level 0), to the primer suppliers (level 4). Ultimate customers are used as generators of demand, so we do not consider them forming part of a level.

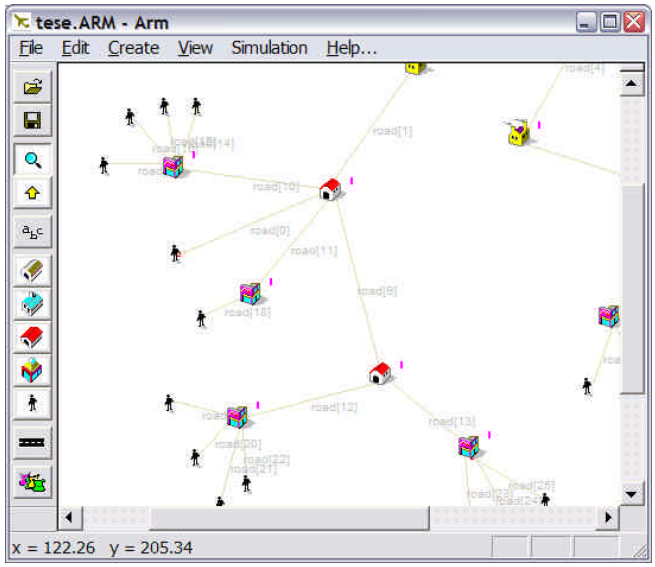

Fig. 4.23 This is a view of small zone of the Supply Chain obtained by the facility of Zoom included in the simulator. The Zoom can be used not only during the process of modelling but also during the simulation run. As previously noticed, this application also allows the user to access the property dialog-boxes of the elements of the Supply Chain even during the process of simulation.

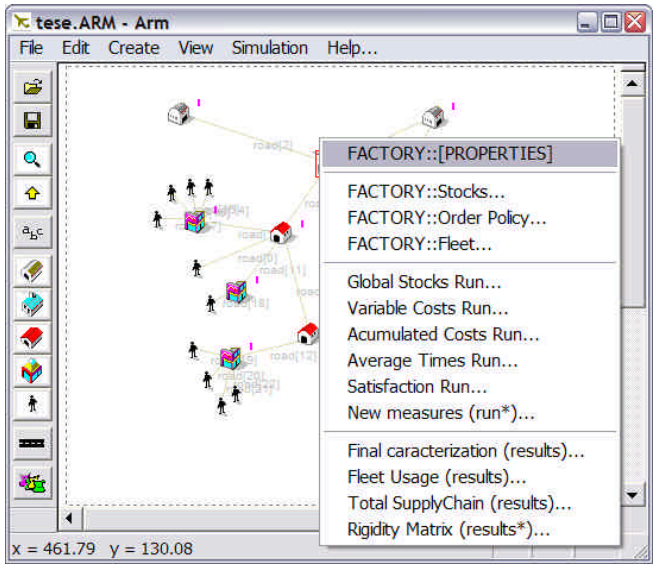

Fig. 4.24 This represents the menu linked to one of the factories represented in the model, which can be accessed by a simple mouse-right-button-click over its icon. From this menu, four main types of operations can be executed: access to the overall properties of the facility by means of its properties dialog-box; access to the most commonly used properties, like the configuration of the stocks, ordering policy and fleet; view the most relevant results presented in the form of graphs; view final metrics.

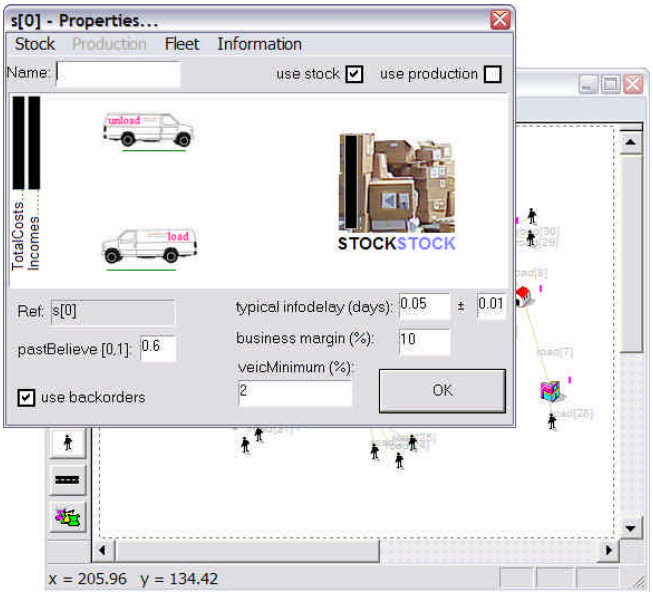

Fig. 4.25 Here is shown the main dialog-box of properties related to the CSU. It is in this dialog that the facility is to be configured in terms of the type of resources it uses (if only the stock, or also the production section). All the properties of the facility can be given access from this dialog and its menu.

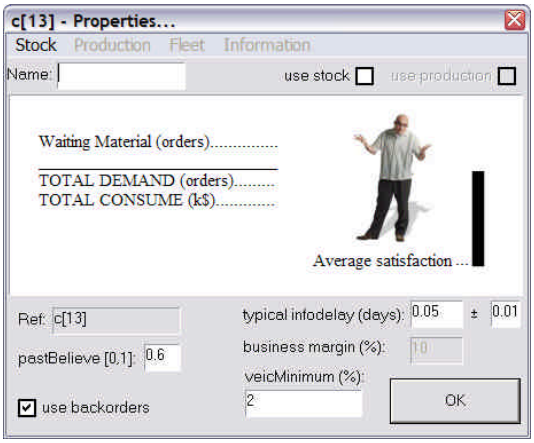

Fig. 4.26 This is the same dialog-box as before, but with the options of no-production and no-stock, that automatically transforms the CSU in the figure of an ultimate customer. In this case, most of the menu is disabled and only access to the product-boxes is allowed (through the stock).

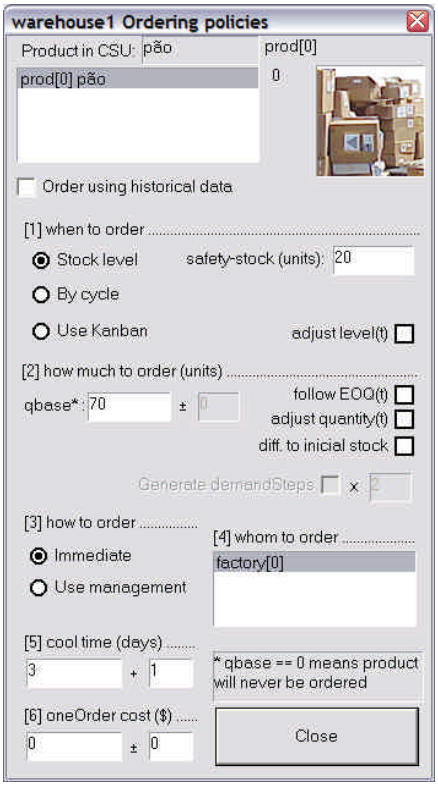

Fig. 4.27 This is the dialog-box linked to the stock properties related to the ordering policies associated with the product-boxes. A similar dialog-box is used for configuring the product-plans of the production section.

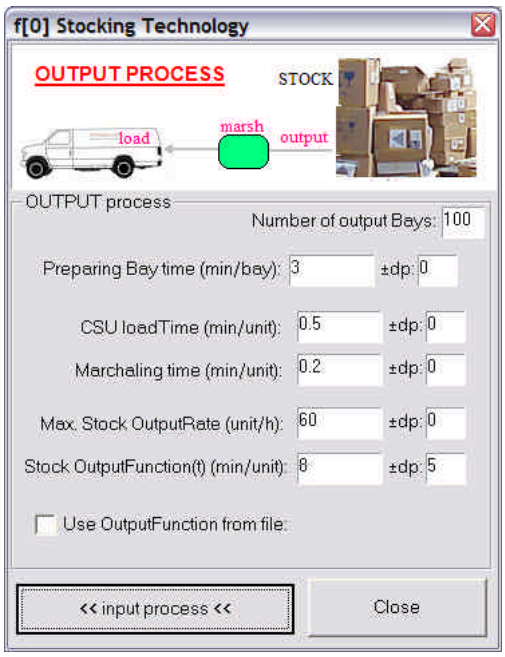

Fig. 4.28 The dialog-box for introducing the data for configuring the stocking technology, mainly times and rates.

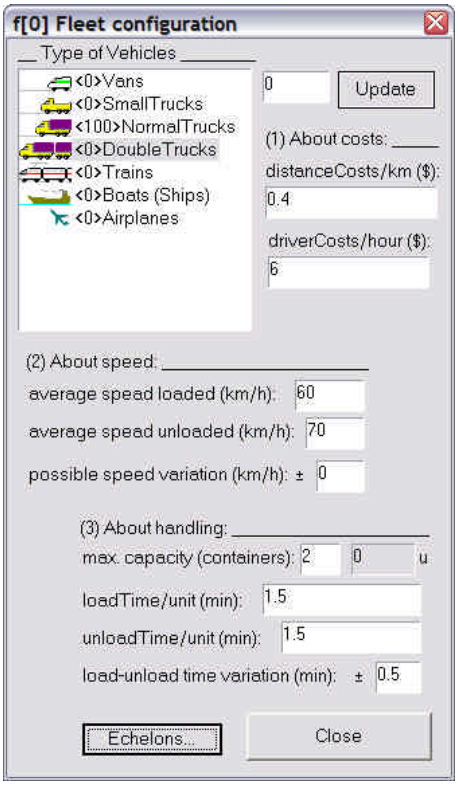

Fig. 4.29 This image shows the dialog-box used to configure the fleet of the CSU. The quantity of vehicles is set here and also the configuration of each type of vehicle, including the definition of the echelons of transport costs.

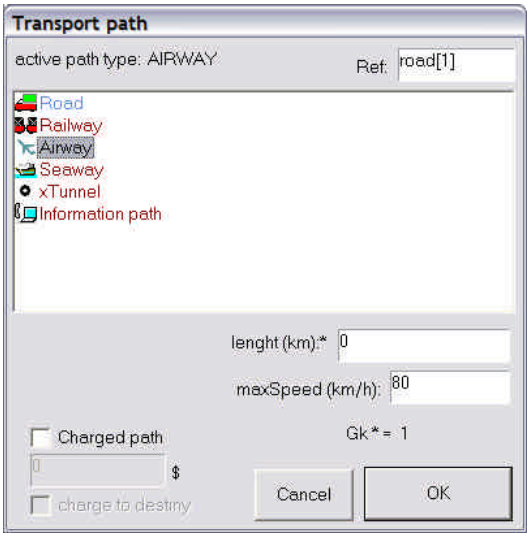

Fig. 4.30 This dialog is used for the configuration of the paths connecting the CSUs. Apart from the normal transport paths, this also gives the possibility of creating xtunnels and information paths. Notice the field where the length of the path can be set, and the constant display of the Gk factor.

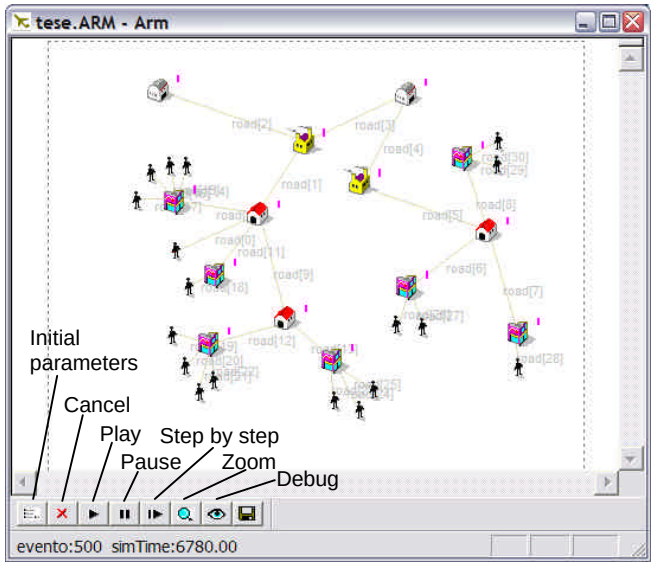

Fig. 4.31 This is what the application looks like when the process of simulation is started. The menu and the drawing toolbar are automatically substituted by a simulation toolbar, where the simulation progress can be controlled and supervised.

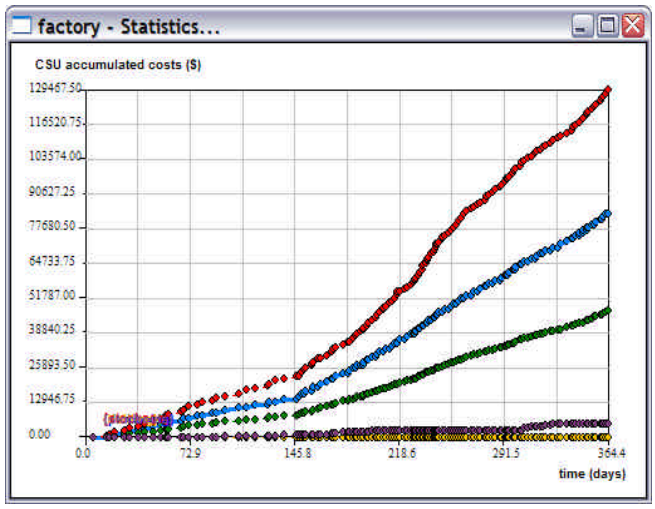

Fig. 4.32 This shows one of the output graphs generated during the simulation process, which in this case represents the behaviour of the accumulated costs of the various cost components, such as the purchasing, the stocking, the delivery, the holding and the stockout costs.

References:

Box, G. E. P., & Muller, M. E. (1958). A note on the generation of random normal deviates. *Annals Math. Stat, 29,* 610- 611.

Everett. (2001). *Generating Gaussian Random Numbers*. Taygeta Scientific Incorporated. 2004, from http://www.taygeta.com/random/gaussian.html

Heizer, J., & Render, B. (2001). *Operations Management*: Prentice Hall.

Knemeyer, A. M., Ponzurick, T. G., et al. (2002). A qualitative examination of factors affecting reverse logistics systems for end-of-life computers. *International Journal of Physical Distribution & Logistics Management, 32* (6), 455-479.

Microsoft. (2000). MSDN Library Visual Studio (Version 6.0): Microsoft.

Ritchie, L., Burnes, B., et al. (2000). The benefits of reverse logistics: the case of the Manchester Royal Infirmary Pharmacy. *Supply Chain Management: An International Journal, 5* (5), 226-233.

Tibben-Lembke, R. S. (2002). Life after death: reverse logistics and the product life cycle. *International Journal of Physical Distribution & Logistics Management, 32* (3), 223-244.

CHAPTER 5 177
CASE STUDIES AND COMMENTS 177
5.1 INTRODUCTION 177
5.2 VERIFICATION STRATEGY 177
5.3 ABOUT VALIDATING THE SIMULATOR 179
5.4 CASE 1: PROCUREMENT OF ADDITIVES IN THE OIL REFINERY OF PORTO, PORTUGAL 179
5.4.1 About the system 179
5.4.2 Building the model 180
5.4.3 Validation, corrective factors 183
5.4.4 The simulation strategy 185
5.4.5 Final results and comments 187
5.4.6 Conclusions 189
5.5 CASE 2: COMPARING STANDARD AND NAIVE ORDERING POLICIES IN AN INLINE SUPPLY CHAIN 190
5.5.1 Introduction 191
5.5.2 Overview on the KANBAN 191
5.5.3 The in-line Supply Chain 192
5.5.4 Simulating the case 193
5.5.5 Results with the KANBAN 194
5.5.6 Results with the naive BanKan 195
5.5.7 KANBAN versus BanKan 198
5.5.8 Conclusions 200
5.6 CASE 3: DISTRIBUTED SIMULATION GAME FOR SUPPLY CHAIN MANAGEMENT TRAINING 201
5.6.1 Introduction 201
5.6.2 The “Cranfield Blocks Game” 202
5.6.3 The client-server application 203
5.6.4 Results and comments 206
5.6.5 Conclusion 208
5.7 CASE 4: COMPUTING THE RIGIDITY MATRIX OF A DIDACTIC SUPPLY CHAIN 209
5.7.1 Inventory-rigidity-to-demand 209
5.7.2 Interpreting the matrix 210
5.7.3 Conclusions 211
5.8 LAST COMMENTS 212
REFERENCES: 213

Chapter 5

Case Studies and Comments

Verification, validation and discussion based on some simulation cases

5.1 Introduction

Several experiments have been done with the application for modelling and simulation described in the previous chapter. Although some of them were for didactic purposes or focused on promoting the interest of modelling and simulation, there were also cases aimed at evaluating the consistency of the approach, and even for studying a real industrial problem. It is interesting to mention that some ERASMUS students also used this tool (Polczynski et al., 2004) to develop a small study about the behaviour of the *rigidity* concept with the variation of certain parameters of the ordering policy (like ROP, *demand-step* and *order quantity*). The students used the ability included in the simulator for computing the *rigidity matrix* of a Supply Chain. Yet the most emblematic cases of applied simulation produced in the context of this work are the four studies described in the present chapter: the first is related to the analysis of the procurement of oil additives in the oil refinery of Porto, a real case study; the second is an interesting study where two different ordering policies are compared; the third is dedicated to the distributed simulator that was also developed for SCM training, including an overview of the application and some results obtained in a session with Master students; and finally, the fourth is about the calculation of the *rigidity matrix* of a didactic Supply Chain. Though, first of all we will dedicate some attention to the strategy used for the *verification* of the application for Supply Chain simulation, and also expose some considerations concerning its *validation*.

5.2 Verification strategy

The *verification* of the code implemented in the simulator was mainly ensured by monitoring the results obtained during the run of small models chosen for specific objectives, and comparing these results with those expected in theory. Following the implementation of a certain feature, like the ability for modelling vehicles of transfer, for example, a small model with examples centred on that issue was build and made to run, and the results analysed. In various situations some mistakes or discrepancies were detected, leading to the review of the code until the problem was solved. This procedure was systematically employed during the phase of development, pointing towards an improved confidence in the data generated in each simulation process.

The small model shown in figure 5.1, for example, was used for: (1) comparing the travel times of the various types of vehicles against the length of the respective paths, using the various types of paths; (2) testing the process of material transfer between vehicles; (3) verifying the consistency of the *output*, *input*, *load*, *unload*, *marshalling* and *production* activities; (4) testing the process of ordering by "Internet"; and (5) testing various simulation replications.

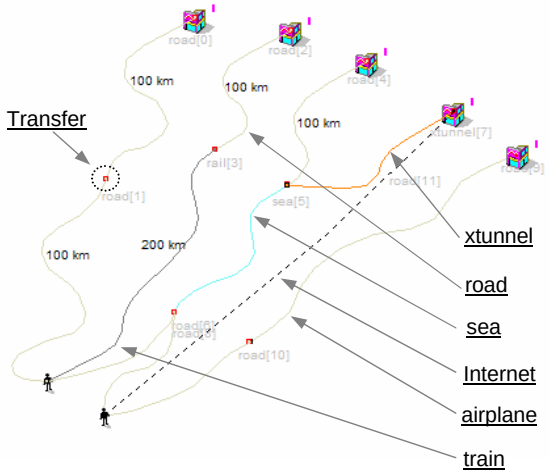

Fig. 5.1 Small model for testing some simulator features

Another example of one of these models is presented in the next figure (Fig. 5.2). This scheme was used for testing: (1) the usage of "*order management*" (a process in which the facility reorganises the material orders at the end of the day and places them to the appropriate suppliers); (2) the production of P+ products ($P[5] = P[3]+P[4]$) and ordering of the respective sub-products to the correct suppliers; (3) the attribution of path costs to the origin or to the destiny; (4) the computation and transfer of holding costs between facilities, including the *in-transit* inventory; (5) the correct calculation of costs in the manufacturing process; (6) diverse types of products ordered by the same customer.

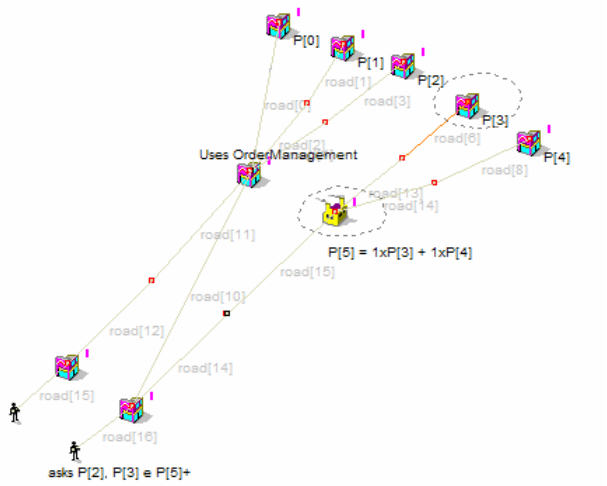

Fig. 5.2 Another small model used for verification

Several examples similar to these were simulated during the phase of more intense development, resulting in an accumulation of certainty concerning the fidelity of the code implemented. One of these latest examples was the model presented in the next figure (Fig. 5.3), with which the results obtained with different ordering policies were tested, each policy being implemented in a column of facilities.

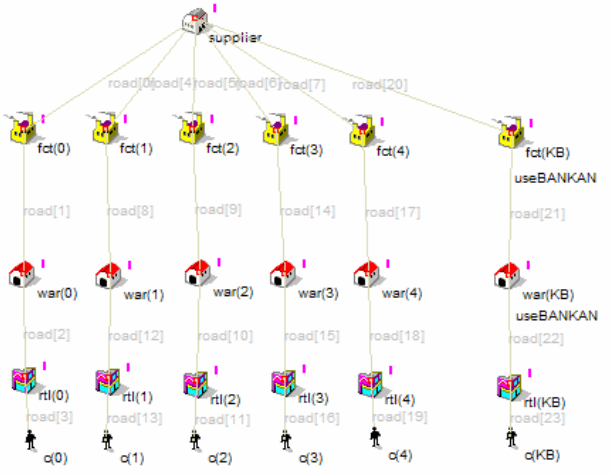

Fig. 5.3 Example for testing different ordering policies

Notice that the rightmost column was dedicated to a naive method of reordering (the *BanKan*) which will soon be discussed in the context of the second case study.

5.3 About validating the simulator

The *validation* of a model is frequently a complex task due to the fact that there is few real data available for comparing with the simulated data, and in some cases this is even the reason for choosing simulation to study the system. If enough real data would be available, probably the need for building a model to characterize the system would not make sense, in many real situations. In predictive simulations, however, like those for simulating the weather, large amounts of real data are useful to help "tuning" the model to the reality, but only for a certain moment, and in truth the validation of the final results will only be possible some time after the simulation. Often they diverge due to some unexpected factor.

Validating a time dependent model[1] is, in many cases, a difficult task, but validating an application for modelling and simulation makes the task several times harder. The results obtained with these applications rely firstly on the credibility of the application itself (or on the one who sells it), and then on the previous validations made in models simulated with the tool. A good strategy of *verification* helps to give confidence to the user, but certainties only come with practice. Probably, the application can even be more reliable for modelling certain types of systems than others.

Some believe that this task could be ensured by *accreditation* institutions, but, in contrast, presently few consider that those institutions would be reliable in the diverse and fast changing world of our times. For the moment, and for obvious reasons, the validation of the present flexible Supply Chain simulator will be reduced to the validation of a few models built (and simulated) with it. The most significant of these models was the one presented in the next section.

5.4 Case 1: procurement of additives in the oil refinery of Porto, Portugal

The data available herein results from a real case study of simulation about the procurement of additives for lubricants in the oil refinery of Porto (Portugal). Here we describe not only the case but also the method chosen to frame it by means of simulation, which finally led us to some interesting results regarding the purchase to the respective suppliers. Costs including transportation, holding and *stockouts* have been considered in the simulation process, as well as in the final conclusions, which in the end led us to recommend a more frequent reordering of materials at the lubricants plant. This was expected to result in the saving of around 0.3M€/year in global costs.

5.4.1 About the system

This refinery (Fig. 5.4) is located in the place of *Leça da Palmeira*, an important suburb of Porto, and has been operating since 1970. Even though it already had been

[1] Notice that if the model is static, or simply spatial, like the model of a house in architecture, for example, it can be made to increasingly approach the reality if its structure is refined step by step. The same is not valid for many dynamical models, as often we do not control certain variables in time.

used with a processing capacity of 7.5 MTon of crude per year in 1975, since 1982 its capacity has been maintained in the 4.4 MTon/year, mainly due to the installation of a second refinery in the south of the country (GalpEnergia, 2003). Even so, since the beginning, the refinery includes a Lubricants-Base plant, which presently works with a producing capacity of 150 KTon/year, and where diverse kinds of lubricants are produced for the industry and for the general market. This study is about the procurement related with this plant, a part of the whole refinery.

Fig. 5.4 View of the refinery of Porto, Portugal

The project has been conducted with the kind participation of the engineer Mr. José Carlos Fernandes, from *GalpEnergia*, the company responsible for the refinery, the person at the time in charge of the Department of Logistics and Stock Management of Lubricants. The main objective was to study in more detail the purchasing of around 200 different additives for lubricant production, ordered from 10 different European suppliers with different transportation costs based on the quantity ordered. Notice that the final lubricants normally include about 10% of additives. The Supply Chain under study was then a procurement network whose final node was this plant, as figure 5.4 shows, with suppliers spreading from England to Spain.

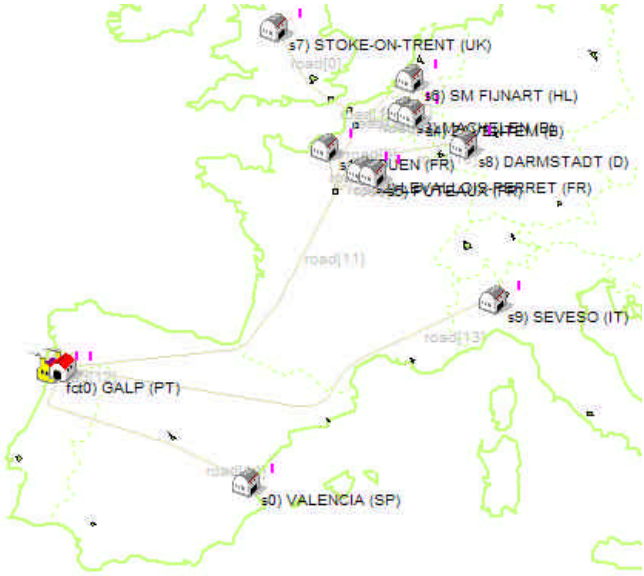

Fig. 5.4 The Supply Chain in study, with 10 suppliers

The price due to the transport of one unit of product from each supplier to the plant was dependent on the supplier and on the quantity ordered, so, the primary intention was to find the *quantity to order* that would minimize theses costs at the plant's level. However, later we have opted to consider not only the transportation costs but also the costs of products, products holding and products *stockouts*, since total costs were considered a better financial measure to support practical decisions. This case was published in Feliz-Teixeira & Brito (2005b).

This study was an important contribution to the validation of the Supply Chain Simulator previously described, as well as to let us draw conclusions on the methodology used and the sort of problems one may expect to face while modelling a real system of the kind.

5.4.2 Building the model

The first important aspect to have in mind while preparing the simulation of a

real system is to dominate the process of building the model, starting at the simulation tool. But, since the tool used here was specifically oriented to model Supply Chains, it was very easy to represent the elements of system. The creation of the model could be reduced to the following 5 procedures:

(1) The representation of the facilities in the "theatre of operations": the facilities have been located in the simulator window without any special geographic precision, as that aspect was irrelevant to the present study. Anyhow, each facility was placed carefully enough to maintain a relative fidelity to its real location, with the help of a map previously drawn in the simulator window (see figure 5.4). To assist in this, *GalpEnergia* has made available a document with the post addresses of the suppliers.

(2) The definition of the transport paths connecting the facilities: these paths have also been traced without great precision in the simulator window, as in this particular case the transport costs were already defined at each supplier by a list of prices per unit transported. So, distances were already included in these transport costs. Nevertheless, the transport paths traced in the simulator tried to follow the most probable lines of delivery used by the suppliers, and serve to give an idea of the delivering times as well as to introduce extra variability in the supplier's lead-times. *GalpEnergia* has also made available a document with the expected lead-time for each supplier.

(3) The definition and characterization of the products available to the system: this important step made us insert into the simulator, by means of a *dialog-box* like the one shown in figure 5.5, the first relevant information about the 195 different additives with which the plant usually works. Notice that the product unit (SKU) for all the products was considered the standard barrel of 200 Kg, the maximum number of SKUs per truck (of 24000Kg) was assumed to be 120.

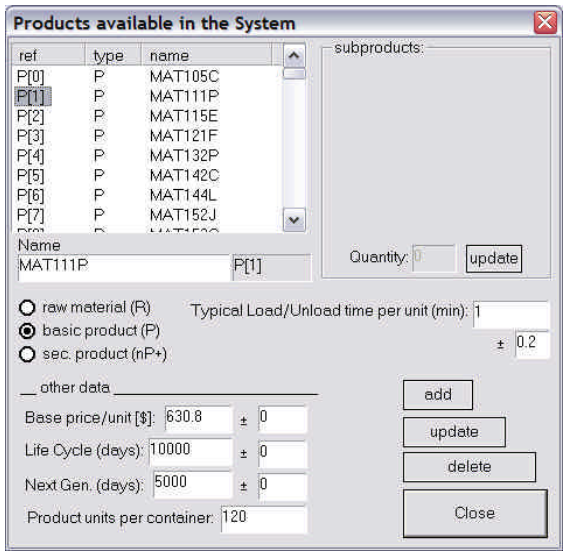

Fig. 5.5 Dialog-box to insert products into the simulator

Also included in the data of each product was its reference-name and its base price (at the supplier's level). This list was carefully inspected several times to ensure no mistakes were inserted in the simulation from the base. This data was primarily provided by *GalpEnergia* in the form of an *MS-Excel* document, which was directly retrieved from the plant's data base.

(4) The configuration of each facility, including the transport policy and the material's reorder model: once the system was geographically "represented" in the simulator window, and all the products

inserted, the general configuration of each facility had to be established, including the definition of its transport cost policy and the material's reorder model.

Firstly, suppliers have been configured to use cost echelons of transport. This data was provided by *GalpEnergia* in the form of costs per Kg based on different ranges of weight, but then have been transformed to express the costs in term of the percentage of occupation in a truck of 24000Kg. Each SKU was taken as a barrel of 200Kg. This data was assigned to the correspondent facility by means of a 5 echelon attribute filled in the *dialog-box* of figure 5.6.

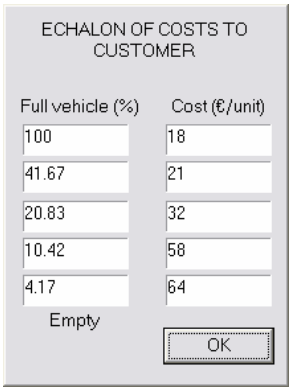

Fig. 5.6 Dialog-box to insert transport costs echelons

The final transport cost of a certain order could therefore be automatically computed as long as the percentage of its occupation in the truck was known. Higher than 100% occupation was made to incur 10% extra costs, as many real suppliers establish.

On the other hand, the definition of the material's reorder model, and other information related to this issue, was inserted at each facility by means of a *dialog-box* similar to the one shown in figure 5.7. Each product was chosen to use a (r, Q) reorder model, being defined the ROP, the quantity to order, and also the setup-cost, the *Time for Full Satisfaction* (TSF) and, finally, the respective supplier. This was done manually to the 195 products on the list, even if only 108 could be successfully linked to a valid supplier, and later simulated. The suppliers were considered supplied by perfect instantaneous delivering suppliers. All the facilities were made to operate with an 11.5%/year bank lending rate, and null business margin, meaning that the purchase cost was only affected by the basic product cost and the transport cost.

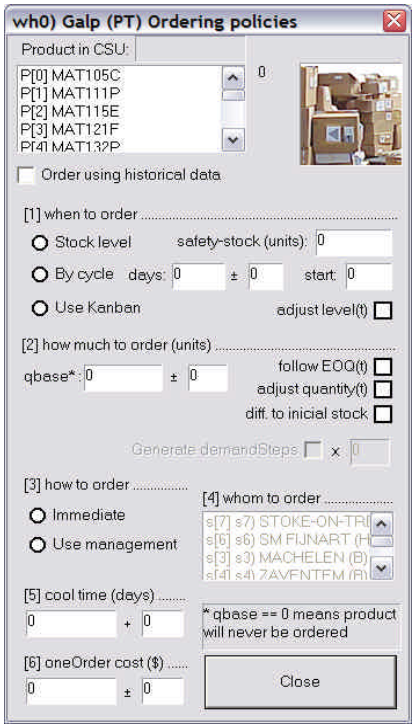

Fig. 5.7 Dialog-box for the reorder policy configuration

(5) The characterization of the demand at the plant: the demand information was obtained from documents of the real plant consumption during the year of 2003. This data was a monthly historical record. Then, based on such data, two main approaches have been chosen to simulate the case: (1) inserting the historical demand directly into the simulator; (2) using a *Normal* statistical

approach of this historical data as the demand to insert in the simulator.

Obviously, in the first case the demand had to be considered monthly, as in the reality, but, in the second case, by means of the statistical approach, we could also "hint" the consumption as if it would be sampled fortnightly or even weekly. This led us to compare results using different frequencies for sampling the real demand. Before inserting this information into the simulator, however, the 195 lines of original historical data, which were in Kg, were transformed into a monthly consumption of barrels of 200Kg. Of course, this implied a rearrangement of the monthly demand pattern, as the old demand had to be integrated in time until reaching the amount of a new barrel. This work has been done with the help of a little C program developed specifically to compute these demand transformations for all 195 independent lines of historical data. Finally, we could create the following 4 basic scenarios to be simulated: hist(30), which was driven by the original monthly demand; stat(30), driven by the *Normal* statistic version of the monthly demand; stat(15), considering the statistical demand sampled fortnightly; and stat(7), representing the statistical demand sampled weekly. These basic scenarios would later be compared by simulation, each one tested under the 5 different transport cost echelons previously considered.

5.4.3 Validation, corrective factors

The simulation process has been planned to help us in reaching two main objectives: first, the confirmation of the validity of the results obtained with the simulator, at least for the present case; second, the studying of different approaches to purchase the products, considering the 5 different transport cost echelons with which the suppliers operated. This section concerns the first process, that is, the model validation.

The process of validation was restricted by the fact that we only had access to documents with data about the *monthly plant's consumption*, *average cost of each product* and *actual inventory le*vel at the plant's warehouse. From this data we could calculate the *average plant consumption*, the *accumulated plant consumption*, the *accumulated costs of products* and the *average inventory level at the plant's warehouse*, regarding the activity in the year of 2003. These were the 4 realistic parameters with which we would later compare the simulation output, in order to validate the model. The following "real" values were retrieved from documentation based on the 108 products that definitely could be simulated (the others had no valid supplier assigned in the network under study):

- Average[2] inventory in 2003 (barrels) = 2484.8
- Average plant consumption in 2003 (barrels) = 477.7
- Accum. plant consumption in 2003 (barrels) = 5731.8
- Accum. costs of products in 2003 (K€) = 2835.3

With the purpose of comparing this data (R_j) with the simulator output data (S_j), each of the 4 basic scenarios previously mentioned was made to run for one year of simulated operations, considering the *consumption* equivalent to the *demand* (all orders satisfied; so, the *stockouts* must be

[2] Not really averaged, but instead estimated as the inventory level in June 2004.

added to the products purchased), and null any costs but the costs of the products (price at the suppliers plus a small value for holding). In this phase, the accumulated costs will simply represent the accumulated costs with the products, thus the real and the simulated data could be compared. For each scenario, the results of one year of simulation have been obtained in the form of graphs, as the examples concerning the scenario hist(30) shown in the next figures (Figs. 5.8 and 5.9). Using this procedure, table 5.1 could be built after simulating all the scenarios.

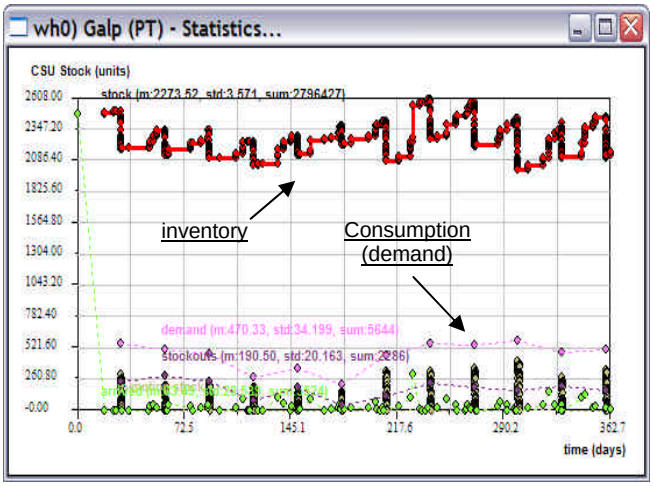

Fig. 5.8 First view of the inventory level and demand achieved by simulating the scenario hist(30) for one year of operations

Notice that, for simplicity, the unitary *stockout* cost was made equivalent to the unitary *product cost*, but this is not always a good approach, since the costs of a *stockout* can even interfere with the image of the company at the client's level. In this case, however, as the client is "internal", this approach was accepted as reasonable. Notice also that *holding costs* were in fact taken into account, as the real data about the *costs of the products* was already including this aspect.

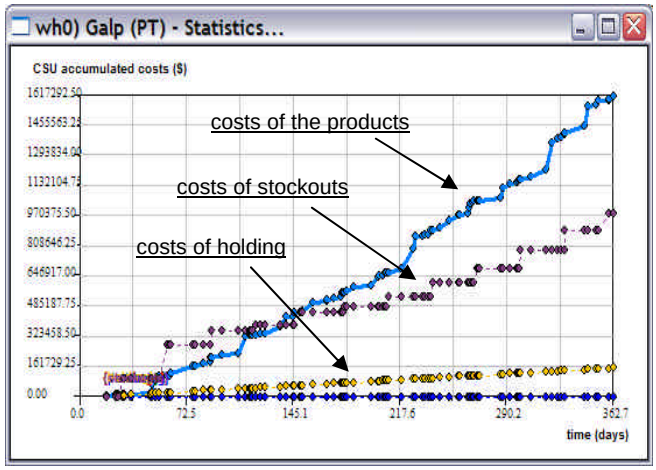

Fig. 5.9 Accumulated costs obtained by simulating the scenario hist(30), where purchase, stockout and holding costs are taken into account

	avrg inventory level (barrel)	avrg consume (barrel)	accm. consume (barrel)	accm. costs with products (K€)
hist(30)	2273.5	470.3	5644	2771.1
stat(30)	2069.9	495.2	6437	2984.9
stat(15)	2067.1	184.8	5358	2491.6
stat(7)	2096.1	70.1	4607	2089.8

Table 5.1 First results obtained by simulation, to be compared with real data

By comparing the data of this table (S_j) with the real data (R_j), another table was built (table 5.2) in which the percent deviations in relation to the real values could be estimated, by applying the formula:

$$dev\% = 100\ (R_j - S_j) / R_j \qquad (5.1)$$

Based on the results of table 5.2, we could already form some idea of what to expect about future deviations in each scenario. Actually, from this we could

expect a deviation from the real values of less than 3% in the hist(30) scenario (that already provides a reasonable validation for the model), as well as expect that the stat(30) scenario will inflate the results, while stat(15) and stat(7) will deflate them.

	avrg inventory level (%)	avrg consume (%)	accm. consume (%)	accm. costs of products (%)
hist(30)	-8.5	-1.5	-1.5	**-2.3**
stat(30)	-16.7	3.7	12.3	**5.3**
stat(15)	-16.8	-61.3*	-6.5	**-12.1**
stat(7)	-15.6	-85.3*	-19.6	**-26.3**

Table 5.2 Estimated deviations between simulated data and real data

Consequently, a corrective factor must be applied to each of these scenarios in order that in the end the results can be compared by means of the same scale of measures. Thinking of K_j as the factor by which each parameter must be multiplied to approach its "real" value, we simply have decided to calculate it from the expression:

$$K_j = Average(R_j / S_j) \qquad (5.2)$$

Where the *Average()* function was applied only to the *accumulated values*, since these were considered more reliable for validation than the simple values (they act like the summation of control in spreadsheets). By means of calculating this value for each scenario, the corrective factors presented in table 5.3 have been found (which have later been applied to the preliminary results):

hist(30)	stat(30)	stat(15)	stat(7)
K_1	K_2	K_3	K_4
1.02	**0.92**	**1.10**	**1.30**

Table 5.3 Corrective factors for the diverse scenarios

5.4.4 The simulation strategy

As the main objective of this study was the search for the most appropriate purchase policy based on the transport echelons established by suppliers, we decided to simulate each scenario according to seven predefined product occupation levels in a truck of 24000Kg. Each level was indirectly establishing the *minimal order quantity* to purchase from the supplier in a particular order, since each product quantity on the order was increased by 1 unit (SKU) till the entire order reached the minimum level's "mark". Notice that the same order can contain several products. These occupation levels were in fact forcing the *minimum truck occupation* to a certain predefined value. However, in the same simulation, the predefined level was maintained the same for all the suppliers in the network, instead of different levels for different suppliers. The following minimum truck *occupation levels* were tested in the study:

MinOccupLevels:
0%, 4%, 8%, 16%, 32%, 64% and 100%.

The procedure for simulating this case was the following: once the 0% level was chosen in the plant's warehouse, the scenario was made to run during 365 days along five replications before data was collected. After collecting this data, the

* Notice that increasing the demand sampling rate implies a decreasing in the average demand. These values have no special interest for the model validation.

new level was chosen and the same simulation process was repeated, while there was levels missing to be analysed (till the full truck level). After such a round, the scenario was considered simulated, and a new scenario was chosen to be analysed. Notice that in this phase the costs of transportation were already included in the purchase costs. Results like those shown in figure 5.10 were obtained for each scenario.

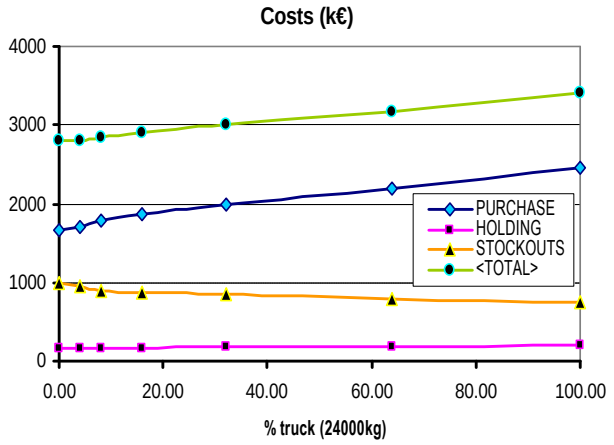

Fig. 5.10 Variation of costs with different truck minimum occupation levels, obtained with the scenario hist(30)

From this type of graph, the best occupation level was chosen as the one that minimizes the total costs, which in this case corresponds to 0% (meaning the best option is to purchase exactly what the inventory model requests without worrying about any transportation effects).

Along with these results, the average inventory level was also calculated and plotted graphically, as the example of figure 5.11 shows for the case of scenario hist(30).

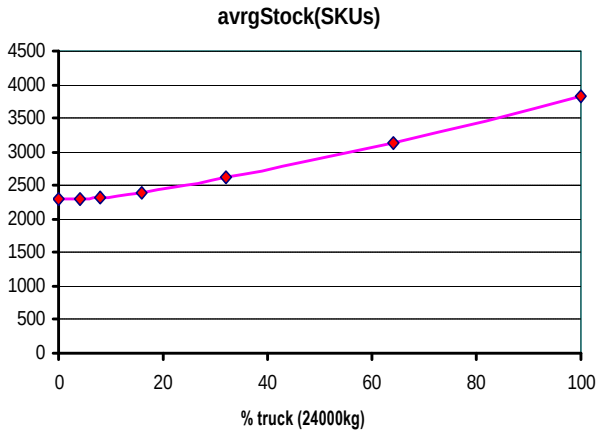

Fig. 5.11 Variation of the average inventory level at the plant's warehouse with different truck minimum occupation levels, obtained with the scenario hist(30)

Finally, the averages and variations of the plant's warehouse performance were also computed and presented in the form of a histogram, and in terms of some standard logistic metrics. In this particular case, these metrics were focused on the following measures: *variable costs*, *turnover*[3], *service level* and *stockout ratio*. As shown in the next figure (Fig. 5.12), this was a method of recording in a single graph the behaviour of each measure along the entire range of the purchasing policies. For example, from the same figure one can expect to observe in practice, along the seven different policies of purchase tested, an average value for the *service level* of 66.2% with a variation extending from 50.8% to 76.8%. On the other hand, the total *variable costs*, computed as the summation of *purchase*, *holding* and *stockout* costs, can be expected in the order of 3209 K€, with a variation between 2811 K€ and 4447 K€. Notice that in the present study the *fixed costs* were not included,

[3] The turnover was computed here like in a single product inventory, so, there will be quite a difference from the value expected in practice.

since they are not relevant to the objective in question.

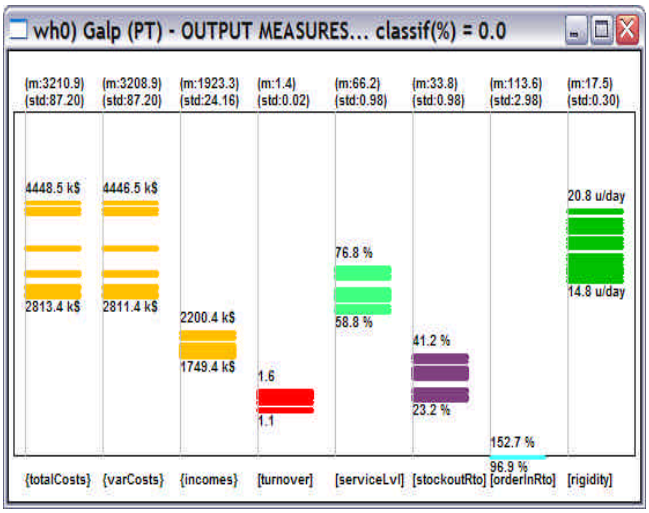

Fig. 5.12 Plant's warehouse performance obtained with scenario hist(30) along the entire range of purchase policies (total runs = 30)

The best results obtained with each scenario were then chosen, and this data organized per scenario and joined together in a new table. In the end, the respective corrective factors (K_j) were applied to this table and the final results obtained. These will be presented in the next section. Before that, however, it is important to observe that the following conditions were considered in the models:

- **BusinessMargin = 0** at all the facilities, meaning that the costs of the products were precisely the same as those provided by *GalpEnergia*.
- **No Backorders.**
- **PurchasingCost**=ProductCost+TransportCost.
- **BankLendingRate** = 11.5 %/year.
- **Transportation always available** at the suppliers, no need to wait for the arrival of a truck.
- **OrderSetupCosts** = 0 since they were in practice almost null in comparison with the products costs.
- **TimeForFullSatisfaction** (TFS) at the plant was considered 3 ± 0.5 days (indirectly rules the stockout ratio)
- **TimeForFullSatisfaction** (TFS) at the plant's warehouse was considered 30 ± 3 days.
- **Reorder model** (r=ro, q = Q(t)), with *ro* the actual safety stock and *Q(t)* to follow the maximum between average demand and the accumulated demand between supplies.

5.4.5 Final results and comments

The four basic scenarios were operating with the inventory model (r=ro, Q(t)), as previously noticed. Yet, to let us test this model with other values, the *average demand* during the 365 days of operations and the *accumulated demand between supplies* were computed, after the simulation of each scenario. Then, the original scenario was duplicated into a new one in which a change would be made in the inventory model: r_o was replaced by this *average demand*, while *Q* was changed to the *accumulated demand between supplies*. Then, the scenario was again simulated. Due to this practice, the following eight (8) scenarios have been in fact simulated:

1) **hist(30)** – original historical data, model (r=ro, Q(t)).
2) **hist(30)_ss_qq** – historical data, model (r=avrgDem, Q=accDemSuppl)
3) **stat(30)** – statistical data, model (r=ro, Q(t)).
4) **stat(30)_ss_qq** – statistical data, model (r=avrgDem, Q=accDemSuppl)
5) **stat(15)** – statistical data, model (r=ro, Q(t)).
6) **stat(15)_ss_qq** – statistical data, model (r=avrgDem, Q=accDemSuppl)
7) **stat(7)** – statistical data, model (r=ro, Q(t)).
8) **stat(7)_ss_qq** – statistical data, model (r=avrgDem, Q=accDemSuppl)

The graph presented in figure 5.13 shows the preliminary results obtained with the simulation of theses scenarios in the case of the best occupation level. These scenarios

have been numbered from 1 to 8, for simplicity of representation.

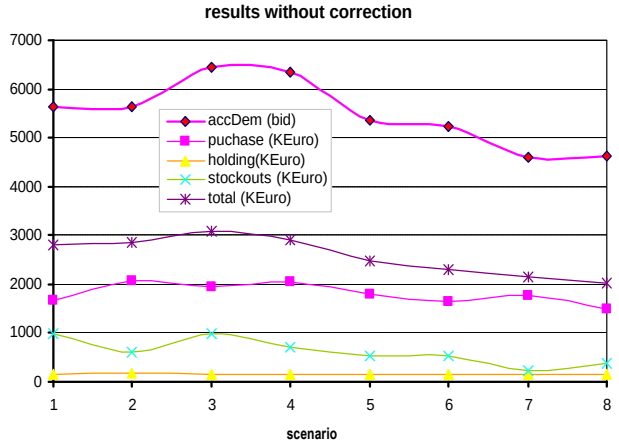

Fig. 5.13 Results before applying the corrective factors

From the data plotted in this figure, the behaviour predicted earlier is now evident. As expected, the scenario 3 (stat(30)) is observed to inflate the results, while scenario 5 (stat(15)) and scenario 7 (stat(7)) deflate them. Much of this is a consequence of the slight differences at the generated demand pattern that accumulate along the 365 days of operations, as also can be noticed from the figure. The upper line of data represents, in effect, this demand accumulated, and ideally should be a horizontal line. It is thus obvious that no conclusions could be retrieved from this graph before applying the corrective factors to it, that is, before transforming the upper line into a horizontal line. By doing so, we could finally obtain the costs figure shown in figure 5.14.

Notice that if this correction would not be applied we would have been misled in choosing the scenario 7 (stat(7)) as the most interesting option, in the preliminary results. In reality, however, the most interesting scenario seems to be the 6th (stat(15)_ss_qq), signalled in the graph, since it obviously led to the minimal total costs accumulated along the year.

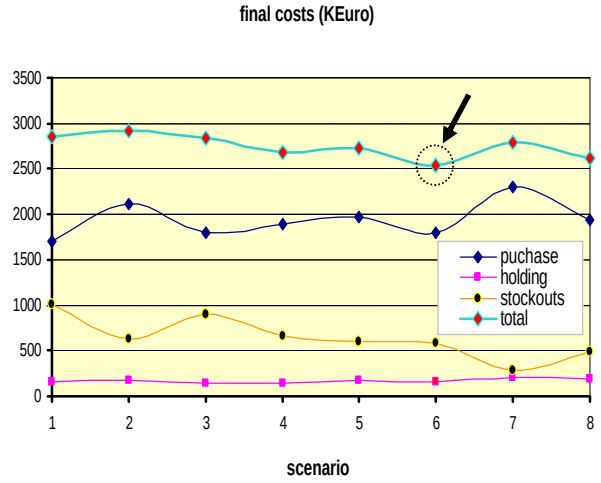

Fig. 5.14 Final results, considering the accumulated costs during one year of operations

Comparing this costs figure with the actual situation, this would mean a saving of around 300 K€ in a year, measured in the graph. And, as it was also observed that the best truck occupation level for this scenario was 8%, one can finally conclude:

"The best approach to operate the plant's warehouse would be to ensure a fortnightly inspection of its inventory and, once operating with the inventory model (ro, Qo), to set ro as the average demand (at each product) and Qo the respective expected accumulated demand between supplies, not forgetting to order always at least 8% of a truck. The total expected savings would be around 300 K€ in one year of operations".

IMPORTANT REMARK: One must not forget that these results have been obtained considering only 108 of the 195 products previously inserted in the simulator, the

ones that could be correctly connected to a valid supplier. So, these results probably do not represent a realistic view of the plant's behaviour.

5.4.6 Conclusions

Any conclusions taken from this study must take regard of this last remark. Only those products which could be linked to the list of suppliers considered in these models have been simulated. Consequently, the results presented here will obviously express this fact, meaning that the conditions of their validity do not reflect the overall plant's demand and, probably, they will not fit perfectly with those obtained in practice. In any case, they seem consistent and quite interesting, they confirm important values previously calculated from real data, and seem to point to a higher frequency in inventory revision with the objective of reducing total costs, a current trend in trying to minimize forecast inaccuracies. Actually, it is in general more difficult to forecast for a long period of time than for a shorter one, at least when the demand is "well behaved", as in this case.

Concerning the original objective of analyzing the costs variations with the different echelons of transport, it was observed that in all the scenarios except the stat(15)_ss_qq the best approach was the 0% minimal truck occupation, meaning it was to purchase exactly what the inventory model would request without worrying with any transportation effects. For that reason, this problem was naturally absorbed by the problem of sampling the demand with different rates, which during the simulation was becoming more relevant than the former issue. This was due to the fact that in most of the cases the original truck occupation was already naturally high, as figure 5.15 suggests for the case of the scenario hist(30), with an average value between 30 and 60 %. Lower levels of occupation were thus automatically "ignored" by the simulator, making this issue less relevant.

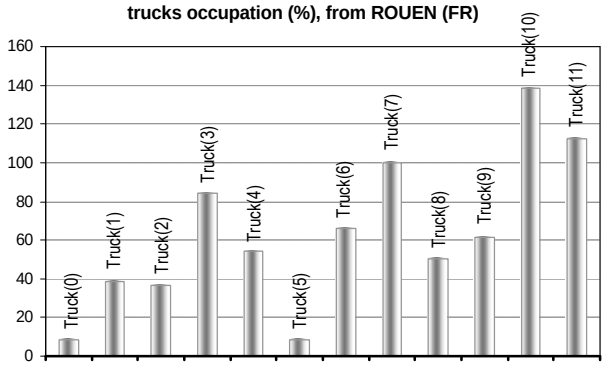

Fig. 5.15 Original truck occupation in hist(30) scenario

With respect to the methodology used in this case, it appeared to be firm and positive, leading to an interesting way not only of validating the model but also of making possible the comparison of different scenarios by means of applying corrective factors. In this process, the *verification* of certain aspects of the simulator could also be achieved, since some software "bugs" were detected and later repaired, mainly in the procedures for computing the holding and the purchasing costs.

The most important issue, however, in what concerns a more serious study of this plant, is related with the fact that not all the important products of the real demand (consumption) could be simulated, and that is expected to result in a higher deviation

between output data and real data. As shown in figure 5.16, it was observed that some of the products were extremely expensive and other relatively cheap, but not all the expensive ones could be simulated in this study. From this figure one may also get a qualitative idea of the probable deviation that could have been committed because of this problem.

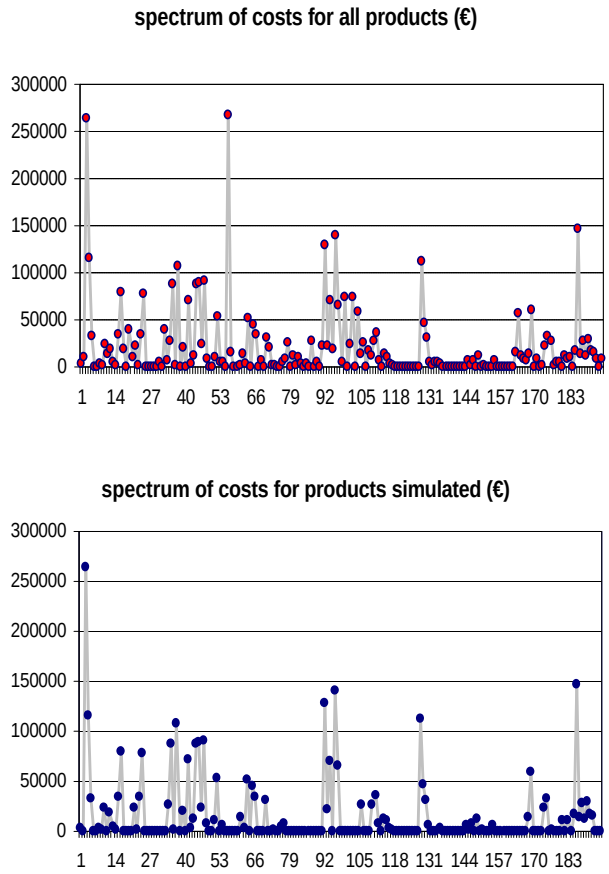

Fig. 5.16 Comparing the costs involving the 195 products used in the real plant (upper graph) and the costs related to the 108 products used in the simulation (bottom graph)

It seems now obvious that, for future studies, first we have to ensure that the most economically relevant products must be represented and simulated. So, in order to improve the results of this case, we would in the future need to focus our attention on the following main aspects:

- Ensure the simulation of all A type and B type products.
- Improve the statistical demand generator, maybe.
- Count with more costs: stocking, order setup, etc.

We hope that a more accurate study about this case can be completed in the future, following the careful analysis of these results that at present is being done by *GalpEnergia*.

5.5 Case 2: comparing standard and naive ordering policies in an inline supply chain

In this case, two different stock refill policies applied to the same in-lined one-product Supply Chain will be compared, by means of a dynamic simulation. The first policy is the Japanese method usually known as KANBAN[4] (similar to the "*Two Bins*" method, in terms of dynamics), and the second is a naive method named by the author as "*BanKan*" for reasons that will be explained later. The surprising results obtained in simulating these methods clearly point to several operational advantages of the naive method, at least under the conditions of such a Supply Chain configuration. The naive method does not make use of any criteria dependent on the actual stock level at the facility, and it was also observed that the materials flow through the Supply Chain as in a "river without dams", and with extremely low local inventory, as in the 1980s was defended by Taiichi Ohno, the father of

[4] In reality, the KANBAN is a signal to authorize the production of the next bin of material, and it is frequently associated with a card.

just-in-time (JIT) systems.

5.5.1 Introduction

This case is the result of a curiosity: the curiosity of confronting the sophisticated and the obvious. The idea was first appearing as an example for helping to test and verify the consistence of the Supply Chain Simulator previously described, but it was also inspired by the vision of materials flow presented by Mr Taiichi Ohno in his Toyota Production System (Ohno, 1988), which has resulted in the KANBAN scheme for implementing JIT. Contrary to the occidental believes of those times, which were strongly based on demand forecasts and have led to large *just-in-case* (JIC) systems, using a *push-into-stock* philosophy, the KANBAN method had changed these sophisticated forecasts with the simpler idea of producing the smallest possible quantities of materials triggered by the real demand, thus *pulling* the materials from the stock. Systems of this kind allowed the Japanese automotive industry to overcome its American counterpart already at the turn of the 1980s, for example, and since then have started to be applied in many other industrial areas all over the world, whenever possible. We have chosen this method as the standard method due to its simplicity and for its excellent performances. Nevertheless, the KANBAN system is still using *bins* of materials, and deciding the moment to ask for the next *bins* based on the quantity that rests in the stock after each stock operation. This method will be carefully explained in the next section. To the contrary, the naive method does not need any information about material levels in the *bins* to reorder the materials. In truth, it is completely independent of the inventory state.

It is also important to notice that the simplicity of KANBAN systems could help to dramatically reduce the inventories as well as improve the throughputs, and in fact the materials flow close to the flow of a river, but, as we will see, the naive method is even more effective in inventory reduction and, unlike the KANBAN, that flows like a river by impulses, the naive *BanKan* will simply flow continuously.

5.5.2 Overview on the KANBAN

Originally the KANBAN method was used to ensure the availability of materials at the input of a work-station in a manufacturing system, but later it begun to also be used as a material reordering policy between facilities in certain types of Supply Chains. Volkswagen, for example, is today applying JIT concepts in the supply network for the production of its new *Beetle* model, between Martorell, Spain, and Puebla, Mexico. But a better understanding of the KANBAN method can be achieved with the help of the next figure (Fig. 5.17), which simply represents a factory serving a retailer.

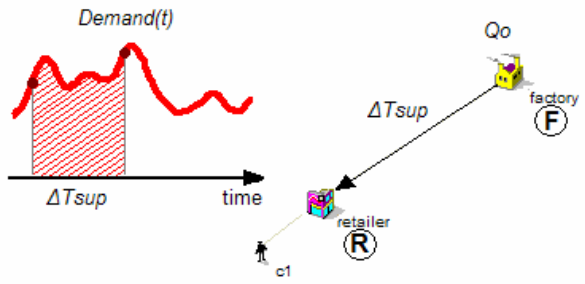

Fig. 5.17 Factory serving a retailer

Let us consider a factory (F) producing a previously optimized lot size (*Qo*) of a certain product, and needing a time ΔT_{sup} to

supply the retailer (R), in which there is a time dependent demand *Demand(t)*. Following the simple idea of the "*Two Bins*" method, the minimal quantity the retailer (R) must order from the factory to avoid *stockouts* is the total demand during ΔT_{sup}, that is, the total demand during the time the new material will need to arrive from the factory. As one can see from the figure, this quantity can simply be given by the integration:

$$\int_{to}^{to+\Delta T\,sup} Demand(t).dt \qquad (5.3)$$

As in most of the cases this quantity is dependent on both *to* and ΔT_{sup}, this integral is usually approached in practice by an estimation of its value calculated from historical data, known as the *Demand During Lead-Time* (DLT), plus a *safety-stock* quantity (SS) which is expected to protect the system against variations in the real *Demand(t)*. Thus, the minimal quantity to order from the factory will be:

$$DLT + SS \qquad (5.4)$$

Or, expressed in terms of factory lot sizes (*Qo*):

$$Ko = \frac{DLT + SS}{Qo} \qquad (5.5)$$

Ko is therefore the minimum number of lots to order from the factory, usually known as KANBAN, and can in practice be signalled by a card located at the appropriate position of the inventory, as figure 5.18 suggests. Each time the inventory level reaches the KANBAN, there must be a new order of material.

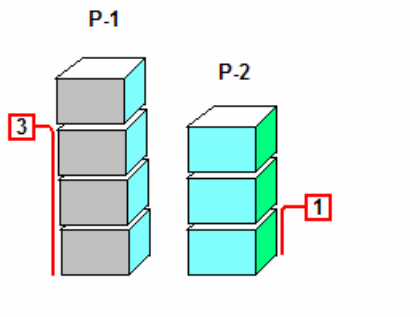

Fig. 5.18 Using KANBANs to signal the reorder points in a stock of two products

Notice that if the factory lot size (*Qo*) would match the *Demand During Lead-Time* (DLT), then *Ko* = 1, and the case would automatically transform into the widely known "*Two Bins*" method, which uses two *bins* of material and decides the moment to ask for the next *bin* based on the quantity that rests in the present *bin*. The full bin quantity is established to represent the DLT+SS, and a new bin of material is ordered whenever the present bin gets empty (or nearly empty).

5.5.3 The in-line Supply Chain

Imagine now that the factory and the retailer become simply too distant for the KANBAN quantity at the retailer to be maintained within acceptable values — for instance, in accordance with the *Economical Order Quantity* (EOQ) of Ford W. Harris (1913) — and thus there is the need to order the material to an intermediate warehouse. For academic purposes, let us consider this new warehouse located precisely at a *Lead-Time* distance of ΔT_{sup}, and that this facility will order the material to a second warehouse at a distance of $2\,\Delta T_{sup}$, and finally that this second warehouse will order the material to the factory at a distance of $3\,\Delta T_{sup}$, as depicted in the figure 5.19. Notice

that a single product is considered. This kind of lineal Supply Chains is commonly used for didactic or academic purposes, as is, for example, the case of the so called "*Beer Game*" developed at MIT during the 1960s (Sterman, 1989).

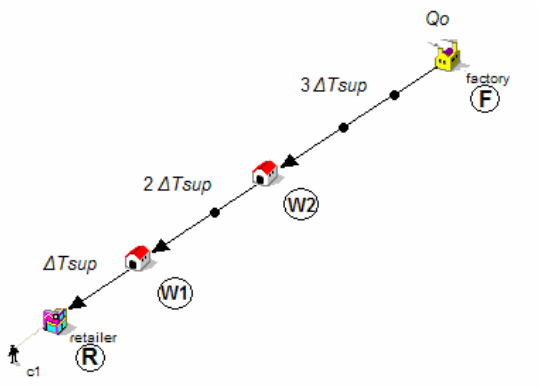

Fig. 5.19 In-line one-product Supply Chain for comparing the KANBAN and to BanKan

In our case, the objective is to compare the KANBAN method with the naive method that soon will be described. With this intent, and for simplicity, let us consider the lot size (*Qo*) matching DLT+SS on the retailer. Then, it emerges from the examination of the last figure that *Ko* = 1 for the retailer, meaning the retailer must order a *bin* = *Qo* to the first warehouse (W1) each time the actual bin gets empty (or nearly). Then, this warehouse will order to the second warehouse (W2) a quantity 2*Qo* each time its stock reaches the KANBAN *Ko* = 2, and this facility will finally order to the factory a quantity 3*Qo* whenever its inventory level reaches the KANBAN *Ko* = 3. Notice that these differences result from different DLT values observed along the Supply Chain.

5.5.4 Simulating the case

This Supply Chain has been modelled with our Supply Chain simulator, and the results have also been achieved with the same tool.

In the case under study, the customer's demand has been defined as *d* = 1 product units per day, with a standard deviation of 0.3 days. This let us choose *Qo* = 10, supposing 10 days of "distance" between the retailer and the warehouse W1. On the other hand, the *Time for Full Satisfaction* (TFS) of the customer was established in 0.3 days, which means that any order arriving from the retailer within this delay will be considered 100% satisfaction[5]. Orders, with longer delays, will contribute to the decline of the customer's satisfaction.

Concerning the retailer (R), whose reorder policy and configuration was kept the same in the two approaches, the initial stock was 12 units, the stock refill policy was the KANBAN (*Ko* = 1), and no-backorders were allowed. In all the other facilities backorders were allowed. These facilities used, in the first approach, the KANBAN method for reordering the material, as it was explicitly discussed in the last section, and only in the second approach they were configured to use the naive method. The retailer, as we have noticed, kept using the KANBAN in the two cases.

Finally, as it was assumed ΔT_{sup} = 10 days, the first warehouse (W1) will be 10 days distant from the retailer, as well as 20 days from the second warehouse (W2), and this 30 days distant from the factory (F). The factory is served by an ideal supplier at a distance of 40 days.

The final results were obtained applying statistics to the data acquired after 5

[5] We use this measure to retrieve a quantitative idea of the clients "satisfaction", and we apply this concept not only to last customers but also to any facility in the supply chain.

simulation replications of each model, with each run representing one year of operations. The measures of interest for each facility were the *inventory state*, the *accumulated variable costs*, the "*satisfaction*" as defined above, and finally some standard metrics usually applied to companies[6], like the *total costs*, *variable costs*, *incomes*, *turnover*, *service level*, and *stockout ratio*. The two refill policies have been compared based on these measures.

5.5.5 Results with the KANBAN

As previously noticed, in any case the retailer was operating with the KANBAN method, and for that reason some of the results related to this facility will be very similar in the two cases. An example is the inventory state, which is shown in the next figure (Fig. 5.20). As one can see, the inventory starts from the initial condition of 12 units and swings, as expected, between 0 and 12 units.

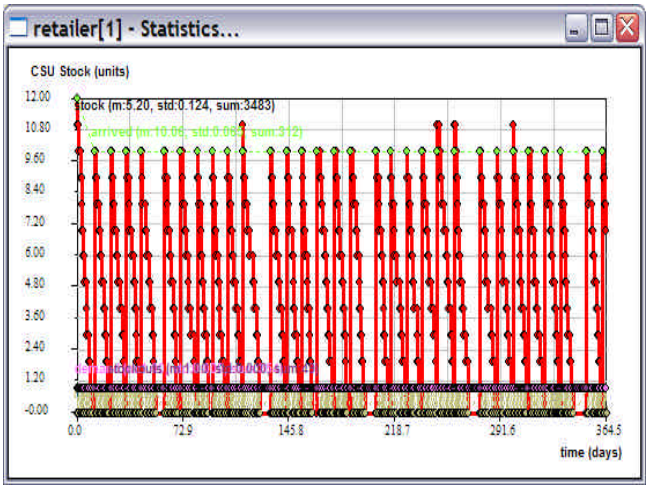

Fig. 5.20 Retailer inventory state during a year of simulated operations

[6] Even if some efforts are being made to establish Supply Chain metrics (Lambert & Pohlen, 2001), affective metrics that look at the network structure of a Supply Chain are not yet precisely defined. Most of the time people still apply the standard logistics measures.

Another similarity of the two systems is the customer "satisfaction", depicted in the figure 5.21, which oscillates between 50% and near 100%. Remember this holds for a *Time for Full Satisfaction* (TFS) of 0.3 days in the customer. Notice that there are also some orders "not served", which correspond to *stockouts* at the retailer.

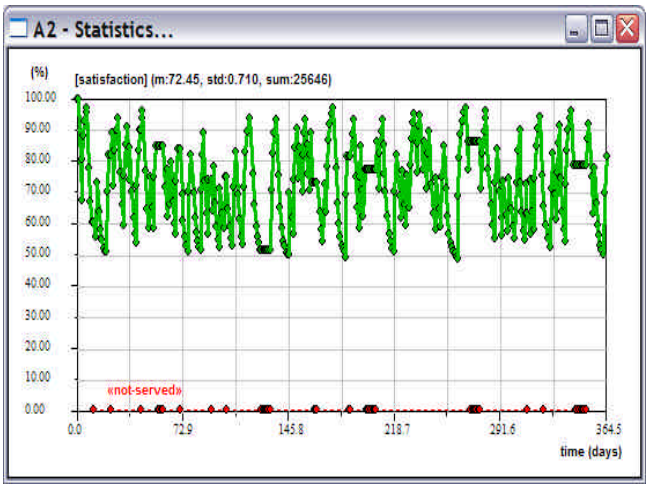

Fig. 5.21 Variation in the costumer's satisfaction and "not-served" orders

As we will see, the big differences will come into sight when comparing other measures, and more dramatically at the upper facilities of the Supply Chain. For the purpose of being compared with the next results, we show in figure 5.22 the inventory state of the second warehouse (W2). By inspecting this figure, it is easy to recognise that the KANBAN was *Ko* = 3, since it always orders 30 units of material. The inventory level swings between 0 – 40 units.

Note that it has also been observed that the first warehouse (W1) and the factory were also exhibiting this same kind of swing, but the first between 0 – 20 units and the latter between 0 – 60 units.

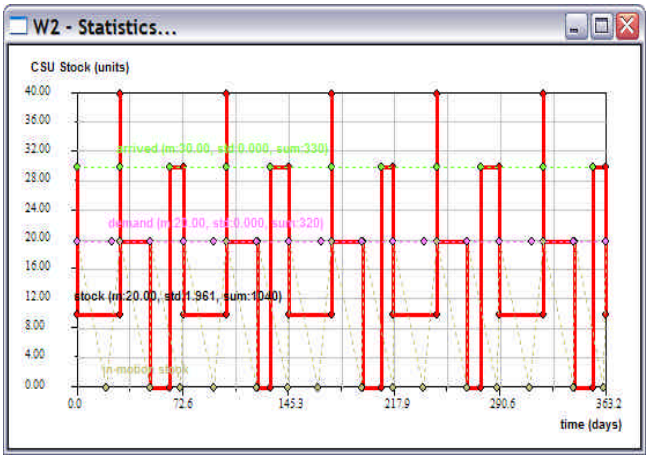

Fig. 5.22 Warehouse W2 inventory during simulation

Finally, in the next figure (Fig. 5.23) we represent the final characterization of the retailer (R) performance in terms of some standard logistic measures, considering also their variability along the 5 simulation runs.

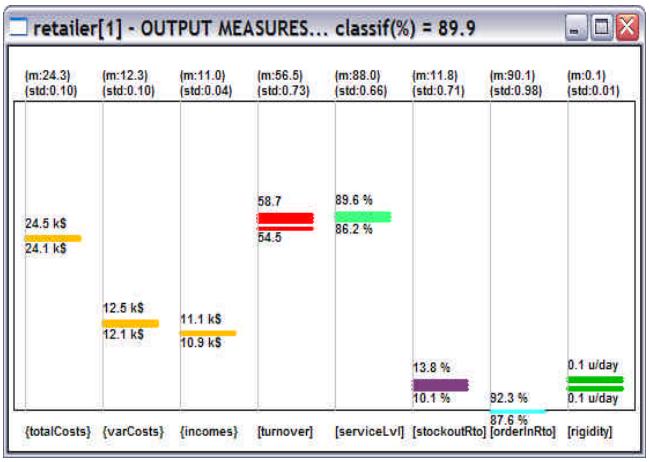

Fig. 5.23 Final characterization of the retailer

Notice that the *turnover* was around 56, the *service level* around 88% and the *stockout ratio* of the order of 12%. The last measure shown in the graph is not standard, it is still under study, but it is related with the "rigidity" of the facility, seen, as we know, as the inverse of "flexibility" (Feliz-Teixeira & Brito, 2004). These results seem plausible to be expected in a JIT system.

5.5.6 Results with the naive BanKan

It is now time to break the secrecy and talk of the naive *BanKan* method. It is always diffident exposing a naive solution, but we hope most of this strange impression will vanish with the presentation of the results achieved with such a scheme.

If we imagine a river, it is easy to understand that the material is moving through the influence of a "vacuum". That is, as long as a certain quantity of material leaves a certain space, the same amount of material is automatically moved to that space in order to fill this "vacuum". Obviously, this process does not have anything to do with the quantity of material in motion, but only with an action that is triggered by another action. If at a certain moment 10 litres of water are moving forward, 10 litres of water will automatically be moved to that same place in order to maintain the coherence of the fluid. If, in another situation, the quantity leaving is 23, for example, then the quantity that enters must be the same 23. There is, in fact, no need for any other reasoning or calculation. On the other hand, if the water stops in one place, all the other "places" will stop as long as they are filled with water. To better visualize this process in a Supply Chain it helps to consider the water having a very high viscosity, thus moving very slowly each time it has to fill a certain "emptied" space. Also, it can help to think of a chain of men unloading rice bags from a truck. Each man passes a bag to the next man as long as there is a "vacuum" in the next man and material enough to fill such a "vacuum".

From these considerations the naive policy comes out as saying: "*I will order whenever someone orders me, and exactly*

the same quantity".

Let us make a pause for smiling, but also to look once more at the Supply Chain at study, which is again represented in the next figure (Fig. 5.24).

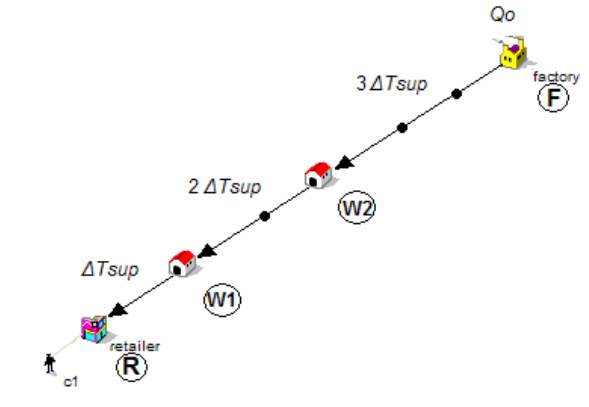

Fig. 5.24 In-line Supply Chain for comparing the KANBAN and the BanKan reordering methods

Taking into account that the retailer must always use the KANBAN, it will order *Qo* from W1. If now all the other facilities use the naive method, at that precise moment W1 orders *Qo* from W2, and this the same quantity from the factory, and the factory the same quantity from its supplier. At the same time, a quantity *Qo* is served to each client facility if there is stock available. As it is now easier to visualize, the quantity of material that will flow between any of the facilities in the Supply Chain is always *Qo*, the lot size produced at the factory which matches the *Demand During Lead-Time* (DLT) plus the *safety-stock* quantity (SS) at the retailer, as previously noticed. The result of this scheme is a continuous flow of materials from the factory to the retailer, precisely as in a river free of dams. Of course, to ensure that there will be no *stockouts* during the start and any restarts of this process one must also ensure a certain minimum inventory at the beginning at each facility. And this minimum inventory is precisely the previous value of the KANBAN at the facility. That is, the first warehouse (W1) must have an initial stock of 2*Qo*, the second warehouse (W2) an initial stock of 3*Qo*, and the factory (F) an initial stock of 4*Qo*.

Let us now present some results obtained with this method. The next figure (Fig. 5.25) represents again the inventory state at the retailer (R), and, as we have predicted, it is very similar to that of the last case. The initial stock is again 12 units, and during one year of simulated operation it was swinging between 0 – 12 units. It is observed, however, that this retailer had incurred in less *stockouts* than the previous one, as we will see soon.

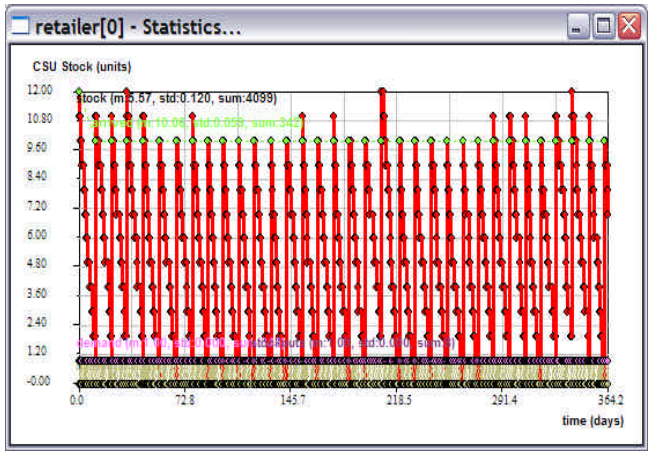

Fig. 5.25 Retailer inventory along one year (BanKan)

Depicted in figure 5.26 is again the customer "satisfaction", which once again oscillates between 50% and near 100%, as in the previous case. Nevertheless, it can also be observed from this figure that there were much less cases of material "not served" to the customer, which obviously

means less *stockouts* at the retailer.

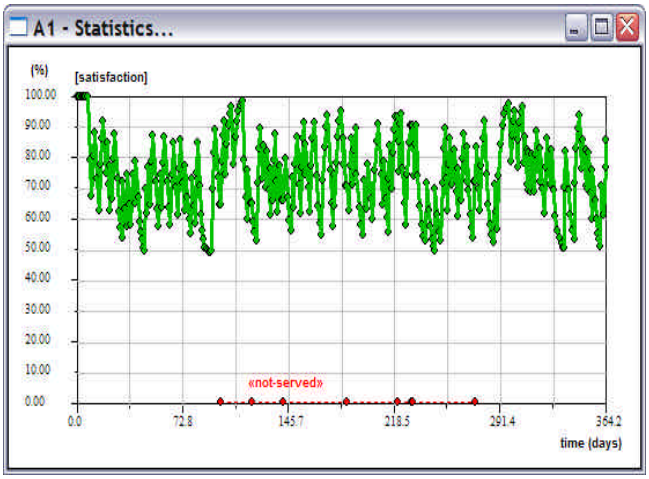

Fig. 5.26 Satisfaction of the costumer and orders "not-served" (BanKan)

Figure 5.27 again shows the inventory at the second warehouse (W2), which must be compared with the one previously shown in figure 5.22. The initial state is 3*Qo* = 30 units, but then the level quickly drops to the range 0 – 10 units. This is because the naive method tends to turn practically all the inventory into motion.

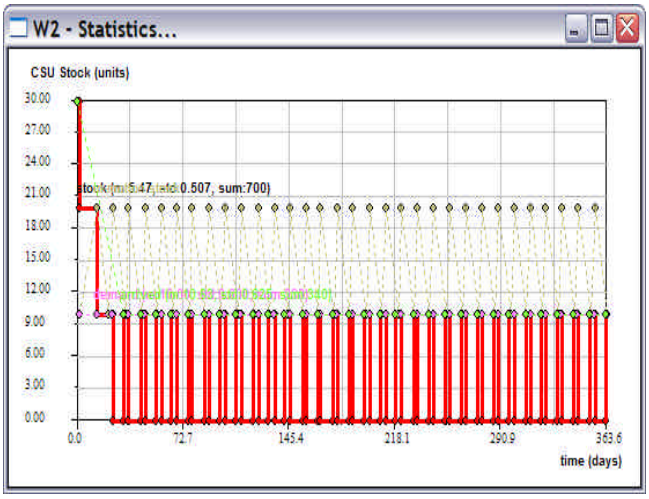

Fig. 5.27 Warehouse (W2) inventory during the period of simulation (BanKan)

This same behaviour have been observed at the other facilities (except the retailer), and the next figure (Fig. 5.28) shows the inventory state at the factory (F). Once again the initial stock of 40 units was fast descending to the range 0 – 10 units, which is precisely the range observed at any of the warehouses.

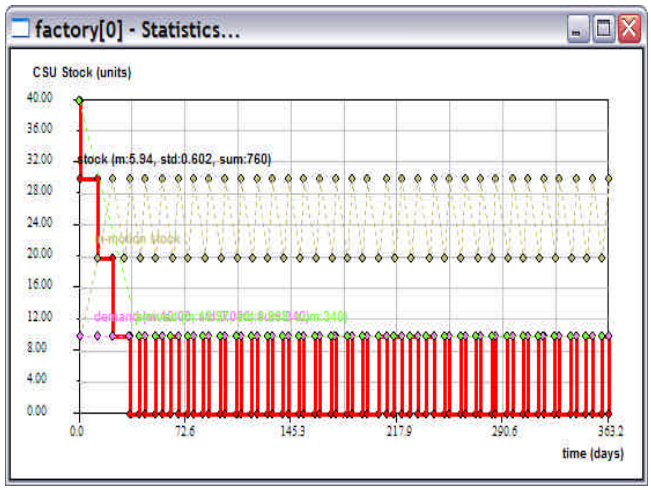

Fig. 5.28 Factory (F) inventory during the period of simulation (BanKan)

Notice that, if for some reason the demand would stop, the inventory in each facility would rise step by step with the last materials moving from the suppliers till the initial level. And there it would stay forever, since no more material would be ordered.

Finally, in the next figure (Fig. 5.29) the final characterization of the performance of the retailer (R) is presented, in terms of the same standard logistic measures. The variability along the 5 simulation runs is also shown, and it is interesting to notice that this variability is substantially lower than before. For the same variability in the demand, the *BanKan* system exhibits a more stable response than the KANBAN.

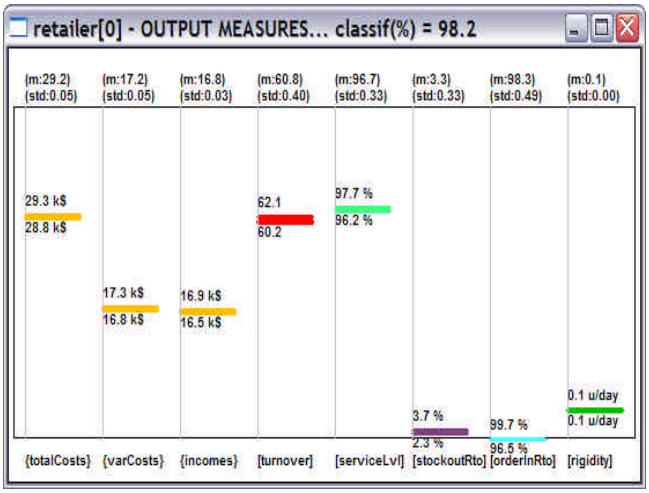

Fig. 5.29 Final characterization of the retailer (BanKan)

But other conclusions can be retrieved on comparing these values with the ones previously shown in figure 5.23. One may, for instance, conclude about the superior general performance achieved with the *BanKan* method, since not only the ratio between *incomes* and *costs* is higher, but also the *turnover* and the *service level*, and the *stockout ratio* clearly lower. About this last measure, it is interesting to notice that it was reduced from 12%, with the KANBAN, to 3%, when using the *BanKan*. This obviously means that the costumer was much better served by the *BanKan*. A more attentive inspection of these results would lead us to conclude that the naive method appears superior in almost every aspect, and that is curious.

We may now explain the reasons to choose the name *BanKan*. Firstly, to keep the Japanese phonetics as a compliment to the sense of perfection with which Japanese have conquered the world's sympathy, as well as made possible the popularisation of technologies during the 1990s, especially informatics and digital, which before seemed to be given to the public in small doses by monopolist enterprises. The second reason is that the *BanKan* is somehow like a jocular inverse of the KANBAN, once the stock "operator" is much more interested in dispatching the material from its stock than in calculating when or how much to reorder. The KANBAN was thought to make the stock available for a certain operation in a production system, while the *BanKan* is thought to make the stock not available at all in the facility, while keeping the system running. We must only keep in hand the *hot-potatoes* that the clients need. Finally, the *BanKan* is also a tribute to Taiichi Ohno and, at the same time, an expression of the inverse reasoning of the standard methods for stock control. In one sentence, the other methods aim to control, the *BanKan* dispatches.

5.5.7 KANBAN versus BanKan

To compare these two policies with some more accuracy, the simulation results were organized by facility and presented graphically. In figure 5.30, for example, is shown the *average inventory* held at each facility during one year of operations.

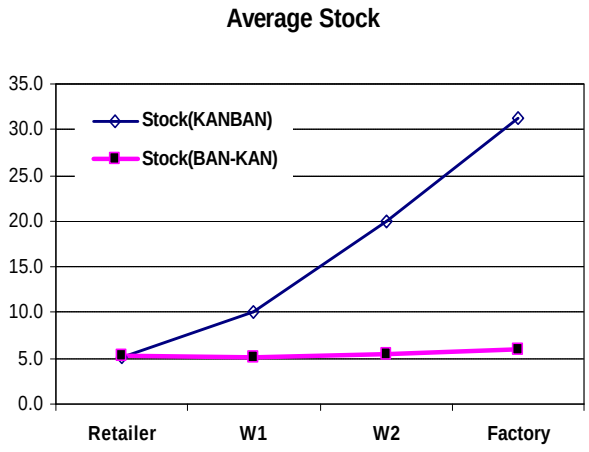

Fig. 5.30 Comparing the average inventory along the Supply Chain

From this graph one can obviously conclude that the inventory significantly grows from the retailer to the factory while using the KANBAN, but stands constant and minimal when using the *BanKan*, since this method tends to set the stock in motion.

Concerning the *turnover* and the *service level*, both represented in figure 5.31, the results show that the *BanKan* solution is obviously preferable at any level of the Supply Chain. With the KANBAN, these measures show a tendency to decline on approaching the factory, while they keep practically constant with the *BanKan*.

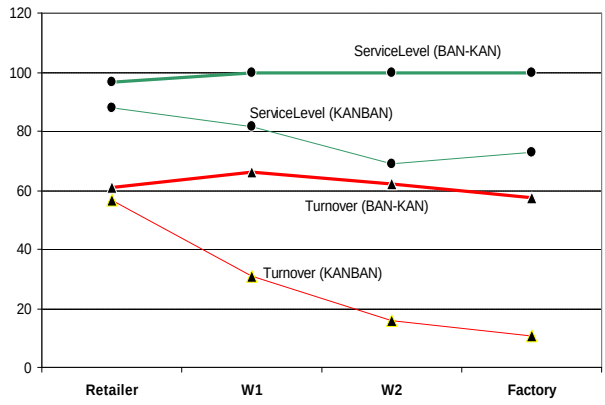

Fig. 5.31 Comparing the turnover and the service level along the Supply Chain

The next comparison concerns *costs* and *incomes*, which are very important measures in realistic Supply Chains. And even in this case the results appear surprising. One could expect the costs to become higher with the naive method due to the more frequent transportation of materials, and that is what simulation have shown. However, as represented in figure 5.32, the *ratio* between *incomes* and *total costs* continues higher in all facilities when compared to the KANBAN. *Holding costs* were obviously included and considered even with the inventory in motion.

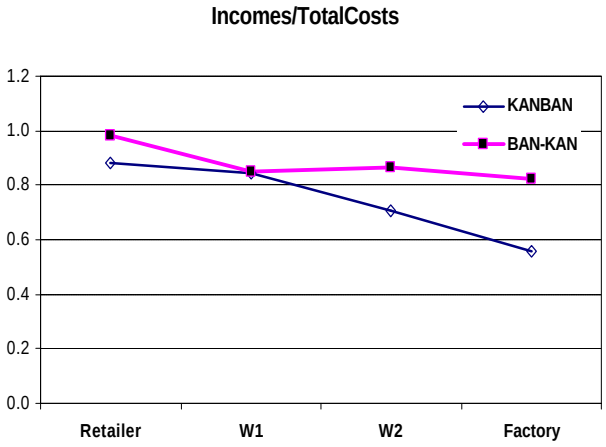

Fig. 5.32 Comparing the ration incomes/totalCosts along the Supply Chain

There is, however, an obvious negative aspect in the *BanKan*, which probably is the reason why it is not used in practice: the final costs of the products get too high, as figure 5.33 shows.

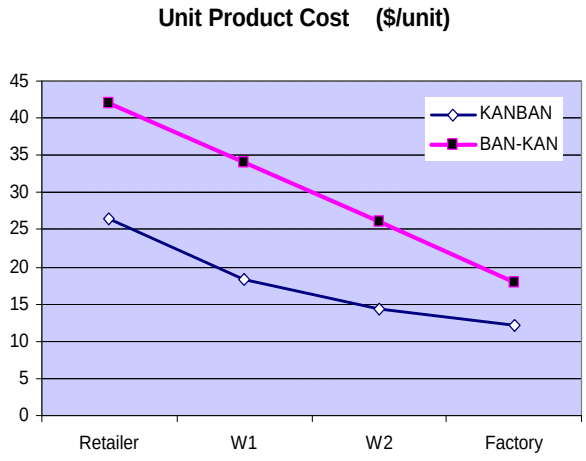

Fig. 5.33 Comparing the unitary product costs along the Supply Chain

Naturally, since the transportation costs have been made to reflect the product price, in both cases the cost of the product

raises in the direction of the last customer. Anyhow, with the KANBAN there could be *discounts of quantity* associated with the transportation, while in the *BanKan* case there were not. This explains the differences observed in the curves shown in this figure.

Notice that, as in this case a null *business margin* was considered at each facility (thus the product price increases only due to the transportation effect), the *BanKan* was imposing a *product cost* at the retailer of around 60% higher that the KANBAN. So, at least this difference would have to be projected to the last customer.

One must also notice that the transport costs were made dependent on the level of occupation in the delivering vehicle, as usually happens. Discounts of quantity were, therefore, considered. Yet, in all the facilities these discounts were the same. As we said, these discounts are responsible for the higher product costs obtained with the *BanKan*, since each facility orders much less material quantity in each order than in the other case. This, obviously, make the product prices higher.

5.5.8 Conclusions

The surprising results obtained by simulation on comparing these two reorder methods let us conclude that at least it could be interesting to study the effects of the naive *BanKan* method in other kinds of Supply Chains, as well as under diverse demand patterns and with more than one product. At least for the demand variability considered in the present case, the results show a superior operational performance in the naive method, and it was also observed significantly less *stockouts* at the retailer level. This, of course, must have a meaning, which in a certain way may be pointing to the idea of *demand pipeline management* (see Hewitt, 2001), or, who knows, in the future, to running tunnels of products.

Another interesting conclusion emerging from this study is the fact that using the *BanKan* method, one does not have to care too much about speed in the delivery, but instead about exactness. Actually, as this method is mainly regulated by time, and not by quantity, it may relieve managers and suppliers of the constant stress of delivering faster and faster, and, instead let them plan their jobs with more precision and tranquillity in the time domain. If a supplier is able to deliver in two hours running like mad, for sure it will also be able to deliver in five hours spending less energy, running slower, or, perhaps, serving more facilities than before. The only difference is that the quantity *Qo* will have to be a bit higher than in the previous situation. Most important is that the supplier will be at the client facility at the scheduled time, so that the synchronism with the demand at the retailer can be maintained as high. Of course, this kind of behaviour would imply a serious sense of *cooperation* between all the Supply Chain operators, revealing once again the value of *trust*.

Apart from this, using the *BanKan* at the "inner" facilities of the Supply Chain would also mean less space needed for inventory, less complex software for processing orders, less expensive stocking processes, which obviously would result in less expensive warehouses. Perhaps then the final price of the product could be reduced enough at the customer level.

This study was published in Feliz-Teixeira & Brito (2005a).

5.6 Case 3: distributed simulation game for supply chain management training

Here we present a computer application planned to be used as an interactive tool for training in Supply Chain management. Implemented in Visual C++, this application embeds the "*Cranfield Blocks Game*" network structure (Saw, 2002), and uses precisely the same demand patterns as the manual version of the game. However, instead of reducing the play to 12 reorder cycles, as the manual version does, it extends those patterns throughout time till any number of reorder cycles, which makes the results become more useful and interesting even for didactic purposes. At the same time, the present application substitutes the classroom table with the computer screen, and can be made to run in *AutoPlay* mode, meaning that the game can also be played with only one player or even automatically, with no players at all. In a certain way, this approach comes closer to a distributed Supply Chain simulation, apart from the fact that each *lead time* is fixed and constant, as the manual "*Cranfield Blocks Game*" states, and the stock policy used in *AutoPlay* is empirical.

Since each "player" application communicate with the "server" by means of the TCP/IP protocol, the players can be spread by different computers, or even be placed at different geographic locations, as long as they are connected to the Internet.

Results achieved both with an automatic running session and with the involvement of a group of students from the *Escola de Gestão do Porto* (EGP)[7] have been recorded and compared.

[7] http://www.egp.up.pt/

5.6.1 Introduction

Interactive games are nowadays used as powerful tools to exercise people's ability to manage a wide range of complex systems, from flight simulation to the control of water flow in dams under extreme conditions of rain, for example. Following such a tendency, *Supply Chain Management* (SCM) also started to apply these techniques to improve the ability of managers in resolving faster and more efficiently some problems they are expected to face in the real world. Training before to fit later is the philosophy of such approaches.

Related to the sciences of management, one can already find games like the *International Logistics Management Game* (Grubbström, 1995); the *Corporation: A Global Business Simulation*; the *Beefeater Restaurants Microworld* (G.S.D.L., 2003); as well as the *Goldratts Game*[8] and many others, but it seems that one of the oldest and most popular games directed to SCM is the "*Beer Game*" developed in the 1960s (Sterman1989), with which the *Bullwhip effect* (amplification of demand at the upper levels of the Supply Chain) have been observed, as well as the importance of communication between partners stated. Many of these games, however, are still being used as manual tools. The "*Beer Game*", for instance, can still be bought inside a box with written instructions, some pieces of paper and some more little blocks representing the material units used for stocking. Nevertheless, at the moment it is also possible to play the "*Beer Game*"

[8] This game is described by Goldratt in his "novel", "*The Goal*"(North River Press, 1992, pp. 104-112.). It can be used to illustrate many concepts in Production Management.

electronically, after David Simchi-Levi[9] and Phil Kaminsky[10] developed their *Web Based Beer Game*[11].

There are obviously some disadvantages to presenting these games in the manual form, the most significant being: (1) the space required to play the game; (2) the need for an extremely well organized manual communication scheme between partners, to ensure time synchronization (usually the job of another person); (3) the usage of cards to communicate demands; (4) and the non existence of a common support where the overall Supply Chain behaviour is to be shown to all the members, during the game or at its end. The record of stock data and the computation of other decision variables are also time consuming in the manual version, and can be highly improved using a computer application.

The present application, specifically focused on the "*Cranfield Blocks Game*", and named "SCGame", has as an objective to give the players these advantages. Since it uses a *client-server* technology, it was split in the applications "SCGame_server" and "SCGame_client". We will present each of these applications in this section.

5.6.2 The "Cranfield Blocks Game"

The original version of this game uses a multi-level structure to represent the Supply Chain, thus it can be considered a more generalist tool than its counterpart "*Beer Game*", which looks at the Supply Chain as a linear structure. Yet, the primary idea of the "*Beer Game*" is maintained, and one could say that the "*Cranfield Blocks Game*" is simply as a more complex version of the former.

The structure of the "*Cranfield Blocks Game*" (Fig. 5.34) is based on four depots (retailers) that are connected to two central warehouses and these warehouses connected to a single factory, which in turn is served by an ideal supplier.

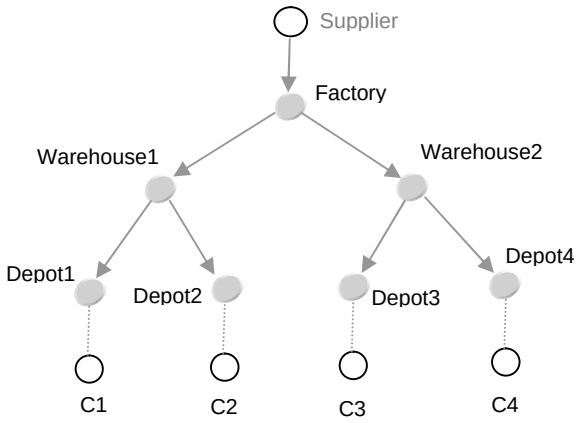

Fig. 5.34 Structure of the "Cranfield Blocks Game"

The intent of this configuration is to give the players a better sense of the behaviour of a real Supply Chain, at least about the need to fulfil the demand of more than just a customer in each facility, and how such a need can already introduce a significant variability in the demand at the upper levels, making it more difficult to establish the optimal management policy, even when a single product is considered. That is the case of this game. In addition to this, this configuration allows the results to give an idea of how the response of the chain depends on the demand amplitudes, since different levels of demand are "established" at the last customers (C_j), directly connected to the depots.

Usually, this game is played by seven groups of students, each group being

[9] Prof. at the Massachusetts Institute of Technology (MIT)
[10] Prof. at the Industrial Engineering and Operations Research Depart., University of California
[11] http://beergame.mit.edu/

responsible for the management of a single facility. The customers demand (C_j) is random generated, and the supplier (up in the figure) is considered an infinite source of materials. Thus, the students are only endorsed to manage from the *Depot1* to the *Factory*. No *backorders* are allowed, and each facility must adjust its *reorder quantity* to a multiple of a certain *Base Quantity* (BQ) established at the beginning of playing and which increases from left to right and as the facility approaches the *Factory*. The activity in the chain begin with the demand of products made by the customers (C_j), and the goal of each facility manager is to fulfil the demand keeping the stock as low as possible without incurring in *stockouts*.

The average amplitude of the costumers demand also increases from left to right, meaning that *C1* < *C2* < *C3* < *C4*. The initial conditions of the game are the following:

Facility:	Demand	Initial Stock	Reorder (BQ)
Depot1	C1	40	20
Depot2	C2	60	50
Depot3	C3	40	40
Depot4	C4	100	80
Warehouse1	...	120	100
Warehouse2	...	200	120
Factory	...	200	200

Table 5.4 Initial conditions of "Cranfield Blocks Game"

And the original customers demand patterns for the 12 cycles are:

C1 = 20, 0, 10, 20, 20, 20, 10, 20, 0, 0, 20, 20.
C2 = 30, 40, 10, 20, 40, 50, 40, 10, 20, 50, 40, 30.
C3 = 30, 30, 30, 40, 20, 10, 10, 20, 30, 10, 30, 20.
C4 = 10, 50, 30, 50, 60, 50, 20, 30, 10, 20, 50, 40.

These patterns will be maintained in the computer version precisely till the 12th cycle, but then they will be replaced by patterns generated based on random *Normal* distributed numbers. This first pattern was strictly preserved so that the results achieved with the computer version could be compared with the results obtained with the manual version.

Finally, in order to be able to compare the management of the different facilities, we have introduced a *classification* in the computer version that probably can diverge from the criteria used by the father of this game. Anyhow, for our purposes we decided to classify the management of each facility by the following criteria:

$$CLASSIF = 100\frac{accDemand - 1.5 \times accStockouts}{accArrived} \tag{5.6}$$

Where *accDemand* is the accumulated demand at the end of the game, *accStockouts* the number of *stockouts*, and *accArrived* the amount of materials arrived from the supplier. Thus, this classification gives an idea of the efficiency of the management, giving special emphasis to the *stockouts*.

5.6.3 The client-server application

To implement this game, we have developed a *client-server* application using TCP/IP, the *server* being responsible for generating the demand of the customers, game initialization, synchronization and control of the time and the communications between partners. On the other hand, the *client* is the interface that each manager uses to manage the inventory of his facility.

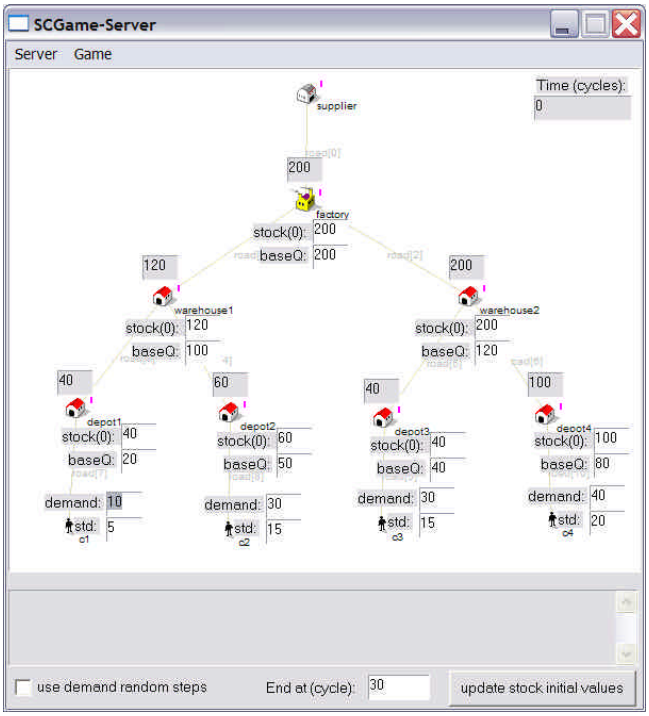

Fig. 5.35 The server application "SCGame_server"

Before starting the game, the *server* (Fig. 5.35) must be launched in the operating system (*Windows*) and will wait for the *clients* to ask permission to connect. As each *client* asks this permission, the *server* assigns to it a free facility, and will refuse any further connections once the Supply Chain is complete. In that moment, the *server* will send a message msg::play to each *client*, signalling that the game may begin. The time will only advance to the next cycle whenever all the *clients* have played the present cycle, thus ensuring the synchronism between them is maintained.

The *server* is the centralised controller of the game. It does not only establish the access to the players but also ensures that the correct time holds for all of them, and also controls the flow of the demand between partners. This is achieved by the simple exchange of messages between the *server* and the *clients*.

As it can be deduced from the previous image, this version can be made to run for longer periods of time than the original 12 cycles of the manual game, and also allows setting the initial inventory levels of the facilities, their *Base Quantities* (BQ) as well as modifying the customers' demand, even if this actually starts with the values of the manual version. Finally, this version gives the manager the possibility of introducing "*random steps*" in the customer's demand, which can be used to train reactions to highly unstable markets, or to make experiments about *flexibility*, for example.

During the game, it will also be possible to have an idea of the overall Supply Chain behaviour, since the inventory of the facilities can be shown graphically to the user, as in the example of figure 5.36.

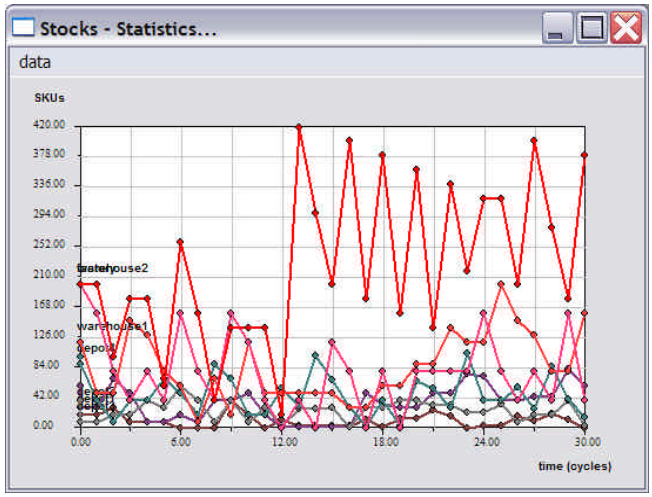

Fig. 5.36 Example of inventory runs of the facilities shown at the "SCGame_server"

As suggested in the next figure (Fig. 5.37), the game can be played in a room with each group of students seated around a computer, while the *server* runs in another computer nearby, if needed with its display projected on the wall as a way of letting the groups access the evolution of the

inventories of their partners. The game can also be played by internet if the *server* is previously running in a reachable IP address.

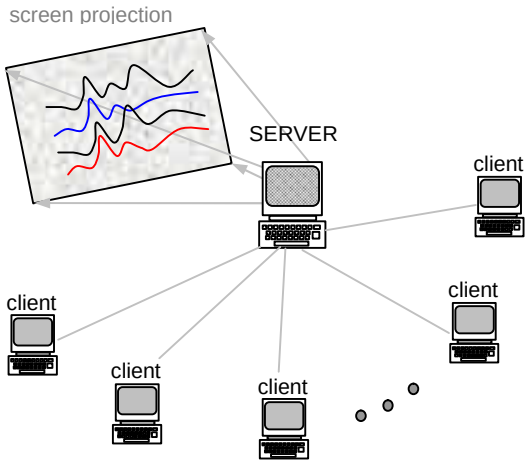

Fig. 5.37 A simple apparatus to play the game

On the other hand, the *client* application is the tool with which the groups manage their facilities, as well as communicate the demand to their suppliers and serve their customers requirements. This application, less complex than the *server*, is the real player's interface, and can be made to run in any location of the computer network as long as it will be able to reach the *server* by TCP/IP.

All the *clients* have the configuration shown in figure 5.38, and differ from each other by the natural amounts of stock and BQs, as well as by the facility name and the designation of their customers and their suppliers. The figure shows a *client* assigned to the *Factory*.

To help the player to better locate its facility in the Supply Chain, this application uses an image of the network where all the facilities are represented.

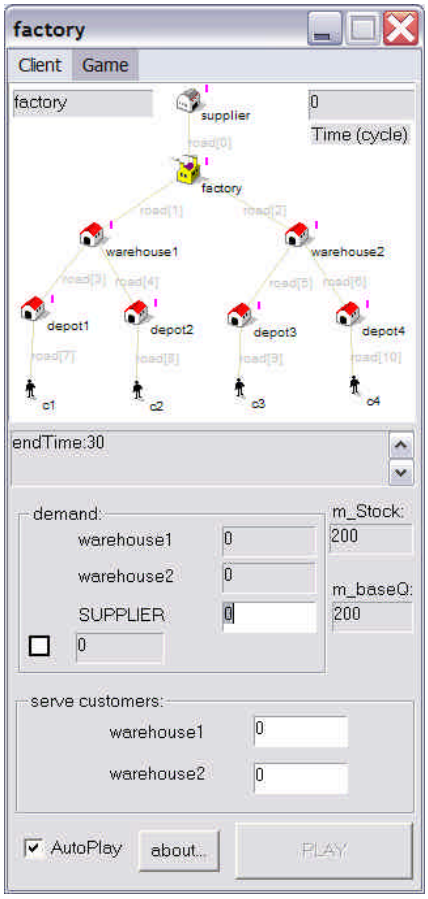

Fig. 5.38 A SCGame_client, in this case assigned to the facility that corresponds to the factory

There is also a window where messages coming from the *server* are displayed, and a check button of *AutoPlay* which lets the application react automatically to the customers demand and decide the quantity of material to order to the supplier. In any case, the criteria used in these decisions are empirical and do not follow any particular model known for stock control, since the intention was to simulate the kind of "empirical" behaviour that often is used in managing small stock resources. Even so, the method was implemented based on an experimented manager advice.

The *client* application has two other important zones, shown in the same figure: the demand zone, where the demand from

customers is automatically updated, and the requirements to the supplier placed; and a serving zone, where the user can fulfil the requests of materials to be sent to the customers. These operations will be executed when the PLAY button is pressed. This button will then be disabled by the application till the next time the *server* enables it, meaning the facility is again allowed to play.

Another feature concerning the *client* application is the possibility of tracking some important parameters during the game, as is the case of the *inventory*, the *demand*, the *quantity arrived* from the supplier, and the *stockout* level, for helping the player in making better decisions. This data is presented in the form of a graph appearing next to the *client* application as long as the menu option *Game->Statistics* is chosen (Fig. 5.39).

Fig. 5.39 Menu statistics

A graph of this type, in the case related with the *Depot1*, is shown in the next figure (Fig. 5.40).

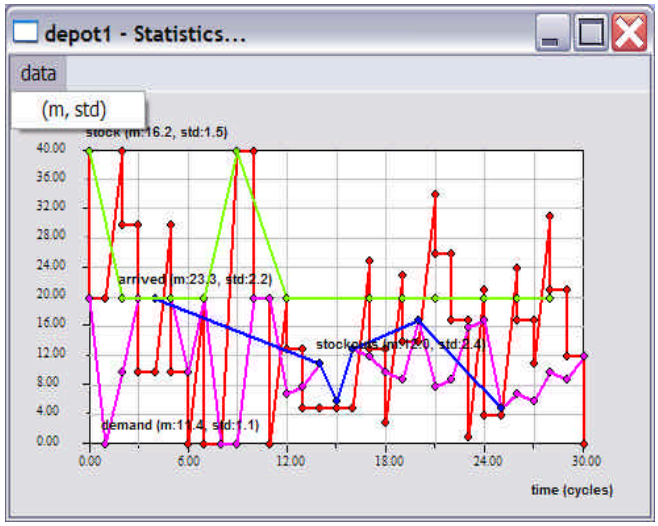

Fig. 5.40 Graph representing the stock, the demand, the materials arrived and the stockouts

In addition to the visualization of these parameters, the player can use the menu option *data->(m, std)* in this window to automatically compute the average and standard deviation of each of the data series represented.

5.6.4 Results and comments

As we said earlier, the results presented here have been obtained in two sessions. Firstly, the game was made to run automatically with all the facilities in *AutoPlay* mode, during 30 demand cycles. In the second session, the game was played by seven groups of students attending a Masters course at the *Escola de Gestão do Porto* (EGP), and also during 30 cycles. No display of partners inventories were used in this experiment, meaning that each facility had to be managed in the "darkness".

In each experiment, data like that of figure 5.40 was collected from each facility, and then its *average values*, *standard deviations* and *accumulated values* computed, as well as the *final classification* of the facility. These results were then

recorded and handled in an *MS-Excel* spreadsheet for the two experiments, in order to make the comparison between them. Notice that the unity of stock used here is again the SKU, known worldwide as *Stock Keeping Unit*.

Different types of calculations could be made based on the raw output data retrieved from the game, but for our purposes we decided to present only some results based on *averages* and on *standard deviations*. Averages will give us the sense of how each facility was in general handling the parameters of interest, while standard deviations will let us have an idea about the variability observed in handling those parameters. Based on this type of analysis, we expect to get a first sense about the capabilities of the present tool.

In figure 5.41, for instance, the average values of these parameters are represented, calculated after the game has been played by the students. It shows that even if the final classification of the management of the facilities exhibits an almost flat behaviour, an obvious tendency in the other parameters (except the demand, due to be an input) is also observed: to grow from the left to the right of the Supply Chain, as well as when approaching the *Factory*. This is precisely the tendency induced in the system by the differences in the BQ quantities, as we have stated earlier. Thus, due to the fact that these results tend to be proportional to the respective BQ of the facility, it seems all groups have used the same kind of empiric reorder policy, and that they faced the same difficulties on managing the facilities.

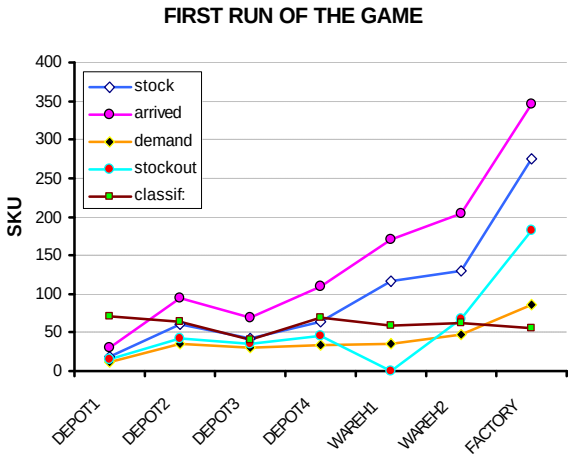

Fig. 5.41 Results obtained in the students session

These results may be compared with the data obtained with the *AutoPlay* session, represented in the next figure (Fig. 5.42), where all the facilities were using precisely the same inventory control empiric policy, as previously noticed.

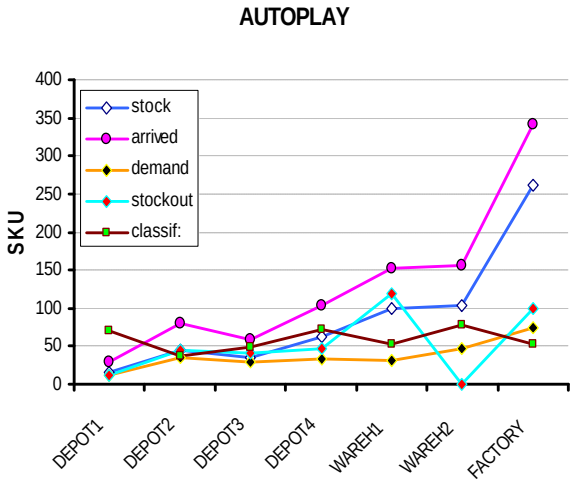

Fig. 5.42 Results obtained in the AutoPlay session

Although the various parameters seem to have improved slightly in the *AutoPlay* case, and the level of *stockouts* have been reduced, the two cases show a similar tendency to follow the induced BQ differences. Maybe this would mean that, in an empiric base, few improvements would

be achieved due to the "experience" or the "ability" of the manager. In fact, in the two cases the classification is maintained almost constant among the facilities.

Though, when we compare the variations observed in the two sessions by means of *standard deviations* charts, shown in figures 5.43 and 5.44, significant differences can already be noticed (notice the difference of vertical scales in the two graphs).

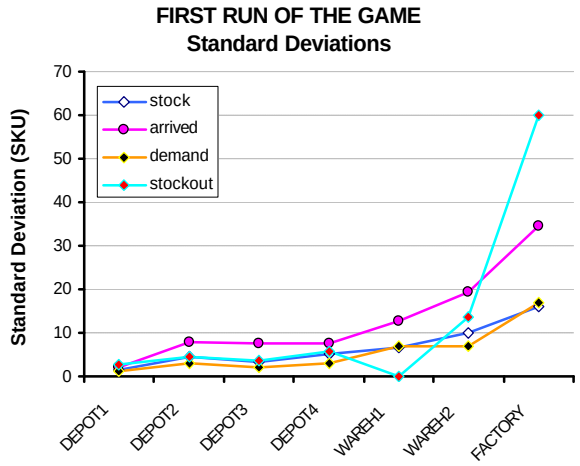

Fig. 5.43 Standard deviations in the students' session

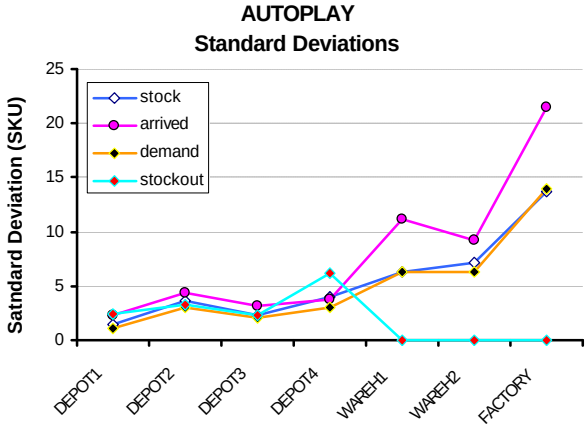

Fig. 5.44 Standard deviations in the AutoPlay session

From these, it seems obvious that an "experienced" manager (*AutoPlay*) would in fact be more able than a beginner to handle the facilities in a smoother way. In effect, the variability in the *stockouts* in the *AutoPlay* session was drastically reduced at the *Warehouse1*, *Warehouse2* and *Factory*. The amount of materials ordered was reduced as well.

Many other tests could have been done using these sessions, but that was not our objective. Other types of experiments will mainly depend on the imagination of the game's supervisor. Also, as the game lets the players visualize and record the evolution of important parameters along time, even tests concerning the *visibility* or *non-visibility* between partners can be made, depending whether the overall Supply Chain data is *projected* or *not-projected* on the wall, where all the inventory levels can be turned *visible* or *invisible* to all partners.

5.6.5 Conclusion

Based more on the potential of this application than on the results presented here (these are just an illustration of such potential), we conclude that this computer application represents an interesting tool for Supply Chain *management training*. In effect, the results achieved with the manual version can also be obtained using this tool, and other interesting aspects are made available additionally, like the portability and the flexibility of a distributed computer application, the capability of considering the scenarios *visible* or *not-visible*, as well as the introduction of *demand steps* in the ultimate customers, which we imagine can be useful in future tests concerning studies of *Flexibility* or *Agility*, for example.

As a last remark, these results were also used in the validation of the non-distributed simulator described in the previous chapter, since the same "*Cranfield Blocks Game*" Supply Chain has been simulated in such a tool. This work was published in Feliz-Teixeira et al. (2004).

5.7 Case 4: computing the rigidity matrix of a didactic Supply Chain

This case deals with the calculation of the *rigidity matrix* for the "*Cranfield Blocks Game*" structure. The results presented here have been published in Feliz-Teixeira & Brito (2004), as previously mentioned. A theoretical base on this matter can be found in this article or in the section "*About the flexibility (matrix)*" of chapter 3. We therefore consider that the reader has some knowledge of this theory. Here, we only mention the procedure and the results obtained by simulation, and comment on them when appropriate.

5.7.1 Inventory-rigidity-to-demand

As we maintain, the *flexibility* is seen as the inverse of the *rigidity*. For convenience, we use the *rigidity* for the representation of the matrix, since it always deals with finite quantities. The matrix of the *inventory-rigidity-to-demand* will be filled in here by means of data acquired by simulation.

For this purpose, all the facilities in the "*Cranfield Blocks Game*" have been set to use a *(r, Q)* stock reorder policy, and the demand at each *Depot* have been initially kept constant at the average value stated by the manual version of the game. Then, at a certain moment of the simulation process, the demand of a certain *Depot* was made to receive a sudden increment in its value, and this new situation was maintained till the end of the replication. This procedure was executed for each of the *Depots* separately. In each case, the *rigidity* was measured at the nodes of the Supply Chain by applying the equation (3.19), and then the matrix was represented. Each simulation has been run with only 3 replications, but, for simplicity, averages and respective uncertainties have been computed for 95% confidence, as if more than 20 samples would have been used. In order to simplify the reading of the results, null values have been substituted by dashed lines in the matrix.

The next figure (Fig. 5.45) is an example of how the inventory level of *Depot1* behaved when subjected to a demand step of 2.5 times its usual average value. Notice that even if the system seems to answer well to the new requirements, a certain instability that did not exist before was introduced.

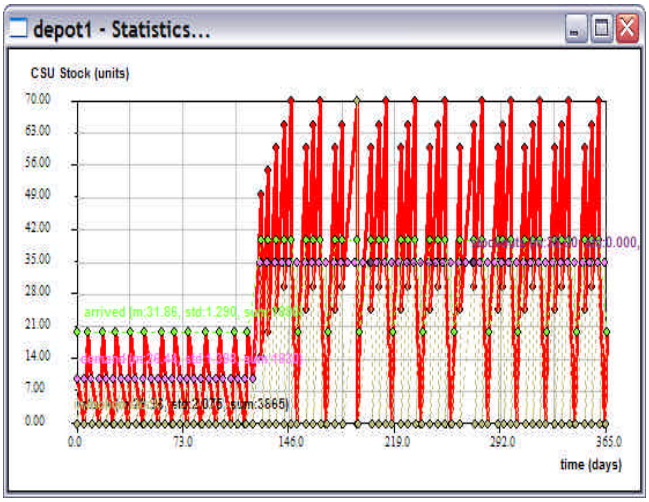

Fig. 5.45 Behaviour of Depot1's inventory in reaction to a step in the demand

In the next figure (Fig. 5.46) the behavior of the *Equilibrium(t)* function for the same

case (*Depot1*), described by equation (3.17), is presented. We opted to use this form of presenting the equilibrium to the reader since it is more interesting to plot graphically than the one resulting from equation (3.18). From this figure, it is already evident that a certain instability has been introduced due to the abrupt change in demand. This instability, however, is mainly due to the lack of an integer relation between the new demand and the quantity ordered to the supplier. By readapting this quantity one could easily reduce this sort of instability. This also lets us induce that different reorder policies can lead to different *flexibility* values.

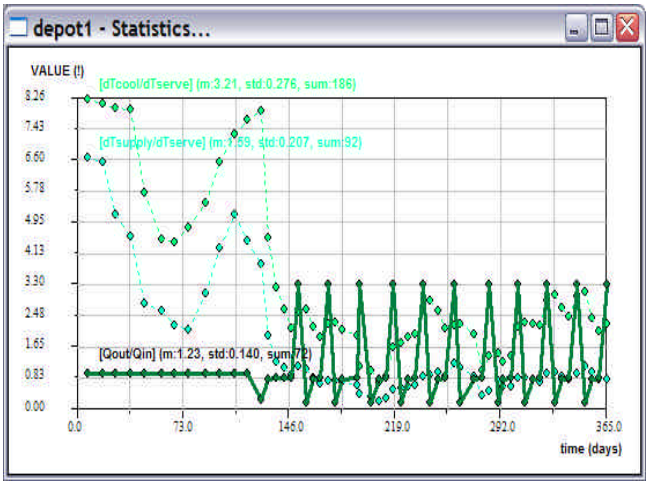

Fig. 5.46 Equilibrium(t) function of Depot1 before and after the step in the demand (bold line)

It appears also obvious from this figure that the "*time to recover the equilibrium*" (*Tq*) would be difficult to compute in cases like these, since the system does not stop oscillating. Nevertheless, as we measured the *rigidities* using the expression (3.19), this problem is eliminated, since this method stabilizes the metric independently of the time observation window.

Finally, in figure 5.47 we present the *rigidity* matrix for all the facilities in the Supply Chain, resulting from the data collected in the four simulation processes where *demand steps* have been applied to the *depots*. The values are in SKUs/day.

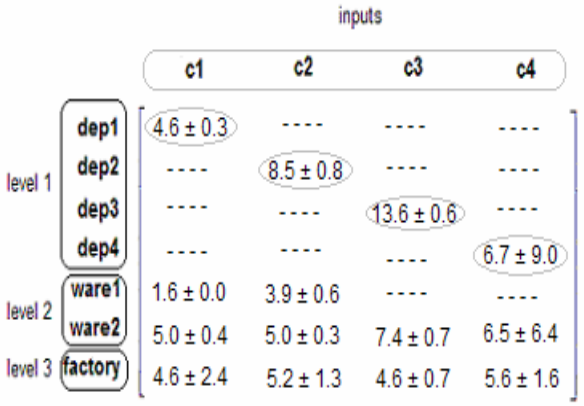

Fig. 5.47 Rigidity matrix for the "Cranfield Blocks Game"

It is important to notice that this matrix is only valid for certain specific conditions, and so, these must be well specified before interpreting the results. In this case, the conditions can roughly be stated as:

- All **Step Demand Amplitudes** = 2.5 x (usual average demand).
- All **Reorder Policies** = (r=ro, Q=Qo), as the game stated.
- Each facility **fleet** = 1 vehicle Van.
- All Depots start from **equilibrium**.
- Etc.

5.7.2 Interpreting the matrix

It is now clear from this matrix that a step in demand of amplitude 2.5x on the *Depot1* will let us observe an average "*power spent in imbalance*" of 4.6 SKU/day in this facility, as well as 1.6 SKU/day at the *Warehouse1*, 5.0 SKU/day at the *Warehouse2* and 4.6 SKU/day at the

Factory. It can also be expected that the same kind of step at *Depot3*, for instance, will lead to a higher rigidity at this facility as well as to a higher influence in the *Warehouse2*, which will exhibit 7.4 SKU/day of imbalance. Anyhow, such a step on *Depot3* will never influence the *Warehouse1* behavior. And so on…

Another interesting aspect that can be inferred from this matrix is that probably the imbalance of 5.0 SKU/day measured at *Warehouse2* is not motivated by the demand steps, as it stays almost constant in the cases *C1* and *C2*. This is probably a residual imbalance due to the natural uncertainty in the demand of *Warehouse2* due to the unsynchronized reordering moments of *Depot3* and *Depot4*.

Another aspect resulting from this measure is that it was impossible to obtain a reliable value for the rigidity in the *C4* case, as the uncertainties are of the same order of the averages. And, in effect, as the next figure shows, such a high *stockout* ratio was observed on *Depot4* that it could hardly lead to reliable values for the *rigidity*.

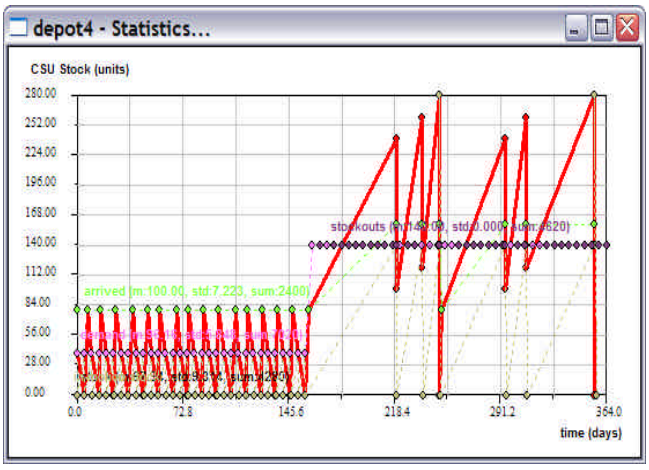

Fig. 5.48 Stock level and stockouts at the Depot4 facility

This has also been confirmed by the "*satisfaction*" observed at the customer *C4*, presented in the figure 5.49, which had dramatically rolled down after the "shock" on the demand.

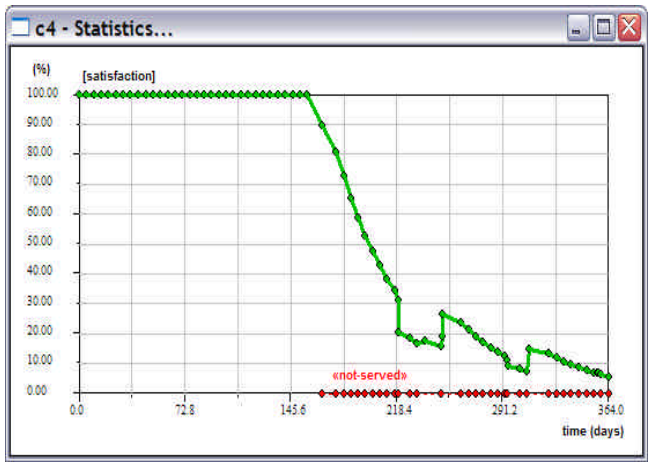

Fig. 5.49 "Satisfaction" at the C4 customer

5.7.3 Conclusions

This experiment lead us to conclude that the proposed methodology for measuring the *rigidity* of a Supply Chain is feasible and the results can somehow be easily interpreted, therefore pointing to the possibility of applying such a concept for reasoning and, through it, support decisions. At the same time, we also conclude that the *flexibility* depends more on the suppliers than on the clients, which automatically remind us of the idea of *cooperation* with suppliers. It seems also natural that the *flexibility* results form an interrelation between facilities from where concepts like *cooperation*, *visibility* and *trust* naturally emerge. In addition to this, it has also been shown that the *flexibility* can be seriously dependent on the type of resources available at the facilities and on how they are handled, such as the number and kind of

vehicles allocated to delivery, or the model used for reordering the materials, for example.

Even if the *flexibility* can potentially be made higher if working with more than a supplier, that is, by using redundancy, the common way to consider *flexibility*, such a method seems not to provide the manager with enough data for a quantitative reasoning, as it is more an empirical suggestion. It forgets, for example, that using more than a supplier for the same product frequently implies difficulties in relationships, obviously due to concurrence.

Finally, one must notice that the results achieved in this case have been obtained considering only "positive" steps in the demand, that is, by incrementing it, and nothing have been said concerning the opposite situation, that is, when there is a sudden *decrement* in the demand. However, as usually the flow of products is only in one way, it follows that the best approach for being aware of these cases is to maintain the minimum stock possible at the facilities, which, at the limit, would be a single unit of material (pure JIT).

Future research on this matter could include studies about other kinds of Supply Chain structures, not only didactic but also representing real systems, and, using the sort of analysis presented here, enrich the knowledge about the idea of *flexibility* with the final intent to make it more precise than what it actually seems at the moment. Tests could be made about *flexibility* in many aspects of the Supply Chain, as, for instance, with different reorder policies, or diverse demand amplitudes, as well as with various resources and policies related to transportation, or even others. This matter seems, in effect, an open field for tests using simulation.

5.8 Last comments

From the case studies presented in this chapter, one could finally get an idea about the type of practical problems that can be addressed by simulation techniques, and the detail that can be used to model and simulate these problems. One of the cases has shown how simulation can also be useful for training personnel in the practice of inventory control, and, even if it was based in a less sophisticated application than the main Supply Chain simulator, these two approaches have shown how the objective of the simulation interferes with the type of application chosen, and also that one does not have to exclude the other. We may notice, however, that the cases committed to the main simulator were simply examples of the sort of problems that can be modelled and studied by means of such an application, but of course do not represent the entire range of possibilities. They have, in fact, contributed to the validation of the simulator, or at least to obtain some degree of confidence about its utility for practical Supply Chain simulation, but, even so, several attributes of it have not been used, and so, many of its features have stayed invisible in this presentation. We hope that other types of Supply Chains and other problems can in future be studied by means of this simulation application.

References:

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2004). *On Measuring the Supply Chain Flexibility.* Paper presented at the 2004 European Simulation and Modelling Conference, UNESCO, Paris, France.

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2005a). *Comparing a Standard and a Naïve Stock Refill Policies by Means of Simulation.* Paper presented at the 2005 European Simulation and Modelling Conference, University of Porto, Porto, Portugal.

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2005b). *Using Simulation to Analyse the Procurement of Additives for Lubricants in the Oil Refinery of Porto, Portugal.* Paper presented at the Industrial Simulation Conference 2005, Berlin, Germany.

Feliz-Teixeira, J. M., Brito, A. E. S. C., et al. (2004). *Distributed Application for Supply Chain Management Training.* Paper presented at the Industrial Simulation Conference 2004, Malaga, Spain.

G.S.D.L. (2003). *Beefeater Restaurants Microworld*: Global Strategy Dynamics Limited.

GalpEnergia. (2003). *A Galp Energia - História*. Galp Energia. 2004, from http://www.galpenergia.com/

Grubbström, R. W. (1995). *International Logistics Management Game (ILMG)*. Linköping Institute of Technology, Sweden.

Harris, F. W. (1913). How Many Parts to Make at Once. *Factory: The Magazine of Management*.

Hewitt, F. (2001, May 2001). After Supply Chains, Think Demand Pipelines. *Supply Chain Management Review, 5,* 28.

Lambert, D. M., & Pohlen, T. L. (2001). Supply Chain Metrics. *The International Journal of Logistics Management, 12* (1), 1-19.

Ohno, T. (1988). *Toyota Production System: Beyond Large-Scale Production*. Cambridge, MA: Productivity Press.

Polczynski, J., Moroz, M., et al. (2004). *Inventory Flexibility to Demand Based on the "Cranfield Blocks Game" (a simulation study)* (students report). Porto, Portugal: GEIN-FEUP.

Saw, R. (2002). *Cranfield Blocks Game*: Centre for Logistics and Supply Chain Management (CSCM), University of Cranfield, UK.

Sterman, J. D. (1989). Modeling Managerial Behaior: Misperceptions of Feedback in a Dynamic Decision Making Experiment. *Management Science, 35* (3), 321-339.

CHAPTER 6 215
CONCLUSIONS AND FUTURE WORK 215
6.1 THE FINAL CHAPTER 215
6.2 SPECIFIC ADVANCES 216
6.2.1 The CSU concept 216
6.2.2 New delivery elements 216
6.2.3 New reordering policies 217
6.2.4 The Gk geometric factor 217
6.2.5 The ultimate customer 217
6.2.6 Ordering by Internet 217
6.2.7 Three levels of results 217
6.2.8 Demand steps 218
6.2.9 The theory of flexibility 218
6.2.10 Events receiving pointers 218
6.3 VERIFICATION OF THE HYPOTHESES 219
6.4 GENERAL CONCLUSIONS 221
6.5 FUTURE WORK 221
6.5.1 Rigidity (flexibility) studies 222
6.5.2 Reverse logistics 222
6.5.3 Holistic metrics 222
6.6 CLOSING 224
REFERENCES: 225

Chapter 6

Conclusions and Future Work

Confirmation of the hypotheses and some proposals for future research

6.1 The final chapter

The simulation approach described in the previous chapters has been developed from the foundations, and the resulting application for modelling and simulation reveals some particularities typical of these types of approaches, that is, a collection of some small advances. Although the basis are not especially new to be considered an advance in the field, but, the structure proposed for modelling the Supply Chain and some aspects related to its implementation may already deserve being thought of in such a way. The most relevant of these aspects will be enumerated in the next section of this chapter.

We may notice, however, that Supply Chain systems are quite difficult to simulate in practice with an interesting level of accuracy, not only due to their natural complexity, but also due to the types of tools available in the market for executing such a task and who is in charge of it (exceptional results cannot be expected if the analyst is simply the wrong person). Strong enterprises working with simulation allocate specialized personnel, computer resources and expensive simulation applications to the cause, but most of the average companies do not, and these reduce the commitment to simulation either by modelling the processes with multi-purpose applications or by outsourcing the task to a consultancy agency. These agencies, in turn, either use simulation packages dedicated to the field, as the present approach could be classified, or will again implement the model based on a multi-purpose modelling tool. Thus, the question is not whether the modelling and simulation must be made by the company or by an external partner; instead, it is more about competence and reliability. And the money available for it, of course.

Due to the simplicity shown, flexibility and potentials, our approach may be seen as a case implying neither the dedication of a specialist on multi-purpose tools, nor the huge amounts of money usually needed to acquire an application dedicated to Supply Chain. The only requirements are that the analyst is a specialist in *Supply Chain Management* (SCM) and that he some solid knowledge concerning statistics for treating and interpreting the outputs generated by the application. In this sense, this simulator becomes a tool for those who directly work in the field, as was intended in the beginning. To use the tool themselves or subcontract the help of a simulation specialist will solely depend on the time available for the task. That is, this tool was designed to be both intuitively used and

economical. At the same time, it may also be considered a small advance in the more general trial of simulating systems with applications specialized in each field of knowledge.

In this short chapter we will firstly review the most relevant features that might be considered advances for the Supply Chain simulation; then we will confirm the hypotheses stated in the beginning of this text; then the general conclusions will be stated; and finally, we will present some thoughts concerning certain aspects that could probably be explored in future.

6.2 Specific advances

The features listed in this section will be touched only superficially, since the main intent is enumerating and discussing them, but not to get into further details. For that, please consult the previous chapters.

6.2.1 The CSU concept

As far as we know, dedicated Supply Chain simulations are usually based on modelling each facility in the system in its particular functionality, and often reduced to the concept of a warehouse. Thus, a Supply Chain is treated as a kind of network of interconnected warehouses. On the other hand, models built with multi-purpose simulation tools represent these facilities either by single blocks with certain response times, or, when more detailed, by means of other blocks with increased complexity, but always focused on the specific type of the facility. The concept of *Costumer Supplier Unit* (CSU) introduced in this work is, in that sense, an advance in the representation of such a kind of system, as any common facility of the Supply Chain can now be represented, configured and simulated based on the same modelling element: the CSU. This approach is, therefore, both a generalization and a specification. This allows the analyst to build the model in a very easy way and configure the facilities with an equivalent simplicity. The dynamics of the relevant processes at each facility are also recorded graphically during the simulation, and can be accessed by the appropriate menu options of the CSU object. This enables the user to speedily inspect the evolution of the simulation and, in the end, its results.

6.2.2 New delivery elements

Another uncommon feature is considering vehicles of *transfer* in the applications for Supply Chain simulation. Most of the times, the process of delivery is simply substituted by an estimation of *lead-times*, frequently retrieved from some statistic function. This does not take into account, for instance, the interference of the *load level* at the facility in the delivery process. Facilities working under stressing regimes are expected to be less able to deliver appropriately, of course. As we defended in chapter 3, in the section "*More about delivery*", our approach is expected to be able to simulate also these situations, since *lead-times* may in fact become dependent on the processes that run in the facilities. The usage of *transfer* vehicles is an additional feature included in this simulator to allow modelling more complex transport structures, as is the case of using private vehicles to a certain point and then the public railway, for example.

Another important feature related to the transportation is the inclusion of the

xtunnel path to connect facilities, which automatically generates the appropriate vehicles in order to simulate a kind of *pipeline* movement. This can be used in future for studying *pipeline management*, for example.

6.2.3 New reordering policies

The explicit usage of reorder policies like the KANBAN as well is not usual to find in Supply Chain simulators, even if such a policy can easily be implemented as a particular case of a common *(r, Q)* policy. Besides, the introduced naive *BanKan* policy allows other levels of experimentation in the Supply Chain simulation, also useful for studying *pipeline management*.

6.2.4 The Gk geometric factor

Using the *Gk* factor (see the section "*Paths*" of chapter 4, for details) to draw the Supply Chain network in the form of a *scaled version of reality* makes possible the representation of long distances without the need for using geographic coordinates, and without affecting the consistency of the visual effects of transportation. This can be very useful when representing complex Supply Chains which include both regional and global structures. That is, for instance, the case of a *global enterprise*, which sources from around the world, produces in strategic points, and then delivers to locals dispersed on the planet.

6.2.5 The ultimate customer

The usage of an *ultimate customer* figure, almost always ignored in the simulation of Supply Chain systems, allows the analyst to create groups of customers driving the demand in a certain facility. As each customer uses a cyclical type of demand (with variations both in *time* and *volume*), this feature enables the user to refine the range of the demand inputted into the system, since it is possible to add the demand of a new product simply by adding a new customer to the facility; and, in other cases, turn the demand as complex as needed by inserting new customers for the same product with different cycles of ordering. Based on this, almost any type of demand can be generated, by the overlapping of the cyclic signals.

6.2.6 Ordering by Internet

Instead of imposing that a facility is to order its products only from the suppliers directly connected to it, the introduction of an *information path* (see chapter 3), gives the analyst the opportunity for modelling processes like ordering by telephone, by catalogue or by the Internet, where the suppliers may be located anywhere in the distribution network. Of course, the product will only be delivered if there are physical paths allowing such a move of materials, but, once it is possible, this is a useful method for representing many Supply Chain systems of nowadays.

6.2.7 Three levels of results

The graphical presentation of several dynamics both at the *product* level (product inventory, for example) and at the level of the *facility* (overall inventory), allows the user to inspect the behaviour of the facility from diverse perspectives. In addition, the presentation of some global Supply Chain measures introduces a third level of inspection: the *network* level. These levels may easily reflect the *operational*, *tactical* and *strategic* levels of management.

6.2.8 Demand steps

The generation of *demand steps* at the last customer figure is obviously an interesting feature for studying the ability of the Supply Chain to respond to sudden demand changes. This has been a hot issue during the last few years in the schools of management, leading to the exploration of concepts like *flexibility* and *agility*.

6.2.9 The theory of flexibility

The *theory of flexibility* developed by the author in the context of the present work, and published in its entire version in Feliz-Teixeira & Brito (2004), can perhaps be pointed out as the most relevant advance emerging from this research. This theory, superficially presented in chapter 3, may be applied to any general time dependent flow-like process, from a single node system to a system composed of a collection of nodes. In this last case it gives rise to the *rigidity matrix* of the system, with which the *flexibility* of the entire Supply Chain can be evaluated, for example.

6.2.10 Events receiving pointers

Finally, another little innovation was the introduction of event handlers that are able to receive any type of variable through a *void* pointer argument. Usually, these handlers, after the method *Loop()* calls the event rooter *Execute()* of the entity, are called by the rooter directly and with no arguments. So, no arguments are normally passed through the mechanism of the *event-list*. Here, however, and since we have added the *void* pointer *ppp* to the structure of the *SimEvent* object (please refer to the section "*The SimEvent object*", of chapter 4), this was made possible. So, from now on, when scheduling a future event, one is able to specify not only the type of the event, the time to occur and the entity to execute it, but also which object in particular it may handle, for example.

As it was stated in the section "*Event handlers*" of chapter 4, this can be used for implementing complex decision criteria for queue serving, for instance, as it allows passing entities directly from a *live state* to another *live state*, ignoring the queue in between. Only after being analysed, the entity would be sent to the appropriate position in the queue, if needed.

This simple feature can, however, result in some interesting changes in the form of representing the state diagram of the system, since now we can think in *queues* beside *live states* and not in serial with them. That is, a *live state* may directly be linked to another *live state*. For example, figure 6.1 shows again the process of imputing materials into the stock, where the *dead states (queues*) are in serial with the *live states*, as normal.

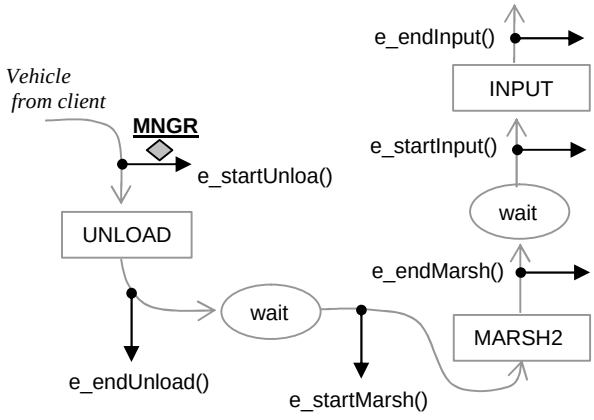

Fig. 6.1 States and events in the input stock process

On the other hand, figure 6.2 represents precisely the same process but taking into account the previous considerations. Notice

that the queues are now located next to the *live states*, almost as forming part of them.

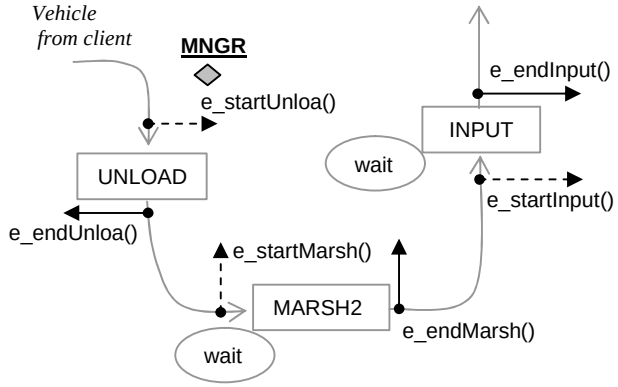

Fig. 6.2 New states and events in the same process

This way, *queues* are seen as properties of the *live states*, and each *live state* links directly to the next *live state* by means of the event of its ending. For example, when processing the event *e_endUnload()*, the entity can already directly call the event *e_startMarsh()*, passing to it the *order* that must be processed. When this event takes place, it will follow the new procedure:

```
e_startMarsh(void* ppp) //order=*ppp
{ If the marshalling queue is empty
    {
     if the MARSH2 activity is free
          {send the order to MARSH2;
           schedule e_endMarsh(order);}
     else //marshalling activity occupied
          {send the order to the waiting queue;}
    }//.....
  else //marshalling queue not empty
    { send the order to appropriate position in the queue;}
}
```

Then, at the end of this activity, the event *e_endMarsh()* will move the *order* to the next *live activity* (INPUT), by calling the *e_startInput()* method with the *order* as the argument, and then restart the marshalling:

```
e_endMarsh(void* ppp) //order=*ppp
{get the order from the MARSH2 activity;
 call directly event e_startInput(order);
 If the marshalling queue is not empty
    { get the newOrder from the top of the queue;
      put the newOrder into the MARSH2 activity;
      schedule e_endMarsh(newOrder);}
}
```

Notice that only the events of "end" had to be scheduled. The others can be simply called directly, without passing through the *event-list* mechanism. In this scheme, each activity may keep its logic completely private, that is, not accessible to other activities. At the same time, only the event handlers related to the "start" of activities need to be of public access.

The concept of "activity", in itself, may also be now understood as a more complex notion; in effect, as an association of a *live state* with *dead states*.

6.3 Verification of the hypotheses

In the statistical testing of hypothesis the comparison between two processes, usually a standard one and a proposed one, is made in order to infer about certain properties of these processes (see Mood et al., 1974). Such an inference results from the comparison of statistical data about those processes, the first being normally well documented, whilst the second is retrieved from some sort of experiment.

Even in these cases, that is, when there is enough quantitative data for supporting a judgment, it can be difficult to infer about the best option, since frequently this choice

must be made by taking in account diverse criteria and diverse parameters. When a single parameter is considered (for instance, the average of a certain quantity), the process gets easier, since it is reduced to a simple arithmetic comparison. However, when, for instance, the variability is also considered, as in the example of figure 6.3, the problem frequently increases in complexity. From the example of this figure, one is already able to recognize that such a confrontation is not trivial anymore. Within a criterium in which "the highest is the best", which of the processes to choose?

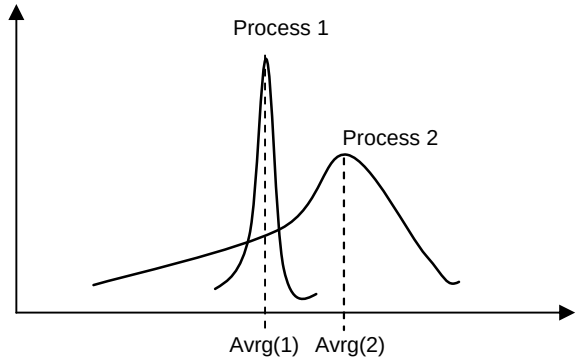

Fig. 6.3 Two different processes to be compared

In effect, even if an *average test* would make us obviously choose process 2 the fact is that, due to variability, this process is much less "trustful" than process 1, and can even give worse results...

In the case of the simulation approach presented in the previous chapters, the difficulties rise, since in truth the proposal is not comparable in statistical terms. This is due to: (1) there not being a standard system available with which the proposal could be compared; (2) there also not being enough quantitative data available for a good characterization of the proposal.

So, looking again at figure 6.3, this means that neither curve 1 nor curve 2 can be traced. The statistical comparison is, therefore, not an option, in this case.

The hypotheses stated in the beginning of this text must, therefore, be tested through a certain use of intuition supported by the *potentials* of the new proposal, which have obviously be confirmed by the case studies described in the previous chapter. Under these circumstances we may infer that the hypotheses H1 and H2, concerning the ability for a "flexible" modelling, and stated as: **H1**) *the simulator will be able to represent diverse kinds of Supply Chains in terms of different network structures, and allow the analyst to draw conclusions about their performances at least based on some standard Supply Chain measures*; and **H2**) *the simulator will also be able to represent different stocking and producing policies at each facility, as well as to test different kinds of vehicles for delivery, in order for these to be compared by means of some standard measures*, are accepted.

On the other hand, we may also deduce that the hypothesis H3, concerning the ability for analysing the "flexibility" of the Supply Chain, stated as: **H3**) *by means of this simulator it will be possible to obtain a quantitative measure of the "flexibility" to demand variations, not only related with each facility but as well with the entire Supply Chain structure; different scenarios are expected to be possibly compared based on these sorts of measures*, is also accepted.

6.4 General conclusions

Technically, we may in general conclude that the expectations at the beginning of this work have been fulfilled, and that the approach for modelling and simulation developed in the meantime has revealed various interesting potentials, not only concerning its ability for representing and studying diverse types of Supply Chain structures, and for estimating the *flexibility* of a Supply Chain, but also due to several other specific advances that may become useful in the analysis of future cases. The present version of the application shows, in effect, that various types of problems may be addressed and a wide range of results obtained, including the characterization of the Supply Chain in terms of some common metrics and the built of the *rigidity matrix* of the system being simulated. Such a diversity of possibilities justifies that this approach is classified as "flexible", in our point of view.

Apart from the simulation application naturally resulting from this work, which is a small step under the level of a commercial product, and for that reason is being used as a consultancy instrument, a distributed application for simulating the "*Cranfields Blocks Game*" has also been developed, and became an interesting tool for training people in SCM. These two applications may be considered the most relevant practical products of this study.

In respect to the strategy used in this project, it is interesting to report that a *top-down* methodology of learning followed by a *bottom-up* methodology of development and implementation has proved especially fruitful, since most of the improvements have been supported by a solid groundwork.

Apart form this, also deserving mention is the fact that no special constraints had been set *a priori* regarding the flow of the work, and this has given space so that other ideas could emerge from the practice and from the studies, as was the case of the *theory of flexibility*, for example.

Other aspects deserving reference, due to their precious contribution to this work, are: interaction with different schools of knowledge, resulting in a wider perception of the reality and enrichment of the research; discussions with people deeply connected to the matter, as an excellent way for achieving a good balance between practical issues and academic knowledge; publication of articles about the developed work in conferences (or magazines), which has revealed an excellent exercise for systematizing the research along time; and, finally, the chance of receiving regular financial support, providing the necessary independence for achieving the goals[1].

6.5 Future work

There are several issues related to the matters exposed here that could become subjects of study in future projects or works. Apart from the obvious interest of simulating some more real cases, with the purpose of contributing to a finer validation of the approach, other studies based on results obtained with games like the "*Cranfields Blocks Game*", for example, could be made in order to infer about the

[1] In this case, thanks to the *Fundação para a Ciência e a Tecnologia* (FCT), the institution of the Portuguese State in charge of overseeing the funds endorsed by the EU for scientific and technological development. Site at: http://www.fct.mctes.pt

role that the different SCM policies play in the operational behaviour of the chain. In this sense, we believe that the present simulation application may be significantly valuable as a test-bed for future Supply Chain developments, and an open field for applied simulation.

In this section, we will suggest some avenues for future research that, in our opinion, are considered of great value in the context of the present SCM.

6.5.1 Rigidity (flexibility) studies

The importance of concepts like *agility* and *flexibility* in current SCM will justify that more studies about this issue will be done, by means of measuring the *rigidity* of the facilities and by analysing the *rigidity matrixes* of a larger number of practical and didactic cases. As mentioned earlier, some ERASMUS students have developed a first work about this topic, but more serious and reliable experiments of this kind must be done, including the study of real Supply Chains. The problem of real cases resides, of course, in the difficulty of confronting the *rigidities* obtained by simulation with real measures of *rigidity*, which most of the time are impossible to achieve, since this would imply injecting real demand-steps in the system. Nevertheless, at least the measure of the *"residual" rigidities* could be tested, since in these cases the *rigidity* may be computed during the normal operation of the facility. This would, in principle, provide very useful information for the validation of the concepts involved. At the same time, such a practice would take to the environment of real companies the idea of using these new measures, which in fact are very simple to implement.

6.5.2 Reverse logistics

Another very interesting subject in the present SCM is the issue commonly called *Reverse Logistics* (RL), that is, the dynamic process dealing with the flow of the products that must be removed from the Supply Chain. These include dead products, products with defects, returned from customers, etc. Normally, this flow is not considered when simulating Supply Chain systems, mostly because the typical volume of materials in question is significantly low, when compared with the normal flow, and the products' value is definitely reduced.

In reality, however, such an amount of "garbage" cannot be abandoned in the streets, and this implies that resources must be assigned to resolving the problem. Even though a significant number of persons are still looking at the Supply Chain as a flow of products in a single direction, in effect, the reverse process is real and cumulative, and, in many cases, it is giving serious problems to the managers. It is, therefore, important to study the dynamics of the Supply Chain in terms of these two grand movements of materials in opposite directions.

The Supply Chain simulator described here could also be used to address this sort of problem. As earlier mentioned, the structure of the object <*product*> already includes some variables aimed to be used for this purpose. Yet, to effectively prepare the application for this, some more time devoted to software development would be needed.

6.5.3 Holistic metrics

In the section "*A step up to holistic metrics?*" of chapter 2, we have briefly discussed the current interest revealed by the simulation community in searching for

metrics to assist with the interpretation and analysis of complex systems. There, we suggested that perhaps an approach inspired by the formalisms of *Fourier analysis* or *Quantum Mechanics* could probably lead to interesting results. Frequently, analysts have the tendency of reasoning in the *time* domain, and not in the *frequency* domain, although the latter is in general less complex that the former.

Our suggestion was that the overall system state (ψ) would be represented as a summation of several base functions (ψ_i), weighted by probability factors (α_i). The benefit of this was that each base function (ψ_i) could be arbitrarily chosen by the analyst, and the probabilities (α_i) easily computed in the simulation. The overall state of the system would then be expressed as follows:

$$\psi = \alpha_1 \psi_1 + \alpha_2 \psi_2 + \ldots \alpha_j \psi_j + \ldots \alpha_n \psi_n \qquad (6.1)$$

which may be interpreted as a sort of generalised histogram, where the categories are the functions ψ_i. From this expression, the manager would simply recognize a probability of α_1 that the system would be found in the state ψ_1, a probability of α_2 that the system would be found in the state ψ_2, etc.

To help explain this, we can imagine a complex Supply Chain, like the one shown in figure 6.4, for example, inspired by the company ZARA, the trendy Spanish clothes manufacturer of La Coruña. This company, from the INDITEX group, is worldwide known as a paradigm of success, despite its owner, and major manager, Mr Ortega, the second richest person in Spain, refusing several conventional practices claimed by most schools of management. ZARA refuses, for example, the idea of advertisement.

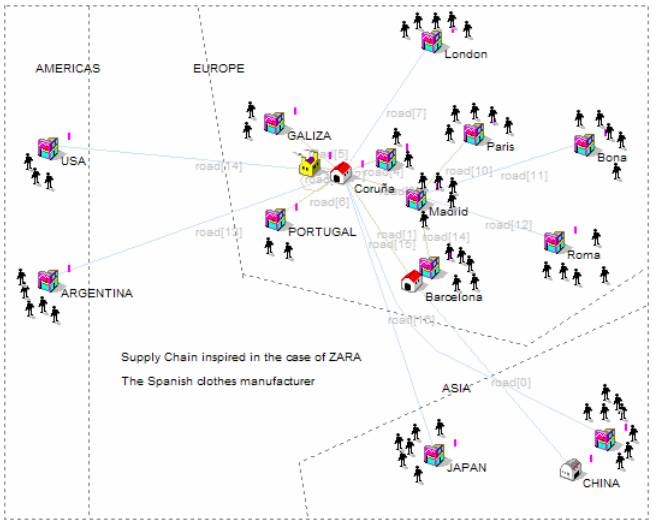

Fig. 6.4 Imaginary Supply Chain inspired by ZARA

Returning to our subject, how can we apply the concept of holistic metrics to retrieve some useful information from such a complex case[2]? First, we have to choose the ψ_i functions into which the measures will be *projected*. We may choose them in terms of some specific *conditions*, like, for example:

ψ_1 – Stockouts superior to 7%
ψ_2 – Holding costs superior to 5%
ψ_3 – Service level inferior to 75%
ψ_4 – Turnover inferior to 2

Then, while the system is simulated, the occurrences of each of these conditions, if they are true, are counted. Supposing n_j the accumulated number of occurrences of the condition related to ψ_j, and N_j the total number of samples, an estimation of α_j can simply be computed as:

[2] In this figure is represented less than perhaps 10% of the real ZARA global Supply Chain structure.

$$\alpha_j = n_j / N_j \qquad (6.2)$$

And the overall system state expressed as:

$$\psi = \alpha_1 \psi_1 + \alpha_2 \psi_2 + \alpha_3 \psi_3 + \alpha_4 \psi_4 \qquad (6.3)$$

Notice that in general *base functions* are chosen to be orthogonal, or independent of each other, but that is not a must for using this type of representation. One can also *project* a system in *non orthogonal* axis. As we said in chapter 2, such a measure of a system can be seen as a characterization of expectations under certain conditions.

6.6 Closing

Due to the itinerant nature of this work, the first pages of this text have been written in Granada, in the south of Spain, during a visit to a colleague from that University; some others have been written in Germany; still others in Alentejo, a province in the south of Portugal, admirable places for inspiration; while this last remark is already produced in Porto, my home town. Apart from the fact that almost two years have vanished since the first pages, I definitely am thankful for the capacity of modern transportation and information technology, which allowed the simultaneous shrinking and opening of space for this work to take shape.

References:

Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2004). *On Measuring the Supply Chain Flexibility.* Paper presented at the 2004 European Simulation and Modelling Conference, UNESCO, Paris, France.

Mood, A. M., Graybill, F. A., et al. (1974). Test of Hypothesis. In *Introduction to the Theory of Statistics* (pp. 401-473): McGraw-Hill.

Bibliographic References

A.A.C.A. (2004). Automotive History - Since 1960. Antique Automobile Club of America. Retrieved Nov 2004, from http://www.aaca.org/history/cars_90.htm

ABX-Logistics. (2004). ABX LOGISTICS (Deutschland) GmbH. ABX LOGISTICS. Retrieved Nov 2004, from http://www.abxlogistics.com/DE/ENGLISH/history/index.aspx

Ahn, H. J., & Lee, H. (2004). An agent-based dynamic information network for supply chain management. BT Technology Journal, 22 (2).

Arief, L. B., & Speirs, N. A. (2000). A UML tool for an automatic generation of simulation programs. Paper presented at the the Second International Workshop on Software and Performance (WOSP2000), New York.

Bagchi, S., Buckley, S. J., et al. (1998). Experience Using the IBM Supply Chain Simulator. Paper presented at the The 1998 Winter Simulation Conference.

Balci, O. (2003). Verification, Validation, and Certification of Modeling and Simulation Applications. Paper presented at the 2003 Winter Simulation Conference, USA.

Balci, O., & Saadi, S. D. (2002). Proposed Standard Processes for Certification of Modeling and Simulation Applications. Paper presented at the 2002 Winter Simulation Conference, USA.

Barjis, J., & Shishkov, B. (2001, June 26-29). UML Based Business Systems Modeling and Simulation. Paper presented at the The 4th International EUROSIM Congress, Delft, The Netherlands.

Beamon, B. M. (1999). Measuring Supply Chain Performance. International Journal of Operations and Production Management, 19 (3), 275-292.

Bell-Labs. (2002). Bell Labs Early Contributions to Computer Science. Lucent Technologies. Retrieved March 2004, from http://www.bell-labs.com/history/unix/blcontributions.html

Benisch, M., Greenwald, A., et al. (2004). Botticelli: A Supply Chain Management Agent. Paper presented at the AAMAS'04, July 19-23, New York, USA.

Berners-Lee, T. J. (1989). Information Management: A Proposal, in-house technical document. CERN, 1989. 2003, from http://www.w3.org/History/1989/proposal.html

Blumberg, D. (1999). Strategic examination of reverse logistics and repair service requirments, needs, market size, and opportunities. Journal of Business Logistics, 20 (2), 141-159.

Bodenstab, J. (2004, October 4). ToolsGroup First to Benchmark and Control Inventory Optimization - Introduces a new Parallel Simulator. ToolsGroup. Retrieved July 2005, from http://logistics.about.com/library/weekly/mpreviss.htm

Box, G. E. P., & Muller, M. E. (1958). A note on the generation of random normal deviates. Annals Math. Stat, 29, 610- 611.

Brito, A. E. S. C. (1992). The Use of CAD Techniques in Configuring Visual Interactive Simulation Models: A New Approach for Warehouse Design. Unpublished Ph.D., Cranfield Institute of Technology, Crandfield, UK.

Brito, A. E. S. C., & Feliz-Teixeira, J. M. (2001). Simulação por Computador. Porto, Portugal: PUBLINDÚSTRIA.

Bruniaux, R., & Pierreval, H. (1999). Simulating Supply Chains with System Dynamics. Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.

Bruzzone, A., & Mosca, R. (1999, October 26-28). Modelling & Simulation and ERP Systems for Supporting Logistics in Retail. Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.

Buxton, & Laski. (1962). Control and Simulation Language. Computer Journal, 5 (3).

Byrne, P. J., & Heavey, C. (2004). An Industrial Supply Chain Simulation Model. Paper presented at the Industrial Simulation Conference 2004, Malaga, Spain.

C.H.M. (2004). Time Line of Computer History: Computers. Computer History Museum. Retrieved March 2004, from http://www.computerhistory.org/timeline/timeline.php?timeline_category=cmptr

C.I.I.C. (2004). Space Industry of China. China Internet Information Center. Retrieved Nov 2004, from http://www.china.org.cn/english/SPORT-c/77126.htm

CACI. (1986). SIMFACTORY. Los Angeles: CACI Inc.

Cacioppo, K. (2000). Measuring and Managing Customer Satisfaction. Quality Digest. Retrieved June 2005, from http://www.qualitydigest.com/sept00/html/satisfaction.html

Carrie, A. (1988). Simulation of Manufacturing Systems: John Wiley & Sons.

Chandrasekaran, U., & Sheppard, S. (1987). Discrete event Distributed Systems - A survey. Paper presented at the Conference of Methodology and Validation, Orlando, Fla., USA.

Chandrashekar, A., & Schary, P. B. (1999). Toward the Virtual Supply Chain: The convergence of IT and Organization. Journal of Logistics Management, 10 (2), 27-37.

Chandy, K. M., & Misra, J. (1979). Distibuted Simulation: A Case Study in Design and Verification of Distributed Programs. IEEE Trans. Software Eng., SE-5, 440-452.

Chapman, R. G. (2005). LINKS Supply Chain Management Simulation: LINKS-simulations.com.

Chatfield, D. C., Harrison, T. P., et al. (2004). XML-based Supply Chain Simulation Modeling. Paper presented at the 2004 Winter Simulation Conference, USA.
Chen, Y., Peng, Y., et al. (1999). A Negotiation-based Multi-agent System for Supply Chain Management. Paper presented at the ACM Autonomous Agents Workshop on Specifying and Implementing Conversation Policies, Seattle.
Christopher, M. (1992). Logistics and Supply Chain Management: Strategies for Reducing Costs and Improving Services (second ed.). London: Pitman Publishing.
Christopher, M. (1994). Logistics and Supply Chain Management. Financial Times.
Chryssolouris, G. (1992). Manufactoring Systems, theory and practice: Springer-Verlag.
Clay, G. R. (2000). Venture Launch: Use of Simulation to Support Strategic Operational Decisions. Paper presented at the 2000 Winter Simulation Conference, USA.
CNN. (2001, December 10, 2001). China's long march to WTO entry. 2004, from http://archives.cnn.com/2001/WORLD/asiapcf/east/09/18/china.wto.timeline/
Crooks, K. (2002). eM-Plant (Power Point Presentation to BMW): Tecnomatrix.
Curran, C. (1991). Integrated Supply Chain Information Systems: The Next Phase after EDI? Logistics Information Management, 4 (1).
DAIMI. (2003). UML + CPN @ Nokia. DAIMI - University of Aarhus. Retrieved January 2005, from http://www.daimi.au.dk/CPnets/UMLCPNatNokia/
Davidrajuh, R. (2000). A Petri Net Approach for Performance Measurement of Supply Chain in Agile Virtual Enterprise: Narvik Institute of Technology, Narvik, Norway.
Davies, R., & O'Keefe, R. (1989). Simulation Modelling With Pascal: Prentice Hall.
DoD/US. (1997). HLA Overview. Defense Modeling and Simulation Office (DMSO), DoD/USA. from http://www.dmso.mil/dmso/docslib/
DoD/US. (2002). DoD Modeling and Simulation (M&S) Verification, Validation and Accreditation (VV&A) (Directive No. 5000.61): Department of Defense of USA.
Dong, M. (2001). Process Modeling, Performance Analysis and Configuration Simulation in Integrated Supply Chain Network Design. Unpublished Ph.D., Virginia Polytechnic Institute and State University, Blacksburg, Virginia.
Dutta, D. (2002). Retailer @ the speed of fashion (ZARA case study): http://www.3isite.com.
E.U. (2004). Gateway to the European Union: the history of European Union. European Union. Retrieved Nov 2004, from http://europa.eu.int/abc/history/index_en.htm
Edelstein, H. (2001). Pan For Gold In The Clickstream. InformationWeek.com.
Edelstein, H. (2003). Using Data Mining to Find Terrorists. Data Mining Review.
Eilon, S., Wantson-Gandy, C. D. T., et al. (1971). Distribution Management: Mathematical modeling and practical analysis: Griffin-London.
Ellram, L. M. (1991). Supply Chain Management: The Industrial Organization Perspective. International Journal of Physical Distribution & Logistics Management, 21 (1), 13-22.
Everett. (2001). Generating Gaussian Random Numbers. Taygeta Scientific Incorporated. 2004, from http://www.taygeta.com/random/gaussian.html
Eymann, T., Padovan, B., et al. (1998, July 2-8). Simulating Value Chain Coordination with Artificial Life Agents, accepted Poster. Paper presented at the 3th International Conference on Multi-Agent Systems (ICMAS'98), Paris.
Feigin, G., An, C., et al. (1996). Shape Up, Ship Out. OR/MS Today (Operations Research and the Management Sciences) Online Edition, 23.
Feliz-Teixeira, J. M., & Brito, A. E. S. C. (1999). Visual C++ Software for Warehouse Simulation (an overview). Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.
Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2003). An Approach for Dynamic Supply Chain Modelling. Paper presented at the 2003 European Simulation and Modelling Conference, Naples, Italy.
Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2004). On Measuring the Supply Chain Flexibility. Paper presented at the 2004 European Simulation and Modelling Conference, UNESCO, Paris, France.
Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2005a). Comparing a Standard and a Naïve Stock Refill Policies by Means of Simulation. Paper presented at the 2005 European Simulation and Modelling Conference, University of Porto, Porto, Portugal.
Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2005b). Using Simulation to Analyse the Procurement of Addictives for Lubricants in the Oil Refinery of Porto, Portugal. Paper presented at the Industrial Simulation Conference 2005, Berlin, Germany.
Feliz-Teixeira, J. M., & Brito, A. E. S. C. (2005c). Using Simulation to Analyse the Procurement of Additives for Lubricants in the Oil Refinery of Porto, Portugal. Paper presented at the Industrial Simulation Conference 2005, Berlin, Germany.
Feliz-Teixeira, J. M., Brito, A. E. S. C., et al. (2004). Distributed Application for Supply Chain Management Training. Paper presented at the Industrial Simulation Conference 2004, Malaga, Spain.
Fiddy, E., Bright, J. G., et al. (1981). SEE-WHY: Interactive Simulation on the Screen. In Proceedings of the Institute of Mechanical Engineers C293/81, I. Mech. E (pp. 167–172). London.
Flood, M. M. (1956). The travelling salesman problem. Ops. Res., 4, 61-75.
Forrester, J. W. (1960). Industrial Dynamics. Cambridge, MA: MIT Press.
Forrester, J. W. (2004). Origin of System Dynamics. System Dynamics Society. Retrieved March 2004, from http://www.systemdynamics.org/DL-IntroSysDyn/orig_f.htm
Fowler, M., & Scott, K. (1999). UML Distilled: Applying the Standard Object Modeling Language: Adddison Wesley Longman, Inc.
Fox, M. S., Chionglo, J. F., et al. (1993). The Integrated Supply Chain Management System: Dep. of Industrial Engineering, University of Toronto, Canada.

Fujimoto, R. M. (1998). Time Management in the High Level Architecture: College of Computing, Georgia Institute of Technology, Atlanta.
Furey, T. (2003). Pump Up the Pipeline: How to cure anemic revenue growth (Report). Bethesda: MarketBridge Corp.
G.S.D.L. (2003). Beefeater Restaurants Microworld: Global Strategy Dynamics Limited.
GalpEnergia. (2003). A Galp Energia - História. Galp Energia. 2004, from http://www.galpenergia.com/
Gan, B. P., Liu, L., et al. (2000). Distributed Supply Chain Simulation Across Enterprise Boundaries. Paper presented at the 2000 Winter Simulation Conference.
Gattorna, J. (1983). Handbook of Physical Distribution Management (3th ed.): Gower Publishing Company Ltd.
Goldman, S., Nagel, R., et al. (1995). Agile Competitors and Virtual Organizations. New York: Van Nostrand Reinhold.
Goldratt, E. M., & Fox, R. E. (1986). The Race: North River Press.
Gonçalves, J. F. (1999). Gestão de Aprovisionamentos (revista ed.). Porto, Portugal: PUBLINDÚSTRIA.
Gordon, G. (1961). A general purpose systems simulation program. Paper presented at the Eastern Joint Computer Conference, Washington, D.C.
Graham, M., & Wavish, P. R. (1991, June). Simulating and Implementing Agents and Multiple Agent Systems. Paper presented at the European Simulation Multi-Conference, Copenhagen.
Grubbström, R. W. (1995). International Logistics Management Game (ILMG). Linköping Institute of Technology, Sweden.
Guedes, A. P. (1994). An Integrated Approach to Logistics Strategy Planning Usig Visual Interactive Modelling and Decision Support. Unpublished Ph. D., University of Cranfield, U. K.
Harris, F. W. (1913). How Many Parts to Make at Once. Factory: The Magazine of Management, 10, 135-136.
Harrison, A., Christopher, M., et al. (1999). Creating the Agile Supply Chain. Cranfield, UK: School of Management, Cranfield University.
Heizer, J., & Render, B. (2001). Operations Management: Prentice Hall.
Herring, C. (1990). MODSIM: A New Object Oriented Simulation Language. Paper presented at the Multi-conference on Object-Oriented Systems, San Diego, CA.
Hewitt, F. (2001, May 2001). After Supply Chains, Think Demand Pipelines. Supply Chain Management Review, 5, 28.
Hill, D. R. C. (2004). Grid Computing and Stochastic Distributed Simulation. Paper presented at the 2004 European Simulation and Modelling Conference, Paris, France.
Hitchings, G. G. (1969). Analogue Techniques for Optimal Location of a Main Facility in Relation to Ancillary Facilities. International Journal of Production Research, 7 (3), 189-197.
Hopp, W. J., & Spearman, M. L. (2001). Factory Physics (second ed.): Irwin McGraw-Hill.
Howell, F., & McNab, R. (1998, Jan 1998). simjava: a discrete event simulation package for Java with applications in computer systems modelling. Paper presented at the First International Conference on Web-based Modelling and Simulation, San Diego CA, USA.
Hurrion, R. D. (1976). The design, use and required facilities of an interactive visual computer simulation language to explore production planning problems. Unpublished PhD, University of London.
Jensen, K. (1992). Coloured Petri nets. Basic Concepts, Analysis Methods and Practical Use (Vol. 1: Basic concepts). Berlin: Spinger-Verlag.
Kaplan, R. S., & Norton, D. (1992, (January-February 1992)). The Balanced Scorecard: Measures that Drive Performance. Harvard Business Review, 70, 71-79.
Kemers, J., & Merkuryev, Y. (1999). Simulation Application for Logistics Strategy Planning in the Baltic Market. Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.
Kim, K., Jr., B. C. P., et al. (2000). Agent-based Electronic Markets for Project Supply Chain Coordination: Department of CEE and Networking Research Center, Stanford University.
Kinnersley, B. (1995). Collected information on about 2350 computer languages, past and present (online document). Lawrence, KS 66045: Computer Science Department, University of Kansas.
Kleinrock, L. (1975). Queueing Systems (Vol. I). New York: Wiley.
Knemeyer, A. M., Ponzurick, T. G., et al. (2002). A qualitative examination of factors affecting reverse logistics systems for end-of-life computers. International Journal of Physical Distribution & Logistics Management, 32 (6), 455-479.
Kristensen, L. M., Christensen, S., et al. (1998). The practitioner's guide to colouredPetri nets: Springer-Verlag.
Kuehn, W. (2005). Blended Learning Applying Interactive Discrete Event Simulation in a Web-based Learning Management System. Paper presented at the Industrial Simulation Conference 2005, Berlin, Germany.
Kulick, B. C., & Sawyer, J. T. (2000). The Use of Simulation Modeling for Intermodal Capacity Assessment. Paper presented at the 2000 Winter Simulation Conference, USA.
Lackner, M. R. (1964). Digital Simulation and System Theory, System Development Corporation. Santa Monica, CA.
Lambert, D. M., & Pohlen, T. L. (2001). Supply Chain Metrics. The International Journal of Logistics Management, 12 (1), 1-19.
Landeghem, H. V., & Bobeanu, C.-V. (2003). Using Meta-Modelling to Process Petri Nets models of Supply Chains. Paper presented at the The 2003 European Simulation and Modelling Conference, Naples, Italy.
Lau, J. S. K., Huang, G. Q., et al. (2002). Web-based simulation portal for investigating impacts of sharing production information on supply chain dynamics from the perspective of inventory allocation. Integrated Manufacturing Systems, 13 (5), 345-358.
Lau, L. (1997). Dynamic Simulation Through the WWW. Paper presented at the Australian World Wide Web Technical Conference, Brisbane, Queensland, Australia.

Law, A. M., & Kelton, W. D. (1991). Simulation Modeling & Analysis (second ed.): McGraw-Hill International Editions.
Lee, Y. T., LcLean, C., et al. (2001). A Preliminary Information Model for Supply Chain Simulation: National Institute of Standards and Technology, Gaithersburg, USA, and Musashi University, Nerima-ku, Tokyo, Japan.
Lehoucq, R. (2005). Contagem Decrescente. Le Monde Diplomatique (portugese version), 11.
linuxHPC. (2003). Universal Weather & Aviation Selects PSSC Labs' Linux-Based Beowulf Clusters. LinuxXHPC.org. Retrieved Jan 2005, from http://www.linuxhpc.org/stories.php?story=03/07/18/3255917
Little, M. C., & McCue, D. L. (1994). Construction and Use of a Simulation Package in C++. C User's Journal, 12 (3).
Maersk-Logistics. (2004). Maersk Logistics: Our history. Maersk Logistics. Retrieved Nov 2004, from http://www.maersk-logistics.com/sw17894.asp
Mason-Jones, R., Naim, M. M., et al. (1997). The Impact of Pipeline Control on Supply Chain Dynamics. The International Journal of Logistics Management, 8 (2), 47-61.
Mason-Jones, R., & Towill, D. R. (1999). Using the Information Decoupling Point to Improve Supply Chain Performance. Journal of Logistics Management, 10 (2), 13-24.
McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys (5), 115-133.
Metz, P. J. (1998). Demystifying Supply Chain Management. MIT-Industry Integrated Supply Chain Management Program. 2003, from http://www.econ.cbs.dk/organizations/logistik/images/myst.htm
Microsoft. (2000). MSDN Library Visual Studio (Version 6.0): Microsoft.
Microsoft. (2003). The Little Chip That Will Change Your Supply Chain Forever: Microsoft Business Solutions.
Minegishi, S., & Thiel, D. (2000). System dynamics modeling and simulation of a particular food supply chain. Simulation Practice and Theory (8), 321-339.
MIT-FSCI. (2002). MIT Beergame. MIT Forum for Supply Chain Innovation. Retrieved Dec 2004, from http://beergame.mit.edu/default.htm
Mood, A. M., Graybill, F. A., et al. (1974). Test of Hypothesis. In Introduction to the Theory of Statistics (pp. 401-473): McGraw-Hill.
Müller, G., Eymann, T., et al. (1999). Economic Risks of Market-based Supply Chain Coordination Using Multi-agent Systems. Albert-Ludwigs-UniversitätFreiburg: Proposal to the SPP "IntellignetSoftwareagenten und BetriebswirtschaftlichAnwendungsszenarien" of DeutcheForschungsgemeinschaft, Institut für Informatik und Gesellschaft.
Müller, R. (2000). Scientific Models: Their Historical and Philosophical Relevance. In I. D.-D. J. Commission (Ed.), The 13th International Conference on History and Philosophy of Science. University of Zurich.
N.H.C. (2000). Logistics & Support Activities, 1950-1953. DEPARTMENT OF THE NAVY - NAVAL HISTORICAL CENTER, WASHINGTON NAVY YARD, WASHINGTON DC 20374-5060. Retrieved March 2004, from http://www.history.navy.mil/photos/events/kowar/log-sup/log-sup.htm
Nance, R. E. (1993, April 20-23). A History of Discrete Event Simulation Programming Languages. Paper presented at the The second ACM SIGPLAN conference on History of programming languages, Cambridge, Massachusetts, United States.
Nikolic, D. M., Panic, B., et al. (2004). Bullwhip Effect and Supply Chain Modelling and Analysis using CPN Tools. Paper presented at the The Fifth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, Department of Computer Science, University of Aarhus.
Nwana, H. S. (1996). Software Agents: An Overview. Knowledge Engineering Review, 11, 1-40.
Nyquist, H. (1928). Certain topics in telegraph transmission theory. Trans. AIEE, 47, 617-644.
O.B. (2005). What does SKU mean?: OnlineBusiness.com. Retrieved March 2005, from http://onlinebusiness.about.com/od/faqa/f/SKU.htm
Ohno, T. (1978). Toyota seisam hoshikiti. Tokyo: Diamond.
Ohno, T. (1988). Toyota Production System: Beyond Large-Scale Production. Cambridge, MA: Productivity Press.
OMG. (1999). Unified Modeling Language Specification, Version 1.3: Object Management Group (OMG).
OR/MS. (1996). Supply Chain Simulation Model. OR/MS Online Edition, Lionheart Publishing, 23 (2).
Orlicky, J. (1975). Meterial Requirements Planning: The New Way of Life in Production and Inventory Management. New York: McGraw-Hill.
Orsoni, A., Bruzzone, A. G., et al. (2003). Framwork Development for Web-based Simulation Applied to Supply Chain Management. International Journal of Simulation, 4 (1).
P.N.W. (2004). Welcome to the Petri Nets World. Petri Nets World. 2004, from http://www.daimi.au.dk/PetriNets/
Pardoe, D., & Stone, P. (2004). Agent-Based Supply Chain Management: Bidding for Customer Orders. Paper presented at the AAMAS'04, July 19-23, New York, USA.
Paynter, H. M. (1961). Analysis and Design of Engineering Systems. Cambridge, MA, USA: MIT Press.
Petri, C. A. (1962). Fundamentals of a Theory of Asynchronous Information Flow (pp. 386-390): IFIP Congress.
Pidd, M. (1984). Computer Simulation in Management Science. New York: John Wiley.
Pidd, M. (1992). Computer Simulation in Management Science. Chichester, England: John Wiley & Sons.
Ping, G. B., Turner, S. J., et al. (2001). Distributed Parallel Simulation of Supply Chain Models (SIMTech Technical Report No. MIT/01/029/MAPS): Singapore Institute of Manufacturing Technology.
Polczynski, J., Moroz, M., et al. (2004). Inventory Flexibility to Demand Based on the "Cranfield Blocks Game" (a simulation study) (students report). Porto, Portugal: GEIN-FEUP.
Porras, J., & Ikonen, J. (1999). Approaches to the Analysis of Distributed Simulation. Paper presented at the 11th European Simulation Symposium, Erlangen, Germany.

PRTM. (2000). U.S. Technology Industry Hits 5.4 Inventory Turns Per Year. Management Consultants.
Ritchie, L., Burnes, B., et al. (2000). The benefits of reverse logistics: the case of the Manchester Royal Infirmary Pharmacy. Supply Chain Management: An International Journal, 5 (5), 226-233.
Rizzoli, A. E. (2004). A Collection of Modelling and Simulation Resources on the Internet. Andrea Emilio Rizzoli, IDSIA, Switzerland. Retrieved Dec 2004, from http://www.idsia.ch/~andrea/simtools.html
Robinson, S. (2004). Discrete-event simulation: from the pioneers to the present, what next? Journal of the Operational Research Society, 1-11.
Saw, R. (2002). Cranfield Blocks Game: Centre for Logistics and Supply Chain Management (CSCM), University of Cranfield, UK.
Schrage, M. (2000). Serious Play: How the World's Best Companies Simulate to Innovate. Boston, USA: Harvard Business School Press.
Schruben, L. (1983). Simulation Modeling with Event Graphs. Communications of the ACM, 26 (11), 957-963.
Schuelke, C. (2001). Data Warehousing Horizons: Best Practice Approaches to Operational, Tactical and Strategic Reporting. DM Review Magazine.
Sethi, A. K., & Sethi, S. P. (1990). Flexibility in Manufacturing: A Survey. International Journal of Flexible Manufacturing Systems, 2 (4), 289-328.
Shankland, S. (2002, November 21). InfiniBand reborn for supercomputing. from http://news.zdnet.com/2100-9584_22-966777.html
Shannon, R. E. (1975). Systems Simulation, the art and the science. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.
Shewhart, W. A. (1931). Economic Control of Quality of Manufactured Product. New York: Van Nostrand.
Silva, M. (1993). Practice of Petri nets in Manufacturing. London: Chapman & Hall.
Slack, N. (1983). Flexibility as a Manufacturing Objective. International Journal of Operations and Production Management, 3 (3), 4-13.
Small, R. D., & Edelstein, H. A. (1997). Scalable Data Mining. Two Crows Corp. Retrieved June 2005, from http://www.twocrows.com/whitep.htm
Soderquist, C., & Harris, B. (2001). Can We Prevent the Next Round of Layoffs?: At Any Rate. Retrieved July 2005, from http://www.pegasuscom.com/AAR/model2.html
SOLE. (1999). SOLE - The International Society of Logistics: Logistics Bibliography. SOLE - The International Society of Logistics. Retrieved Nov 2004, from http://sole.org/bibliography.asp
Stephens, J. (2003). Custodians of Memory: The Lost Figures of May '68. Australian Review of Public Affairs.
Sterman, J. D. (1989). Modeling Managerial Behaior: Misperceptions of Feedback in a Dynamic Decision Making Experiment. Management Science, 35 (3), 321-339.
Sterman, J. D. (1992). The Beer Distribution Game: an Annotated Bibliography Covering its History and Use in Education and Research. Cambridge, MA: Sloan School of Management, Massachusetts Institute of Technology (MIT).
Stipp, D. (2005). The Pentagon's Weather Nightmare (Article): FORTUNE (online edition) http://www.fortune.com/fortune/technology/articles/0,15114,582584-1,00.html.
Sudra, Taylor, S. J. E., et al. (2000). Distributed Supply Chain Simulation in GRIDS. Paper presented at the Winter Simulation Conference, USA.
Sussams, J. E. (1992). Logistics Modelling. London, UK: Pitman Publishing.
Swaminathan, J. M., Smith, S. F., et al. (1998). Modeling Supply Chain Dynamics: A Multiagent Approach. Decision Sciences, 29 (3).
Taylor, F. W. (1903). Shp Management. Transations of the ASME, 24, 1337-1480.
Taylor, S. J. E., Fujimoto, R., et al. (2002). Distributed Simulation and Industry: Potentials and Pitfalls. Paper presented at the 2002 Winter Simulation Conference.
Taylor, S. J. E., Sudra, R., et al. (2002). GRIDS-SCF: An Infrastructure for Distributed Supply Chain Simulation. SIMULATION, 78 (5), 312-320.
Tecnomatix. (1999). Customer Success: ATS Automation-France. Tecnomatix Technologies Ltd. Retrieved Dec 2004, from http://www.tecnomatix.com/showpage.asp?page=521
TheHistoryPlace. (1999). The Vietnam War, Seeds of Conflict. The History Place. Retrieved April 2004, from http://www.historyplace.com/unitedstates/vietnam/index-1945.html
Tibben-Lembke, R. S. (2002). Life after death: reverse logistics and the product life cycle. International Journal of Physical Distribution & Logistics Management, 32 (3), 223-244.
Tocher, K. D. (1963). The Art of Simulation. London: The English Universities Press.
Towill, D. R. (1992). Supply Chain Dynamics - The Change Engineering Challange of the Mid-90's. Institute of Mechanical Engineers on Engineering Manufacture, 233-245.
Townsend, M., & Harris, P. (2004). Now the Pentagon tells Bush: climate change will destroy us. The Guardian Newspapers Limited (online) (February 22, 2004).
Tranouez, P., Lerebourg, S., et al. (2003). Changing the Level of Description in Ecosystem Models: an Overview. Paper presented at the The 2003 European Simulation and Modelling Conference, Naples, Italy.
Travers, M. D. (1996). Programming with Agents: new metaphors for thinking about computation. Unpublished Ph.D., Massachusetts Institute of Technology (MIT).
Turner, S. J. (2001). A Generic Architecture for Large-Scale Distributed Simulations (PowerPoint presentation). Singapore: UKSim.
UN. (2004). United Nations Department of Economical and Social Affairs, Division for Sustainable Development. United Nations. Retrieved Dec2004, from http://www.un.org/esa/sustdev/natlinfo/2004survey_E.pdf
Veith, T. L., Kobza, J. E., et al. (1998). World Wide Web-based Simulation. Int. J. Engng Ed., 14 (5), 316-321.

Vijayan, J. (2001, MAY 07, 2001). Inventory Turns. COMPUTERWORLD. 2005, from http://www.computerworld.com/industrytopics/retail/story/0,10801,60182,00.html
Vossebeld, R. (2002, May 2002). Simplifying the Supply Chain. Logistics Manager, 25-26.
Waller, A. P., & Ladbrook, J. (2002). Experiencing virtual factories of the future. Paper presented at the the 2002 Winter Simulation Conference, IEEE, Piscataway, NJ.
Weber, A. (1909). Uber den Standort der Industrien: Tuebingen.
Weisstein, E. W. (1999). ErrorPropagation. From MathWorld - A Wolfram Web Resource. Retrieved 2004, from http://mathworld.wolfram.com/ErrorPropagation.html
Whitin, T. M. (1953). The Theory of Inventory Management. Princeton, NJ: Princeton University Press.
Wikipedia. (2004a). 1950's Events and Trends. Wikipedia, the free encyclopedia. Retrieved Mar 2004, from http://en.wikipedia.org/wiki/1950s
Wikipedia. (2004b). Empowerment. Wikipedia, the free encyclopedia. Retrieved Nov 2004, from http://en.wikipedia.org/wiki/Empowerment
Wikipedia. (2004c). Globalization. Wikipedia, the free encyclopedia. Retrieved Dec 2004, from http://en.wikipedia.org/wiki/Globalization
Wikipedia. (2005). Quantum computer. Wikipedia, the free encyclopedia. Retrieved Jan 2005, from http://en.wikipedia.org/wiki/Quantum_computer
Williams, J. L. (2003). Oil Price History and Analysis. WTRG Economics. Retrieved Nov 2004, from http://www.wtrg.com/prices.htm
Wilson, G. (1994). The History of the Development of Parallel Computing. PARALLEL.RU. Retrieved Nov 2004, from http://parallel.ru/history/wilson_history.html
Zeigler, B. P. (1976). Theory of Modelling and Simulation. New York, NY: John Wiley and Sons.
Zeigler, B. P., Kim, D., et al. (1999). Distributed Supply Chain Simulation in a DEVS/CORBA Execution Environment. Paper presented at the Winter Simulation Conference, USA.
Zobel, R. N. (2001). A Personal History of Simulation in the UK and Europe. I. J. of Simulation, 1 (1-2), 69-79.

APPENDIX 1 235
C++ SIMULATOR CLASSES 235
CLASS <AREA>: 235
CLASS <CSU>: 235
CLASS <MATERIAL>: 240
CLASS <ENCOMENDA>: 240
CLASS <GRAPHDLG>: 241
CLASS <NODO>: 242
CLASS <PRODUCT>: 243
CLASS <PRODUCTBOX>: 243
CLASS <PRODUCTPLAN>: 245
CLASS <SIMENTITY>: 246
CLASS <SIMEVENT>: 247
CLASS <SIMMOVIE>: 247
CLASS <SIMTIMEMARK>: 247
CLASS <SIMULATOR>: 247
CLASS <SPECTRUM>: 248
CLASS <TRAMO>: 249
CLASS <VEICDATA>: 249
CLASS <VEICULO>: 250
CLASS <VIA>: 251

Appendix 1

C++ Simulator Classes

Class <Area>:

```
class Area : public SimEntity
{
	public://ATRIBUTOS:
		COLORREF		m_color;
		UINT			m_nPenWidth;
		FPoint			m_pfCentro;
		SHORT			m_selectON;
		CRect			m_rectBound;
		CArray<FPoint, FPoint> m_fPoint;

	public: //MÉTODOS:
					Area();
		void			Bound();
		virtual BOOL		Draw( CDC* pdc );
		void			Set(FPoint pti, FPoint ptf);
		void			Move(View* pView, FPoint pti, FPoint ptf);
		virtual void		MoveTo(View* pView, FPoint ptf);
		virtual void		RotateAlfa(float teta, float sinal, float dx, float dy);
		virtual void		Translation(float dx, float dy);
		virtual void		CopyTo(Area* pElem );
		virtual void		Serialize( CArchive& ar );

	DECLARE_SERIAL(Area)
};
```

Class <CSU>:

```
class CSU : public Area
{
 public: //ATRIBUTOS:

 public: //MÉTODOS:
				CSU();
				CSU(UINT tipo);//construtor público
	virtual BOOL		Draw(CDC* pdc);//Desenha o CSU
	virtual void		CopyTo(CSU* pcsu, UINT info);//Copiador de CSUs
	virtual void		Serialize(CArchive& ar);//Serializador

	virtual BOOL		Executa(UINT event, void* ppp=NULL);//Encaminhador dos eventos
	virtual void		e_SimShow(void* ppp=NULL);//Visualizador do movimento
	virtual void		e_SimNewDay(void* ppp=NULL);//um novo dia
	BOOL			 e_OrderNow(void* ppp=NULL);//encomenda aos suppliers
	BOOL			 e_OrderArrive(void* ppp=NULL);//Chegada de uma encomenda do cliente
```

```
        BOOL            e_OrderIn(void* ppp=NULL);//início de processamento das encomendas na entrada
        BOOL            e_MaterialIn(void* ppp=NULL);//Chegada de uma encomenda do supplier
        BOOL            e_EnterOutput(void* ppp=NULL);//entra actividade de Output do Stock
        BOOL            e_StartOutput(void* ppp=NULL);//inicia actividade de Output do Stock
        BOOL            e_EndOutput(void* ppp=NULL);//fim actividade de Output do Stock
        BOOL            e_StartMarsh(void* ppp=NULL);//inicia actividade de Marshaling
        BOOL            e_EndMarsh(void* ppp=NULL);//fim actividade de Marshaling
        BOOL            e_StartLoad(void* ppp=NULL);//inicia actividade de Load
        BOOL            e_EndLoad(void* ppp=NULL);//fim actividade de Load
        BOOL            e_StartUnload(void* ppp=NULL);//inicia actividade de Unload
        BOOL            e_EndUnload(void* ppp=NULL);//fim actividade de Unload
        BOOL            e_VeicAtHome(void* ppp=NULL);//fim actividade de Unload
        BOOL            e_EnterProduction(void* ppp=NULL);//entra actividade de Produção
        BOOL            e_StartProduction(void* ppp=NULL);//fim actividade de Produção
        BOOL            e_EndProduction(void* ppp=NULL);//fim actividade de Produção

        int             CSUManager(float time, UINT event, void* pent, void* ppp=NULL);
        void            SimDraw();//desenha o CSU na simulação (com indexes)
        void            ComputePathToDestiny(Encomenda* penc);//cria itinerário para a encomenda
        int             ComputeOrderQuant(ProductBox* pbox, Material* material=NULL, float maxTime=99999.9f);
        int             ComputeProductionQuant(ProductPlan* pplan, int wantQuant=0);//calcula quantidade a produzir
        float           rndNormal(float med, float dp);//distribuiçao normal do csu
        UINT            FindPath(Nodo* pnodo, Nodo* pnodofim);
        float           ComputePrice(ProductBox* pbox);//retorna o preço actual (venda)
        void*           SelectSupplier(ProductBox* pbox);//selecciona um fornecedor
        bool            InspectEncomenda(Encomenda* penc);//calcula npals e volume
        void            Stockout(float time, Encomenda* penc, float qstockout, float valstockout);
        int             QuantOrdered(ProductBox* pbox);//quantidade de material já encomendado
        int             QuantWaiting(ProductBox* pbox);//quantidade de material à espera
        void            Clean();
        void            DoNewMeasures(float time);

public:
        CPtrList        m_fleet;//frota do CSU (só para runtime)
        CTypedPtrList<CObList, ProductBox*> m_stock;//stock do CSU
        CTypedPtrList<CObList, ProductPlan*> m_prodSection;//produção do CSU
        Nodo*           m_nodo; //ligação runtime ao seu nodo...
        CPtrList        m_suppliersList; //Lista de runtime dos possíveis suppliers
        CArray<CString, CString>    m_suppliersYes; //ref dos suppliers realmente utilizados
        BOOL            m_useStock;
        BOOL            m_useProduction;
        BOOL            m_useBackorders;
        float           m_margin; //Margem do negócio (%)
        float           m_marginDp;
        float           m_infoDelay;//delay da informação
        float           m_infoDelayDp;
        float           m_pastBelieve;//para definir a forma como faço as médias
        float           m_lastTimeStock;//última vez que se mexeu no stock (para custos posse)
        BOOL            m_useSTEP;

        //listas de espera.....................................................
        CPtrList        m_inorderList;//lista entrada de ordens (encomendas) no CSU
        CPtrList        m_waitOutputList;//lista de ordens à espera de output
        CPtrList        m_waitMarshList;//lista de espera pelo marshaling
        CPtrList        m_waitLoadList;//lista de espera pelo load
        CPtrList        m_waitMaterialList;//lista de espera por material encomendado
        CPtrList        m_waitProductionList;//inorders waiting production
```

```
CPtrList        m_waitProdList;//orders waiting production
CPtrList        m_orderToLoadList;//lista de espera para encomendas a saírem
CPtrList        m_freeVeicList;//lista de veículos free
CPtrList        m_workingVeicList;//lista de veículos em trabalho
CPtrList        m_inputBaysList;//cais de input
CPtrList        m_outputBaysList;//cais de output
CPtrList        m_veicWaitUnloadList; //lista de wait por unload (antes dos cais)
CPtrList        m_toOrderList; //lista de encomendas que serão realizadas em management


//listas de estados.....................................
CPtrList        m_onOutput;//em estado de output
CPtrList        m_onEnterOutput;//em estado de enter output
CPtrList        m_onMarsh;//em estado de marshaling
CPtrList        m_onLoad;//em estado de load
CPtrList        m_onEnterLoad;//em estado de enter load
CPtrList        m_onUnload;//em estado de unload
CPtrList        m_onEnterProduction;//em estado de enter produção
CPtrList        m_onProduction;//em estado de produção

CPtrList        m_auxList; //lista auxiliar cálculos

int             m_totalOrders;//número de encomendas ao supplier (acumulado)
float           m_Qin;//paletes entradas no stock
float           m_Qout;//paletes saídas do stock
float           m_Qenergy;
float           m_QenergyDias;
float           m_refStockIndex;//para desenhar index do stock na simulação
float           m_actStockIndex;//idem...
float           m_prevStockIndex;//idem...
float           m_avrgStockIndex;//...
float           m_actProdIndex;//para index de produção na simulação
float           m_prevProdIndex;//idem...
float           m_refCostsIndex;//para index de custos na simulação
float           m_varCostsIndex;//idem...
float           m_fixedCostsIndex;//idem...
float           m_prevCostsIndex;//idem...
float           m_actIncomesIndex;//para index de incomes na simulação
float           m_prevIncomesIndex;//idem...
float           m_avrgSatisfaction;
float           m_actHoldingCosts;//custo total de posse actual (por unidade tempo)
float           m_avrgDelivCosts;
float           m_avrgBuyingCosts;
float           m_avrgDelivTime;
float           m_avrgServingTime;
float           m_avrgSupplyTime;
float           m_avrgTimeArrive;//tempo médio entre arrivals
float           m_avrgTimeClientOk;
float           m_avrgmesure1;
float           m_avrgmesure2;
float           m_avrgmesure3;
float           m_actStockTransit;
int             m_accIn;//para acumular o material que entra durante um prazo de entrega
int             m_accOut;//para acumular o material que sai durante o mesmo prazo
float           m_lastTimeArrive;//última vez que entrou material
float           m_classif;//classificação, segundo o mesmo critério que usei no jogo do dick
int             m_accDirectSells;//quantidade de material directamente servido de stock
```

```
//float         m_selldirectCosts;//para o turnover final, custos do produto vendido directamente de stock


SPECTRUM      run_stock;//stock local
SPECTRUM      run_stocktransit;//stock em trânsito
SPECTRUM      run_satisfaction;//satisfação deste cliente

SPECTRUM      run_deliveringcosts;//para guardar resultados (runtime)
SPECTRUM      run_stockingcosts;//para guardar resultados (runtime)
SPECTRUM      run_producingcosts;//para guardar resultados (runtime)
SPECTRUM      run_buyingcosts;//para guardar resultados (runtime)
SPECTRUM      run_holdingcosts;//para guardar resultados (runtime)
SPECTRUM      run_stockoutcosts;//para guardar resultados (runtime)
//SPECTRUM    run_cashflow;//para guardar resultados (runtime)
//SPECTRUM    run_totalcosts;//para guardar resultados (runtime)

SPECTRUM      run_accDeliveringcosts;//para guardar resultados (runtime)
SPECTRUM      run_accStockingcosts;//para guardar resultados (runtime)
SPECTRUM      run_accProducingcosts;//para guardar resultados (runtime)
SPECTRUM      run_accBuyingcosts;//para guardar resultados (runtime)
SPECTRUM      run_accHoldingcosts;//para guardar resultados (runtime)
SPECTRUM      run_accStockoutcosts;//para guardar resultados (runtime)

float         m_accDeliveringcosts;//(runtime)
float         m_accStockingcosts;//(runtime)
float         m_accProducingcosts;//(runtime)
float         m_accBuyingcosts;//(runtime)
float         m_accHoldingcosts;//(runtime)
float         m_accStockoutcosts;//(runtime)

SPECTRUM      run_stockouts;//para guardar resultados (runtime)
SPECTRUM      run_notserved;//para guardar resultados (runtime)
SPECTRUM      run_demand;//para guardar resultados (runtime)
SPECTRUM      run_arrived;//para guardar resultados (runtime)
SPECTRUM      run_avrgDelivTime;//para guardar resultados (runtime)
SPECTRUM      run_avrgServingTime;//para guardar resultados (runtime)
SPECTRUM      run_avrgSupplyTime;//para guardar resultados (runtime)
//SPECTRUM    run_avrgFleetTimeUsage;//para guardar resultados (runtime)
//SPECTRUM    run_avrgFleetVolumeUsage;//para guardar resultados (runtime)

//testes sobre o conceito de agilidade
SPECTRUM      run_mesure1;//(servingTime/supplyTime)
SPECTRUM      run_mesure2;//(servingTime*buyingCosts/supplyTime*deliveringCosts)
SPECTRUM      run_mesure3;//(servingTime*buyingCosts/supplyTime*deliveringCosts)

//caracterização final do CSU:
SPECTRUM      end_totalcosts;//custos totais
SPECTRUM      end_varcosts;//custos totais var
SPECTRUM      end_incomes;//incomes totais
SPECTRUM      end_buyingcosts;//custos purchase totais
SPECTRUM      end_holdingcosts;//custos posse totais
SPECTRUM      end_stockingcosts;//custos aglomerados de stocking
SPECTRUM      end_deliveringcosts;//custos aglomerados de delivering
SPECTRUM      end_producingcosts;//custos aglomerados de produção
SPECTRUM      end_stockoutcosts;//custos aglomerados de stockouts

SPECTRUM      end_turnover;//turnover
```

```
SPECTRUM        end_servlevel;//service level
SPECTRUM        end_stockoutratio;//stockout ratio
SPECTRUM        end_orderinratio;//inverso do tempo entre arrival de orders
SPECTRUM        end_agility;//(?)flexibility?
//SPECTRUM      end_avrgtimeserve;//tempo médio para servir
//SPECTRUM      end_avrgtimesupply;//tempo médio demorado pelos abastecimentos

CSU_Dlg*        m_dlg;//ligação runtime ao diálogo, quando activo.

//Atributos referentes aos processos de INPUT/OUTPUT do Stock:
CString         m_infuncFilename;
CString         m_outfuncFilename;
float           m_inputRate;
float           m_inputRateDp;
float           m_loadTime;
float           m_loadTimeDp;
float           m_outputRate;
float           m_outputRateDp;
BOOL            m_useInputFromFile;
BOOL            m_useOutputFromFile;
float           m_unloadTime;
float           m_unloadTimeDp;
float           m_inMarshaling;
float           m_inMarshalingDp;
float           m_inFunction;
float           m_inFunctionDp;
int             m_nInputBays;
int             m_nOutputBays;
float           m_outFunction;
float           m_outFunctionDp;
float           m_outMarshaling;
float           m_outMarshalingDp;
float           m_inBayTime;
float           m_inBayTimeDp;
float           m_outBayTime;
float           m_outBayTimeDp;

//Atributos referentes aos custos Stocking:
float           m_bankLendingRate;
float           m_bankLendingRateDp;
float           m_inMarshCost;
float           m_inMarshCostDp;
float           m_inputStockCost;
float           m_inputStockCostDp;
float           m_loadCost;
float           m_loadCostDp;
float           m_stockFixedCosts;
float           m_stockFixedCostsDp;
float           m_outMarshCost;
float           m_outMarshCostDp;
float           m_outputStockCost;
float           m_outputStockCostDp;
float           m_unloadCost;
float           m_unloadCostDp;
float           m_manageCosts;
float           m_manageCostsDp;
```

```
        //Atributos referentes aos custos Fixos de Produção:
        //Note-se que os custos dos processos de produção
        //estão já associados aos ProductionPlan, dependendo
        //assim do produto que se quer produzir. Aqui se
        //refere somente os Custos Fixos da produção, tais
        //como occupancy, management, personel, etc.

        float           m_productionFixedCosts;
        float           m_productionFixedCostsDp;
        float           m_productionLabourCosts;
        float           m_productionLabourCostsDp;

        //Atributos referentes à constituição da frota:
        veicData m_vans;
        veicData m_smallTrucks;
        veicData m_normalTrucks;
        veicData m_doubleTrucks;
        veicData m_trains;
        veicData m_boats;
        veicData m_airplanes;
        veicData m_xtunnel;//tem sempre um!

        float           m_fleetFixedCosts;
        float           m_fleetFixedCostsDp;

 protected:
        HICON           m_icon;//icon que representa o CSU

 DECLARE_SERIAL(CSU)
};
```

Class <Material>:

```
class Material
{
 public:
        CString         m_prodName;//nome do produto
        CString         m_prodRef;//referência do produto
        int             m_quantity;//how much, product units
        int             m_containerUnits;//units per container
        float           m_averageWait;//espera média pelo produto
        float           m_prodCost;//colocado pelo supplier
        ProductBox*     m_pbox;//apontador para um prodBox
        int             m_stockState;//estado em stock
        CPtrList        m_palList;//lista de paletes

        Material(){m_stockState = MAT_STOCKYES; m_prodCost = 0.0f; m_averageWait = 0.0f; m_quantity = 0;}
};
```

Class <Encomenda>:

```
class Encomenda
{
 public:
```

```
	CSU*		m_csuOrigin;
	CSU*		m_csuDestiny;
	CPtrList	m_materialList;
	float		m_timeToBeCool;//em dias
	float		m_timeToRefuse;//em dias
	float		m_timeOrdered;//momento em que foi pedido
	int		m_totalPal; //número total de paletes na encomenda
	float		m_volume; //em containers
	CPtrList	m_nodePath; //para conter o path de nodos até ao destino
	int		m_endProdQuant;//runtime, quantidade do produto final (produção)
	void*		m_pplan; //usado para conter um apontador para um pplan
	void*		m_pveic; //a apontar para o veículo que deve levar a encomenda
	float		m_varCost; //custo final (definido pelo supplier)
	float		m_setupCost;//custo fixo de encomendar (setup)
	BOOL		m_wasWaiting;//TRUE se não for atendida directamente de stock
	BOOL		m_okInspected;//TRUE se já foi inspeccionada

	Encomenda(){m_totalPal=0; m_varCost=0.0f; m_setupCost=0.0f; m_wasWaiting=FALSE;}
	bool		TryToServe(float time, void* penc);//tenta servir uma encomenda a partir desta
	bool		RemoveMaterial(void* mat);//remove uma linha de material e actualiza os vários parametros
	bool		AddMaterial(void* mat);//adiciona uma linha de material e actualiza os vários parametros
	void		Trash(void* csu);//delete encomenda
};
```

Class <GraphDlg>:

```
class GraphDlg : public CDialog
{
// Construction
public:
			GraphDlg(CWnd* pParent = NULL);   // standard constructor
	void		Histogram(CDC* pdc, CRect* prect);
	void		Graph(CDC* pdc, CRect* prect);
	SPECTRUM*	m_spectrum;
	CPtrList	m_spectrumList;//lista de spectrums
	SimEntity*	m_owner;
	int		m_graphType;
	CPoint		m_po;
	CPoint		m_pf;
	float		m_xpixel;
	float		m_ypixel;
	int			m_bold;

// Dialog Data
	//{{AFX_DATA(GraphDlg)
	enum { IDD = IDD_GRAPH1};
		// NOTE: the ClassWizard will add data members here
	//}}AFX_DATA


// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(GraphDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL
```

```
// Implementation
protected:

	// Generated message map functions
	//{{AFX_MSG(GraphDlg)
	afx_msg void OnPaint();
	afx_msg void OnSize(UINT nType, int cx, int cy);
	afx_msg void OnClose();
	afx_msg void OnMenuMore();
	afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
	afx_msg void OnMenuGraphDimxy();
	afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
	afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
	afx_msg void OnMouseMove(UINT nFlags, CPoint point);
	afx_msg void OnMenuGraphBold();
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};
```

Class <Nodo>:

```
class Nodo : public Area
{
 public:		//ATRIBUTOS:
	UINT		m_tipoNodo;//Tipo de nodo (NORMAL, SEMAFORO, CSU)
	float		m_inputTime;//Tempo médio de criação de veículos (só para ENTRADA)
	float		m_inputTimeDp;//desv.padrão do m_inputTime
	int		m_nServico;//número do servico deste nodo (run time)
	float		m_raio;//raio do nodo em metros...
	CPtrList	m_veiculoList;//lista de veículos... (não serializa)
	CPtrList	m_veicOutList;//lista de veículos para out... (não serializa)
	CPtrList	m_materialList;//lista de encomendas à espera de transporte (não serializa)
	UINT		m_semaforo;//0-VERDE ou 1-VERMELHO
	CSU*		m_csu; //CSU que corresponde ao nodo, se não for NULL
	float		m_unloadTimePal;//unload time per palete in a TRANSFER node
	float		m_loadTimePal;//load time per palete in the new vehicle of a TRANSFER node
	float		m_totalChargePal;//total charge per palete in a TRANSFER node
	float		m_pauseTime;//pause time in a SEMAFORO node
	float		m_pauseCharge;//pause charge in a SEMAFORO node
	BOOL		m_useByCycle;//para os nodos Transfer
	BOOL		m_useByTrigger;//para os nodos Transfer
	BOOL		m_useManagement;//idem
	float		m_daysCycle;//idem
	float		m_daysCycleDp;//idem
	veicData	m_roads;//veículos para roads
	veicData	m_trains;//veículos trains
	veicData	m_boats;//veículos boats
	veicData	m_airplanes;//veículos airplanes


	CTypedPtrList<CObList, Via*> m_inViaList;//lista de vias de input no Nodo (serializável)
	CTypedPtrList<CObList, Via*> m_outViaList;//lista de vias de output no Nodo (serializável)

 public:		//MÉTODOS:
			Nodo(UINT tipo, float r);//Construtor público
```

```
        virtual BOOL        Draw(CDC* pdc);//desenha a mesa com a palete...
        virtual void        Serialize(CArchive& ar);//Serializador
        virtual void        CopyTo(Nodo* pnodoTo);

        ~Nodo();

 protected:
        Nodo();//construtor privado

        //MÉTODOS RELACIONADOS COM A SIMULAÇÃO:
  public:
        virtual BOOL        Executa(UINT event, void* ppp=NULL);//Encaminhador dos eventos
        virtual void        e_SimShow(void* ppp);//Visualizador do movimento
        BOOL                e_Start(void* ppp);
        BOOL                e_Arrive(void* ppp);
        BOOL                e_Cycle(void* ppp);
        void                GiveViaTo(Via* pvia, Veiculo* pveic);
        Via*                GetNextRandomVia();

 DECLARE_SERIAL(Nodo)
};
```

Class <Product>:

```
class Product : public SimEntity
{
        public:     //ATRIBUTOS:
                    CString             m_name; //nome do produto na cadeia
                    CString             m_ref; //referência do produto na cadeia
                    UINT                m_type; //tipo do produto na cadeia (R, P ou 2P)
                    COLORREF            m_color; //cor do produto na cadeia
                    float               m_typLoadUnloadTimePal; //tempo de Load/Unload típico
                    float               m_typLoadUnloadTimePalDp; //e desvio padrão
                    float               m_To;//tempo de creação da geração
                    float               m_Vo;//valor inicial da geração
                    float               m_VoDp;
                    float               m_dTLife;//tempo de vida da geração
                    float               m_dTLifeDp;
                    float               m_dTNext;//evento da próxima geração
                    float               m_dTNextDp;
                    int                 m_containerUnits;//quantas unidades cabem num contentor
                    CArray<int,int>     m_subprodQuant; //quantidades de cada subrod. (9,3,0,1,0,0,4..etc)

        public: //MÉTODOS:
                                        Product();
                    virtual void        Serialize(CArchive& ar);
                    virtual BOOL        Executa(UINT event);//encaminhador de eventos

                    //evento das novas gerações:
                    BOOL                e_newGeneration();

        DECLARE_SERIAL(Product)
};
```

Class <ProductBox>:

```
class ProductBox : public SimEntity
{
    public:    //ATRIBUTOS:
               int                m_maxStock; //Stock máximo (pal)
               int                m_initialStock; //Stock inicial (pal)
               int                m_actStock; //Stock actual (pal)
               BOOL               m_useHistory; //para usar dados historicos
               CString            m_historyFile; //ficheiro dos dados historicos
               BOOL               m_useByLevel; //TRUE se for para encomendar assim
               int                m_levelToOrder; //Nível para encomendar (pal) (runtime)
               int                m_rop; //level to order... (antes)Stock de segurança no caso de ser por nível (pal)
               BOOL               m_useByCycle; //TRUE se for para encomendar assim
               float              m_daysCycle; //Ciclo de dias para a encomenda
               float              m_daysCycleDp; //desvio padrão
               float              m_daysCycleStart; //início do ciclo
               BOOL               m_useKanban;//byCycle & byLevel
               int                m_baseOrderQuantity; //quantidade base a encomendar (só multiplos disto)
               int                m_actOrderQuantity; //runtime, quanto realmente encomendar...
               int                m_orderQuantityDp; //desvio padrão
               BOOL               m_useDifToInitialStock; //usar dferença para o stock inical
               BOOL               m_useTrackDemandX;//política de encomenda segue a demand no tempo
               BOOL               m_useTrackDemandY;//política de encomenda segue a demand na Quantidade
               BOOL               m_useEOQ;//usa a EOQ e adapta-se-lhe
               BOOL               m_useImediatOrder; //se a encomenda é feita no momento
               BOOL               m_useOrderManagement; //se a encomenda é gerida a outro nível
               BOOL               m_useDemandSteps;//usa steps random na demand
               float              m_supplierRespTime; //tempo de entrega do supplier
               float              m_supplierRespTimeDp; //tempo de entrega do supplier dP
               CArray<int,int>    m_suppliersIndex; //que suppliers do CSU utiliza (0,1,4..etc)
               float              m_actPrice; //preço actual (para venda, depende do CSU)
               float              m_actCost; //preço no supplier (depende deste)
               float              m_coolTime; //tempo aceitável para satisfação do cliente (dias)
               float              m_coolTimeDp; //em dias
               float              m_avrgSuppDelay;//tempo médio de espera (dias) pela entrega (runtime)
               float              m_avrgDemand;//demand média (dinâmica)(runtime)
               float              m_avrgTimeDemand;//tempo médio entre duas demands
               float              m_avrgEOQ;//segue a média da EOQ
               float              m_lastTimeDemand;//para média entre as demands
               float              m_lastTimeSupply;//para calcular demand acumulada
               float              m_lastTimeProduce;//para calcular demand acumulada
               float              m_lastTimeStock;//última mexida no stock
               float              m_lastTimeAsked;//última vez que foi pedido (diferente de encomendado!)
               float              m_accDemandSupply;//demand acumulada entre supplies
               float              m_accDemandProduce;//demand acumulada entre produções
               int                m_acc0;//acumulador 0
               int                m_acc1;//acumulador 1
               float              m_accHoldingCost;//acumulação do custo de posse (para calcular EOQ)
               float              m_accDemand;//acumulação da demand (para calcular EOQ)
               float              m_demandPZE;//demand no prazo de entrega do supplier (runtime)
               float              m_oneOrderCost;//custo de colocar uma encomenda
               float              m_oneOrderCostDp;//...
               ProductPlan*       m_pplan;
               int                m_qBase;//demand base (último customer), para steps na demand
               float              m_mxStep;//factor multiplicativo máximo para os demandSteps.

               SPECTRUM           run_stock;//para guardar resultados (runtime)
```

```
        //SPECTRUM      run_stockouts;//só dá stockouts para a encomenda, para já.
        SPECTRUM        run_production;//para guardar resultados (runtime)
        SPECTRUM        run_demand;//para guardar resultados (runtime)
        SPECTRUM        run_suppdelay;//para guardar resultados (runtime)
        SPECTRUM        run_accdemand0;//para guardar resultados (runtime)
        SPECTRUM        run_EOQ;//economical order quantity (runtime)

        //Material*     m_material; //para encomendar material

        //----------------
        CSU*            m_csu; //meu CSU (runtime)

        CPtrList        m_suppliers; //lista de suppliers (CSU*) (runtime)
        Product*        m_product; //ligação ao produto na cadeia (runtime)

    public: //MÉTODOS:
                        ProductBox();
        void            CopyTo(ProductBox* pbox);
        virtual void    Serialize(CArchive& ar);
        float           HoldingFactor();
        int             UpdateStock(float time, int quant, void* ppp=NULL);
        void            OrderByFile(FILE* file, void* psup);
        BOOL            AskMaterial(float time, int q);
        void            Clean();


    //MÉTODOS RELACIONADOS COM A SIMULAÇÃO:
        virtual BOOL    Executa(UINT event, void* ppp=NULL);//Encaminhador dos eventos
        BOOL            e_Cycle(void* ppp=NULL);
        BOOL            e_askMaterial(void* ppp=NULL);

    DECLARE_SERIAL(ProductBox)
};
```

Class <ProductPlan>:

```
class ProductPlan : public SimEntity
{
    public:     //ATRIBUTOS:
        ProductBox*     m_pbox; //ligação em runtime ao Stock
        BOOL            m_useHistory; //para usar dados historicos
        CString         m_historyFile; //ficheiro dos dados historicos
        BOOL            m_useByLevel; //TRUE se for para encomendar assim
        BOOL            m_useByCycle; //TRUE se for para encomendar assim
        BOOL            m_useDemand; //TRUE se a produção for Demand-driven
        int             m_levelToProduce; //Nível para produzir no caso de ser por nível (pal)
        float           m_lotsDayRate; //Ciclo de dias para a encomenda
        float           m_lotsDayRateDp; //desvio padrão
        float           m_daysCycleStart; //início do ciclo
        int             m_lotQuantity; //quantidade a produzir
        int             m_prodQuantity; //quantidade a produzir (runtime)
        int             m_lotQuantityDp; //desvio padrão
        BOOL            m_useDifToInitialStock; //usar dferença para o stock inical
        BOOL            m_useImediatProduction; //se a produção é feita no momento
        BOOL            m_useProductionManagement; //se a produção é gerida a outro nível
```

```
        float           m_productionTime; //tempo unitário de produção
        float           m_productionTimeDp;
        float           m_productionCost; //custo unitário da produção
        float           m_productionCostDp;

        Material*       m_material; //para encomendar material

        CSU*            m_csu; //meu CSU (runtime)

    public: //MÉTODOS:
                        ProductPlan();
        void            CopyTo(ProductPlan* pplan);
        void            OrderByFile(FILE* file, void* psup);
        virtual void    Serialize(CArchive& ar);
        BOOL            AskToProduce(float time, int q);
        void            Clean();

    //MÉTODOS RELACIONADOS COM A SIMULAÇÃO:
        virtual BOOL    Executa(UINT event, void* ppp=NULL);//Encaminhador dos eventos
        BOOL            e_Cycle(void* ppp);

    DECLARE_SERIAL(ProductPlan)
};
```

Class <SimEntity>:

```
class SimEntity : public CObject
{
    public:
        BOOL            m_livre;//variável lógica para uso genérico
        CString         m_text;//texto usado como designação da entidade (título)
        CString         m_nome;//nome da entidade
        CString         m_ref;//referência da entidade
        float           m_time;//tempo da entidade na simulação
        float           m_initdayTime;//tempo da entidade na simulação
        UINT            m_estado;//estado da entidade
        Simulator*      m_pSim;//apontador para o Simulador
        UINT            m_tipo;//tipo da entidade
        float           m_x;//posição x da entidade
        float           m_y;//posição y da entidade
        int                     m_nivel;//nivel da entidade (altura)

                        SimEntity(CString nome); //Construtor público
        virtual void    Serialize(CArchive& ar); //Serializador

    protected:
        SimEntity(); //Construtor privado

    public:  //MÉTODOS RELACIONADOS COM A SIMULAÇÂO:
        virtual BOOL    Executa(UINT event, void* ppp=NULL);//Encaminhador de eventos
        virtual void    e_SimShow(void* ppp=NULL); //Visualizador do movimento
        virtual void    e_SimNewDay(void* ppp=NULL); //Marca de dia na simulação

    DECLARE_SERIAL(SimEntity)
};
```

Class <SimEvent>:

```
class SimEvent : public CObject
{
 public:
                float           m_time;//tempo do evento
                UINT            m_event;//número do evento
                SimEntity*      m_pEntity;//entidade associada ao evento
                void*           m_ppp;//apontador auxiliar de uso genérico
};
```

Class <SimMovie>:

```
class SimMovie : public SimEntity
{
        public:
                SimMovie(Simulator* pSim);//construtor...
                virtual BOOL Executa(UINT event, void* ppp=NULL);//Encaminhador de eventos

        protected:
                BOOL e_ShowON();//Resposta ao evento SIM_SHOW
};
```

Class <SimTimeMark>:

```
class SimTimeMark : public SimEntity
{
        public:
                SimTimeMark(Simulator* pSim);//construtor...
                virtual BOOL Executa(UINT event, void* ppp=NULL);//Encaminhador de eventos

        protected:
                BOOL e_SimNewDay();//Resposta ao evento SIM_NEWDAY
};
```

Class <Simulator>:

```
class Simulator : public CWnd
{
// Construction
public:
                        Simulator(View* pView);
        float           Time(){return m_time;}
        BOOL            Init();
        BOOL            Loop();
        void            Schedule(float time, UINT event, SimEntity* pEnt, void* ppp=NULL);
        void            SimMaster(float time, UINT event, SimEntity* pEnt, void* ppp=NULL);
        void            RemoveEvent(UINT event, SimEntity* pEnt);
        float           rndNormal(float med, float dp);//qualquer intervalo
        void            OutputReportToFile();
```

```
        void            FinalCalculations();
        void            LimpaTudo();
        void            LogText(CString text);

// Attributes
        CPtrList                                *m_veiculoList;
        CTypedPtrList<CObList, Nodo*>           *m_nodoList;

protected:
        void            LancaNodosVias(BOOL preview);
        void            LigaNodos();
        void            LancaFrotas();

        // Listas dos objectos físicos existentes no armazém:
        CTypedPtrList<CObList, Area*>           *m_areaList;
        CTypedPtrList<CObList, Via*>            *m_viaList;

        //Listas privadas do simulador:
        CPtrList        m_eventList; //Lista de eventos
        float           m_time; //tempo interno da simulação


public:
        CTypedPtrList<CObList, Product*>        *m_productList;
        CPtrList   m_objMovingList; //Fila de objectos em movimento
        CPtrList   m_ordersList; //Lista de objectos encomenda lançados na simulação

        float           m_stepTime; //intervalo de tempo para desenho
        float           m_endTime; //tempo de fim da simulação
        BOOL            m_usePreview; //para rodar o teste da rede
        float           m_maxCostsIndex; //para custos fixos
        View            *m_view; // view...
        int             m_seed;
        int             m_nReplications;//número de runs
        FILE*           m_fileLog;

protected:
        Document        *m_doc;  //documento...
        SimMovie        *m_simShow;
        SimTimeMark     *m_simTimeMark;

public:
        virtual ~Simulator();

        // Generated message map functions
protected:
        //{{AFX_MSG(Simulator)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
```

Class <SPECTRUM>:

```
class SPECTRUM
{
 public:
```

```
    float       LMin;//Valor mínimo no eixo X
    float       LMax;//Valor máximo no eixo X
    float       dL;  //filter step no eixo X
    float       Ymin;//Valor mínimo no eixo Y
    float       Ymax;//Valor máximo no eixo Y
    CString     NomeX;
    CString     NomeY;
    CString     NomeGraf;
    CArray<FPoint, FPoint> Data; //pontos do espectro
    short       okDone; //abs,trms
        float   mean;//mean on y
        float   stdev;//standard deviation on y
        float   total;//somatório dos valores de y (integral)
        float   ymin;//valor mínimo
        float   ymax;//valor máximo
        COLORREF color;
        void*   window;

                SPECTRUM();
        void    Calculate();
        void    Clean();
};
```

Class <Tramo>:

```
class Tramo : public CObject
{
 public:
                UINT        m_tipoTrm;//NORMAL - ESTREITO- TAPETE
                FPoint      m_pi;//Ponto de início
                FPoint      m_pf;//Ponto de fim
                float       m_comp;//Comprimento

                            Tramo();//construtor default
                            Tramo(FPoint pi, FPoint pf);//construtor...
                virtual void    Serialize(CArchive& ar);//Serializador
                BOOL        IsEqual(Tramo* ptr);

 DECLARE_SERIAL(Tramo)
};
```

Class <veicData>:

```
class veicData : public CObject
{
 public:
        int             m_nVeic;
        int             m_veicType;
        float           m_speedLoaded;
        float           m_speedUnloaded;
        float           m_speedDp;
        float           m_loadTimePal;
        float           m_unloadTimePal;
        float           m_loadUnloadTimeDp;
```

```
        float           m_driverCostHour;
        float           m_distanceCostKm;
        float           m_maxContainers;//em contentores
        //.....................
        float           m_percentLevelFull;
        float           m_percentLevel4;
        float           m_percentLevel3;
        float           m_percentLevel2;
        float           m_percentLevel1;
        float           m_unitCostFull;
        float           m_unitCost4;
        float           m_unitCost3;
        float           m_unitCost2;
        float           m_unitCost1;

                        veicData(){};
        virtual void    Serialize(CArchive& ar);

 DECLARE_SERIAL(veicData)
};
```

Class <Veiculo>:

```
class Veiculo : public Area
{
        public:
                UINT            m_sentido;//sentido de evolução no tramo actual
                UINT            m_tipoVias;//tipo de vias por onde pode seguir
                UINT            m_modelo;
                float           m_velMedia;
                Percurso*       m_actPercurso;
                Tramo*          m_actTramo;
                Tramo*          m_prevTramo;
                Tramo*          m_nextTramo;
                float           m_actLambda;
                Tramo*          m_tramoDestino;
                float           m_lambdaDestino;
                float           m_fullIndex;//[0,1] indica quanto cheio está o veículo
                float           m_batery;//Carga da bateria
                float           m_timeBatery;//Tempo de carga da bateria
                float           m_actVeloc;//velocidade horizontal actual
                Veiculo*        m_prevVeiculo;//veículo anterior (de trás)
                Veiculo*        m_nextVeiculo;//veículo posterior (da frente)
                float           m_totalTime;//tempo total de utilização (acumulado)
                float           m_startTime;//início da distribuição (desde EndLoad())
                float           m_totalKm;//total de kilómetros feitos (acumulados)
                float           m_startKm;//km ao início da distribuição (=0)
                int             m_actPos;//posição actual no Percurso
                Via*            m_actVia;//via actual onde se encontra o veículo
                float           m_startDeliveryTime;//tempo de início da delivery (desde Nodo->Start())
                UINT            m_nextEvent;
                double          m_accTimeVolume;//produto tempo x volume(%) acumulado
                float           m_lastTimeVol;//tempo da última carga ou descarga
                int             m_nTrips;//número de viagens ida-volta
                float           m_lastStart;//última vez que fez Start
```

```
                //.....................
                int             m_nodoIndex;
                CPtrList        m_nodoList;//lista de nodos por onde irá passar
                veicData* m_data;//dados referentes à supply chain...
                CPtrList        m_container;//onde irão as encomendas
                float           m_tLoadUnloadStart; //simulation (runtime)
                float           m_tLoadUnloadEnd; //idem
                void*           m_csu; //csu a que pertence o veículo
                float           m_cashToAsk;//money to ask in the end of a xtunnel

                //MÉTODOS: ************************************

                                Veiculo(veicData* newdata); //Construtor público

                //Métodos de expansão da classe Area:
                virtual void    CopyTo(Veiculo* pVeiculo);
                virtual void    MoveTo(View* pView, FPoint ptf);
                void            Show();//Representa o veículo


                void            Modelo(UINT modelo); //Cria o modelo do veículo
                int             CalcSentido(Tramo* tramo);//Calcula o sentido com que percorrer o tramo
                float           CalcTimeToStop();//Calcula o tempo para chegar ao destino
                float           CalcTimeToEnd();//Calcula o tempo para atingir o próximo tramo

 protected:
                Veiculo();//construtor default

                //MÉTODOS RELACIONADOS COM A SIMULAÇÃO: **********************
 public:
                virtual BOOL    Executa(UINT event, void* ppp=NULL);//Encaminhador dos eventos
                virtual void    e_SimShow(void* ppp=NULL);//Visualizador do movimento


private: //resposta aos eventos :
                BOOL            e_Start(void* ppp=NULL);//início do tramo
                BOOL             e_End(void* ppp=NULL);//fim do tramo
};
```

Class <Via>:

```
class Via : public Area
{
 public: //ATRIBUTOS:
        UINT            m_viaTipo;//Tipo de via: ROAD, RAIL, AIR, SEA, NULL, INFORMATION)
        float           m_chargeValue;//valor fixo a cobrar pela utilização da via
        BOOL            m_viaCharged;//se a via for paga!
        BOOL            m_chargeDestiny;//se for para pagar no destino!
        float           m_Gk;//constante de multiplicação da velocidade
        float           m_maxVel;//velocidade máxima na via (km/h)
        float           m_dtime;//delay de comunicação para vias de informação (ou dt no xtunnel)
        float           m_dtimeDp;//e respectivo desvio padrão

        Percurso m_tramoList; //Lista ordenada de tramos
```

```
        //NOTA: a lista de tramos não é serializada, é sim recalculada no
        //início de cada simulação. Isto é mais versátil e permite copiar vias
        //dando origem a diferentes listas de tramos (percursos):

        CPtrList            m_veiculoList;//lista de veículos...
        BOOL                m_nodoOK;//se já tem atribido um nodo de entrada
        Nodo*               m_nodoOut;//nodo de saída da via
        Nodo*               m_nodoIn;//nodo de entrada na via
        CArray<float, float> m_fluxoOut;//array de fluxos para as próximas vias
        float               m_comp;//comprimento da via (km) = soma dos tramos
        float               m_ocup;//comprimento que está ocupado.


    public: //MÉTODOS:
                            Via(UINT viaTipo);//Construtor público
        virtual BOOL        Draw(CDC* pdc);//Desenha a via com os cais e parques
        virtual void        CopyTo(Via* pVia);//Copiador de vias
        virtual void        Serialize(CArchive& ar);//Serializador

    protected:
                            Via();//construtor privado

    DECLARE_SERIAL(Via)
};
```

APPENDIX 2 255
WEB LINKS 255
SOME WEB LINKS ABOUT SIMULATION: 255
USEFUL WEB LINKS FOR SUPPLY CHAIN MANAGEMENT: 256
CONFERENCES, MAGAZINES, JOURNALS, DATABASES: 257
INSTITUTIONS RELATED TO SUPPLY CHAIN MANAGEMENT: 258
OTHER WEB LINKS: 259

Appendix 2

Web links

Some web links about Simulation:

http://gein.fe.up.pt/simgroup/
eSimGroup, Simulation for Complex Systems Group, for research and industrial projects (Author's group).

http://www.sosresearch.org/simulationeducation/simsoftware.html
Simulation Software for the Classroom. 200+ software packages in alphabetical order.

http://www.topology.org/soft/sim.html
Simulation software development frameworks preferably discrete-event, open source and free. by Alan Kennington.

http://www.acims.arizona.edu/PUBLICATIONS/PDF/7sysent.pdf
Object Oriented Simulation with Hierarchical, Modular Models: Selected Chapters Updated for DEVS-C++. By Bernard P. Zeigler.

http://www.omnetpp.org/
OMNeT++ is a public-source, component-based, modular and open- architecture simulation environment with strong GUI support and an embeddable simulation kernel.

http://scgwiki.iam.unibe.ch:8080/SCG/415
Useful tools for Petri Nets for PECOS Project.

http://www.cse.fau.edu/~maria/COURSES/CEN4010-SE/C10/10-7.html
Introduction to Petri Nets, by Kevin Mcleish. A Petri net is a graphical and mathematical modeling tool. It consists of places, transitions, and arcs that connect them.

http://www.informatik.uni-hamburg.de/TGI/PetriNets/
Welcome to the Petri Nets World. The purpose of the Petri Nets World is to provide a variety of online services for the international Petri Nets community.

http://www.daimi.au.dk/CPnets/
Petri nets. These Web pages present the activities of the CPN group at the Department of Computer Science, University of Aarhus, Denmark.

http://liinwww.ira.uka.de/bibliography/Distributed/dist.sim.html
Bibliography on distributed simulation. The Collection of Computer Science Bibliographies.

http://pages.cpsc.ucalgary.ca/~gomes/HTML/sim.html
Simulation Bookmarks. By Fabian Gomes.

http://www.virtc.com/
Virtual Technology Corporation. Distributed simulation being used across the DoD to provide analyses, simulation-based acquisition, system of systems test and evaluation, distributed training, and mission rehearsal, etc.

http://www.cs.mcgill.ca/~qxu2/pads.html
By Samir Das (outdated, but interesting). Parallel or distributed simulation refers to the execution of discrete-event simulation programs on a multiprocessor system or network of workstations.

http://www.simula.no/departments/scientific/publications/SC.5.Cai.2002.a
Simula - Developing Parallel Object-Oriented Simulation.

http://www.systemdynamics.org/
System Dynamics Society. System dynamics is a methodology for studying and managing complex feedback systems, such as one finds in business and other social systems.

http://www.raczynski.com/pn/simball.htm
SimBall. System Dynamics Simulation. Applications in business dynamics, ecology, feedback systems, general purpose dynamic modeling

http://www.public.asu.edu/~kirkwood/sysdyn/SDIntro/SDIntro.htm
System Dynamics Methods: A Quick Introduction.

http://charm.cs.uiuc.edu/
Parallel Programming Laboratory. Parallel Object-oriented Simulation Environment.

http://www.idsia.ch/~andrea/sim/simweb.html
Simulation tools and links. By Andrea Emilio Rizzoli (outdated, but good)

http://www.gambitcomm.com/site/products/snmp_simulator.shtml
MIMIC® SNMP Agent Simulator. Creates a network of up to 10,000 SNMP-manageable devices on one Intel-based PC or Sun Sparc.

Useful web links for Supply Chain Management:

www.sap.com
mySAP Supply Chain Management

www.capslogistics.com
Innovation

www.baan.com
iBaan for Supply Chain Management

www.manugistics.com
Supply Chain Management

www.peoplesoft.com
PeopleSoft CRM

www.epicor.com
Epicor, eFrontOffice/Clientele

http://www.barkawi.com/index_en.html
BARKAWI

http://www.infineon.com/
INFENION

http://logistics.about.com/
About, Inc

http://en.wikipedia.org/wiki/Supply_chain_management
Wikipedia. Supply chain management.

http://www.accenture.com/
Accenture

http://www.supplychaintoday.com/
SupplyChainToday

Conferences, Magazines, Journals, Databases:

http://biomath.ugent.be/~eurosis/
EUROSIS conferences

http://gein.fe.up.pt/esm2005/
2005 European Simulation and Modelling Conference

http://www.eurosim.info/index.php?id=2
EUROSIM - Federation of European Simulation Societies

http://www.scs.org/
Society for Computer Simulation

http://www.wintersim.org/
Winter Simulation Conference

http://www.liophant.org/scsc/
Summer Computer Simulation Conference

http://www.modelingandsimulation.org/
Modeling and Simulation International (SCS)

http://www.manufacturing.net/scm/
Supply Chain Management Review

http://sim.sagepub.com/
SIMULATION

http://epubs.siam.org/sam-bin/dbq/toclist/MMS
Multiscale Modeling & Simulation

http://www.medwellonline.net/jeas/fp.html
Journal of Engineering & Applied Sciences (JEAS)

http://www.sciencedirect.com/
ScienceDirect. Welcome to the world's largest electronic collection of science, technology and medicine full text and bibliographic information

http://www3.interscience.wiley.com
WileyInterscience

http://www.emeraldinsight.com/Insight/
Emerald publishes the world's widest range of management, library information science journals as well as a strong specialist range of engineering and applied science and technology journals

http://www.springerlink.com
SpringerLink

http://scholar.google.com/
Search scholarly articles

http://www.csa.com/

Cambridge Scientific Abstracts

http://www.ieee.org

IEEE

http://www.ingentaconnect.com/content/mcb/088
Journal of Enterprise Information Management

http://www.emeraldinsight.com/Insight/viewContainer.do?containerType=JOURNAL&containerId=1420
Logistics Information Management = Journal of Enterprise Information Management

http://www.logisticssupplychain.org/
The International Journal of Logistics Management

http://www.emeraldinsight.com/info/journals/bpmj/bpmj.jsp
Business Process Management Journal

http://www.emeraldinsight.com/info/journals/scm/scm.jsp
Supply Chain Management: An International Journal

http://www.engnetbase.com/

EngNetBase (e-Books)

http://www.engineeringvillage2.com/

Elsevier e-Books

Institutions related to Supply Chain Management:

http://www.fe.up.pt/si/unidades_geral.visualizar?p_unidade=19

GEIN – Faculdade de Engenharia da Universidade do Porto, Portugal

http://www.egp.up.pt/
EGP - Escola de Gestão do Porto

http://www.som.cranfield.ac.uk/som/research/centres/lscm/agilesupplychain/

The Agile Supply Chain Research Centre

http://www.som.cranfield.ac.uk/som/msclogistics/index.asp
The Cranfield MSc in Logistics and Supply Chain Management

http://www.som.cranfield.ac.uk/som/about/overview.asp
Cranfield School of Management, UK

http://www.uni-erlangen.org/
University of Erlangen-Nuremberg

http://www.cscmp.org/
Council of Supply Chain Management Professionals

Other web links:

http://en.wikipedia.org/wiki/Main_Page
Wikipedia, the free encyclopedia. Home

Wikipedia, the free encyclopedia. Author rights

http://www.aaca.org/
Antique Automobile Club of America

http://www.computerhistory.org/
Computer History Museum

www.tecnomatix.de/
TECNOMATIX

http://www.fujitsu-siemens.com/
Fujitsu-Siemens

http://www.emplant.de/simulation.html
eM-Plant

http://www.daimi.au.dk/CPNtools/
CPN Group, University of Aarhus, Denmark

http://cnls.lanl.gov/avalon/
Los Alamos National Laboratory, USA

http://www.galpenergia.com/
GalpEnergia, Portugal

Index

#define statements, 143
(r, Q) model, 61, 182
(r, Q) policies, 118
.NET, 78
“Bible” of Agents, 80
“byCycle” policies, 119
“byLevel” policies, 119
“data-driven” approach, 102
“globalized” world, 37
“intelligent” components, 80
“river without dams”, 190
“satisfaction” of the client, 111
“Two Bins” method, 192
“virtual” companies, 39
“virtual” services, 30
“virtual” world, 39
11SET2001, 37
1960s, 193
1980s, 191
3D graphics, 40
3D view, 33
3PL, 28, 38, 58, 65
About accumulated costs, 170
About costs, 170
About inventory, 170
About materials handling, 170
Accreditation, 92
accreditation institutions, 179
accumulated demand, 203
accumulated demand between supplies, 187
accumulated values, 185, 206
accumulated variable costs, 194
accuracy of the model, 89
activities, 76, 126, 139, 143
activity cycle diagrams, 76, 140
activity diagrams, 43, 76, 84
activity-scanning, 19, 77
additives, 177
additives for lubricants, 179
Afghanistan, 37
Agent-based simulation, 57
Agents, 2, 33, 42, 80, 81
Agents’ technology, 81
Agile-Manufacturing, 31
agility, 71, 88
Agility, 31, 38, 208
AI, 42, 80
AIR, 155
air traffic control, 79
airfreights, 58
airlines, 106
airplanes, 58, 121
Alhambra, 1
Amazon.com, 54
American Dream, 13
amplification of demand, 201
Anti-Globalization Movement, 32
antithetic variates, 92
AOL, 30
Apollo 11, 19
Apple Computer, 23
APPLE II, 23
Arab Oil Embargo, 21
arcs, 75
Area, 153, 155
ARENA, 32, 33, 77
ARPANET, 19, 21
Artificial Intelligence, 28, 42, 80
AskMaterial(), 163
assemblage, 63
Assemble-to-Order, 35, 63
assembly-line, 63
ATM, 39
ATO, 35, 63
Atomic Bomb, 16, 17
ATS Automation-France, 33
attractors, 92
attributes, 137
Australia, 43
automobiles, 58
AutoMod, 32, 41
automotive industry, 24
AutoPlay, 201, 205, 208
AutoPlay mode, 96, 201, 206
AutoPlay session, 207
AVALON, 43
average, 206
average demand, 187
average inventory, 198
average values, 206
average variance, 94
Average() function, 185
average/high-volume, 63
AweSim, 32
Backorder Level, 70
backorders, 125, 193, 203
Balanced Scorecard, 70
Balkans, 31
Baltic countries, 33
bank cash machines, 39
bank lending rate, 61, 182
BanKan, 60, 179, 190, 191, 200
BanKan method, 195, 198
BanKan system, 197
Barbie, 13
BARKAWI, 5
base functions, 74
base order quantity, 118
Base price, 129
base price of the products, 115
base product, 64
Base Quantity, 203
BASIC, 16, 22
beatnik generation, 17
Beefeater Restaurants Microworld, 201
Beer Game, 36, 42, 53, 75, 82, 193, 201, 202
Beetle model, 191
Beijing, 37
Beira Alta, 2
Belgrade, 31
Bell Labs, 15, 25
Benetton, 26, 58, 108
Beowulf Clusters, 43
Bergson, 44
Bergsonian inspiration, 44
Berlin, 30
Berlin Wall, 16, 27
best occupation level, 186, 187
best results, 187

Bill Gates, 22, 23
bins of materials, 191
BMW, 5, 41, 67, 78, 101
boats, 121
Boeing, 41
Bond Graphs, 3, 39
BOSS, 20
bottom-up, 5, 60
Box-Muller method, 146
Brito, 68
Bullwhip Effect, 35, 36, 75, 201
business margin, 115, 182, 200
by trigger, 157
ByCycle, 123, 125, 157, 163
ByLevel, 123, 163
C, 28
C language, 25
C++, iii, 2, 5, 25, 125, 137, 143, 148, 201
C++ prototype, 144
C++SIM, 32, 77
CAD, 29, 30, 40
cartoon time-bombs, 58, 111
cash flow, 116
Cash flow, 69
Cash-to-Cash Cycle Time, 70
causal diagrams, 75
causal loop diagrams, 29
CDROM, 25
cell-phone, 57
CERN, 27
certification, 70
chain of value, 115
Chain Simulator, 77
characterization of the demand, 182
characterization of the products, 181
chat, 28
cheap supercomputers, 43
chemical products, 58
chemicals, 58
China, 31, 37
Chinese, 34
chip, 21
classification, 203
client, 203
client application, 96, 206
client order, 76
client-serve, 203
client-server technology, 202
climate change, 34, 37
coal, 58
COBOL, 16
code-bar references, 36
Coding the simulator, 137
Cold War, 17, 26
collaborative network, 42
collections of SKUs, 166
Coloured Petri nets, 29, 41
COM, 78
COMMODORE, 22
common activities, 76
common Supply Chain structure, 103
communications, 33, 115
Complex behaviour, 92
complex systems, 201
CompuServe, 30
ComputeOrderQuant, 169
ComputePathToDestiny, 169
ComputePrice, 169
ComputeProductionQuant, 169
Computer Aided Design, 17
computer application, 201
Computer World, 18
Computer-Aided Design, 25
conceptual models, 51
conclusions, 189
concurrence, 34
confidence level, 94
configuration of each facility, 181
Connecting the CSUs, 106
conservative methods, 80
consumption, 183, 189
container, 159
Continuous, 51
continuous production, 63
continuous-like flow, 36
controller, 204
cooperation, 31, 34, 134, 200, 211
CORBA, 43
Corporation: A Global Business Simulation, 201
corrective factor, 185, 187
correlation graphs, 92
costs, 198, 199
costs figure, 188
costs of the products, 184
CPN/Tools, 41, 43
CPtrList, 143, 144, 167, 169
Cranfield, 34
Cranfield Blocks Game, 6, 53, 95, 133, 201, 202, 209
Cranfield School of Management, 95
Cranfield University, 88
CRAY, 24
cross-docking, 36
CSL, 19, 20
CSU, 104, 105, 110, 115, 121, 156
CSU class, 137
CSU entity, 166
CSU node, 108, 131, 157
CSU nodes, 104
CSU object, 125, 152
CSU of destiny, 159
CSU of origin, 159
CSU states, 125
CSU-manager, 117, 123, 126, 163, 166
CSUManager, 169
CTypedPtrList, 169
Cuban Missile Crises, 16
current event, 146
customer “satisfaction”, 196
customer profile monitoring, 73
Customer Satisfaction, 69
Customer Supplier Unit, 6
customer value, 31
customers, 53
customer-supplier exchange activity, 103, 104
customer-supplier-unit, 104, 105
customization, 36
CWinThread, 148
cybernetic toys, 39
cycles, 54
Da Vinci, 37
dangerous cargo, 58
Data Mining, 73, 92

data series, 206
data-array, 169
DDE, 41
dead activities, 76
dead states, 76, 124
Decision Analysis, 19
DecisionCraft, 82
decoupling-point, 35, 36
definition of the transport paths, 181
delivering, 104
delivery, 65, 117, 124, 200
delivery costs, 55, 109
Delivery costs, 113
delivery flexibility, 71
delivery path, 130
delivery policy, 55
delivery process, 127
DELL, 36, 56, 64, 69
demand, 183, 206
demand amplitudes, 202, 212
demand driven, 36
Demand During Lead-Time, 192, 196
demand figures, 54
demand pattern, 188, 200, 201
demand pipeline, 3
demand pipeline management, 36, 200
demand signal, 54
demand steps, 210, 211
Denmark, 43
dependent demand, 62
depot location, 19
depots, 202
Deutche Bahan, 107
developed world, 57
DEVS, 22
DEVS/CORBA, 33, 43
DFT, 92
DHL, 58
dialog-box, 138, 147, 181
dialog-box of properties, 173
different cultures, 25
DIGITAL, 23
digital computer, 16, 17
Digital Equipment Corp, 18
direct demand, 54
discounts, 200
discounts of quantity, 200
Discrete, 52
discrete event, 138, 139
Discrete Fourier Transform, 92
discrete simulation, 29, 137
distributed, 42, 74
distributed applications, 40
distributed computer systems, 43
distributed games, 79
distributed memory, 79
distributed objects, 78, 79, 81
distributed simulation, 24, 28, 43
Distributed Simulation Agents, 28
distributed simulation tool, 96
Distributed simulator, 94
distributed Supply Chain simulation, 201
distribution, 19, 25, 65, 113
distribution network, 65
Distribution policies, 65
distribution policy, 67
Distribution Resource Planning, 67
DLT, 61, 192, 196
DM, 73, 92
Doctoral thesis, i
Document, 144, 167
DoD, 2, 24, 42, 87
DoNewMeasures, 169
DPM, 36
Draw(), 154
driver cost per day, 106
drivers, 58
DRP, 67
Dynamic Data Exchange, 41
dynamic information systems, 81
dynamic simulation, 66, 67, 80
DYNAMO, 15
e_Cycle(), 163
e_newGeneration(), 162
e_SimNewDay(), 150
e_SimShow(), 149
eBay.com, 54
e-Book, 39
e-Business, 39, 55, 57
echelons of transport, 182
echelons of transport cost, 114, 170
e-Commerce, 31, 35, 39, 54, 57, 64, 73, 81, 88
Economical Order Quantity, 13, 60, 118, 192
ECSL, 26
EDI, 25, 27, 35, 36, 54, 55, 57
EEC, 28
EGP, 5, 96, 201, 206
Electronic Data Interchange, 25, 54
Elements of the chain, 52
email, 54, 55, 57
empiric policy, 207
eM-Plant, 41, 67, 78, 101
eM-Plant simulator, 41
Empowerment, 25
eM-Workplace, 33
Encomenda, 161
end of simulation, 146
Engineering-To-Order, 35
Enterprise Dynamics, 32
Enterprise Resource Planning, 30, 57
entities, 76, 139
environments of trust, 82
EOQ, 60, 61, 118, 192
EPICOR, 30
e-Procurement, 57
Equilibrium function, 132
Equilibrium(t) function, 209
ERASMUS, 177
ERP, 30, 35, 39, 40, 57, 67
Escola de Gestão do Porto, 201, 206
ESP, 20
ETO, 35
EU, 28, 37
EURO, 37
Europe, 28
European Economical Community, 16, 28
European Union, 28, 37, 57
EUROSIM, 18
EUROSIS, 39

event approach, 140
event graphs, 25
event handler, 146
Event handlers, 168
event method, 141, 146
event-list, 140, 144, 145, 168
events, 2, 126
events.h, 143
event-scheduling, 19, 22, 29, 77, 139
exactness, 200
Execute(), 141, 150, 168
executive, 138, 139
Executive, 144, 146
executive loop, 146
expected savings, 188
exponential smoothing, 170
EXPRESS, 26
facility location problem, 29, 66
facility of Zoom, 172
Facility related metrics, 68
factionary delivery, 67
factories, 53, 103
factory, 191, 192, 193, 197, 202
factory ship, 36
Fast Fourier Transform, 92
fast reactions, 88
FAX, 57
feedback, 37
Feliz-Teixeira, 68
FFT, 92
fill level of inventory, 118
final classification, 206
Final metrics, 170
Finance, 20
FindPath, 169
fixed cost, 55, 58
fixed lot size, 61
fixed-period revision, 62
Flash Gordon, 13
fleet, 54, 67, 117, 121, 167
fleet management, 20
Fleet related metrics, 71
Fleet-manager, 124, 126, 130
flexibility, 3, 5, 70, 71, 88, 204, 209
Flexibility, 38, 208
flexibility (matrix), 131
flexible assembly-line, 64
Flexible Manufacturing, 24
FlexSim, 77
flow of a river, 191
flow of money, 57, 116
flow rate, 75
fluorine atoms, 34
Ford W. Harris, 192
forecast errors, 70
forecast inaccuracies, 189
forecasting, 87
forecasts, 59, 64
Forrester, 15, 29
FORSIM-IV, 20
fortnightly inspection, 188
FORTRAN, 16, 28
fractionary delivery, 112
France, 37
fraud detection, 73
Frederick Taylor, 72
frequency of replenishment, 55
fuel, 58
Fujitsu, 30
Fujitsu-Siemens, 38, 64
full truck level, 186
GalpEnergia, 5, 180, 190
game, 202
game's supervisor, 208
Gantt Charts, 67
GASP, 26
GASP II, 20
general purpose primitives, 101
GENESIS, 21
GENETIK, 26
geometric factor (Gk), 155
Germany, 27, 37
GiveViaTo(), 158
Global costs, 68
global enterprise, 53
global world, 30
globalization, 31
globalization of information, 33
globalized markets, 88
Goldratts Game, 201
GoldSim, 78
GPSS, 20
GPSS/H, 26
Granada, 1
graph, 188
GraphDlg, 151, 152, 164, 170
graphic animations, 32
graphical tools, 138
graphics microprocessor, 23
Graphics User Interface, 29
Grid computing, 83
GRIDS, 43
groups of students, 202, 2004
Gyroscope Company, 18
Haeger & Schmidt GmbH, 25
Harley-Davidson, 36
Heisenberg, 33
Henry Ford, 13, 63, 72
Hewlett-Packard, 25
hidden patterns, 73
hierarchy, 39
hierarchy of decision, 58
High Level Architecture, 42
high-utilization, 63, 64
high-variety, 63
high-volume, 64
highways, 106
Hiroshima, 17
histograms, 138
historical data, 120, 183, 192
historical demand, 182
HLA, 42
HLA/RTI, 43
Ho Chi Min, 15
HOCUS, 26
holding costs, 61, 111, 184, 199
holistic, 7
holistic approach, 73
holistic metrics, 72
hot potatoes, 58, 111
how much to deliver, 66
how to deliver, 66
HTML code, 82
HTTP, 81
human genome, 34
human-on-loop, 96
Hypothesis, 4
IBM, 14, 80
IBM Research, 32
IBM Supply Chain Analyzer, 33
Icon, 167
imbalance, 70, 132, 211

imbalance period, 70
imbalance ratio, 132
inbound Logistics, 38, 65
Income, 69
incomes, 194, 198, 199
independent demand, 61, 67
independent of the inventory state, 191
INDITEX, 36
Information, 52, 57
INFORMATION, 155
information and money, 115
information flow, 57
information pipeline, 52, 88
Information Society, 18, 27, 30, 37
Information Technology, 1, 30, 54
inheritance, 39
init(), 144
initial condition, 194
initial phase shift, 119
initial stock, 196
input bays, 120
input process, 125, 126
InspectEncomenda, 169
integrated circuit, 16
interactive tool, 201
International Logistics Management Game, 201
International Quality Standard, 31
Internet, 19, 25, 28, 30, 31, 33, 38, 39, 54, 64, 80, 81, 155, 201
in-transit, 110
in-transit inventory, 111, 178
inventory, 54, 188, 206
inventory characteristics, 163
inventory in imbalance, 131
inventory in motion, 199
inventory level, 193
Inventory Management, 5, 15
inventory model, 187, 188
inventory operator, 76
Inventory policies, 60
inventory revision, 189
inventory state, 194, 196
inventory system, 28
inventory-rigidity-to-demand, 209
inverse of flexibility, 132, 195
IP address, 205
Iranian Revolution, 23
Iraq-Iran war, 23
iron, 58
ISO, 31, 92
IT, 30, 33, 38, 54
Japanese automotive industry, 191
Japanese phonetics, 198
Java, 32
JAVA applet, 82
Jewish, 13
JIC, 191
JIT, 30, 35, 36, 56, 58, 63, 64, 69, 73, 191, 212
JIT philosophy, 44
JIT system, 195
jobs, 62
Job-Shop, 62
just-in-case, 62, 191
Just-In-Time, 3, 24, 191
Kanban, 25
KANBAN, 3, 60, 62, 190, 191, 192, 194, 196, 198
KANBAN method, 193
KANBAN quantity, 192
KANBAN systems, 191
Kant, 44
key drivers, 66
Korean War, 14
La Sabika, 1
labour, 112
LAN, 38
last customer, 30, 103, 119, 134, 163, 167
Law & Kelton, 93
LCD-projector, 96
lead-time, 25, 55, 61, 118, 201
Lead-Time distance, 192
lean, 36
lean production, 36
Learning Organization, 31
Leça da Palmeira, 179
level of carbon dioxide, 34
level of occupation, 200
life-cycle, 25, 38, 129
Life-Cycle Diagram, 38
LigaSuppliers()., 169
line of material, 162
lineal Supply Chains, 193
Linear Programming, 66, 84
LINKS, 82
LISP, 16
list of input paths, 156
list of materials, 130
list of orders, 157
list of output paths, 156
list of product-boxes, 163
list of vehicles, 121, 157
lists of objects, 143
live activities, 76
live activity, 141
live state to another live state, 168
live states, 76, 124
local area network, 21
location of facilities, 66
Logistics, 11, 25, 27, 28
Logistics Management, 11
Logistics Modelling, 28
Logistics Strategy Planning, 29, 33, 66
Logistics systems, 32
LogisticsWorld.com, 11
LOGO, 16
London, 30
loop controller, 147
Loop(), 144, 146
Los Alamos National Laboratory, 43
lot size, 122, 196
lot sizes, 192
LOTUS-123, 23
love is…, 72
low local inventory, 190
low/average-variety, 63
low-utilization, 63
low-variety, 64
LP, 84
LSP, 29, 33, 66
LT, 61
Macintosh, 23

maintenance, 58
Make-to-Order, 62
Make-to-Stock, 35, 64
management, 124
management costs, 109
Management costs, 114
Management functions, 59
management levels, 59
managing, 104, 117
manned travel to Mars, 34
manufacturing, 104, 117, 124
manufacturing costs, 109
Manufacturing costs, 112
manufacturing policy, 122
manufacturing process, 112, 121 128, 178
Manufacturing Resource Planning, 24
Market Estimation Agent, 81
Marketing, 20
marshalling, 127, 166
Marshalling time, 121
marshalling zone, 120
Martorell, 191
Mass Customization, 35
Mass Customization Era, 31
Mass Production, 17, 63
Mass Production Era, 17, 31
Mass-customization, 64
material, 129
Material, 162
material “not served”, 196
Material Requirements Planning, 22
material-order, 129, 159, 162
material-orders, 128, 159, 161, 166
materials, 52
Materials, 58
materials flow, 191
materials pipeline, 52, 88
Matrix representation, 133
Matsushita, 36
Max stock input rate, 121
maximum level of inventory, 118
May 1968, 18
MEA, 81
measures, 186, 194
mechanical models, 15
medicines, 58
Mercantile, 25
merge-in-transit, 36
method Model(), 159
methods, 137
metrics, 186
Mexico, 191
Michael Pidd, 29
Micro Saint, 32
microchips, 25
microcomputer, 24, 33
Microelectronics, 18, 21
Microprocessor, 21
microprocessors, 33
MicroSaint, 41
Microsoft, 22, 28, 30, 53, 143
Microsoft dot-NET, 43
Microsoft Visual C++, 139, 143
Mikhail Gorbachev, 26
minimal order quantity, 185
minimal quantity, 192
minimum inventory, 196
minimum truck occupation, 185
MIT, 14, 36, 42, 193
mix flexibility, 71
mobile-phones, 64
mode of TRANSFER, 157
model, 51
Model consistency, accuracy, 89
model description, 44
model implementation, 44
model validation, 183
Modelling the vehicles, 130
MODSIM, 32
modular design, 63
money, 52
Money, 57
Monte Carlo, 52, 74, 80, 83
monthly historical record, 182
mother, ii
Motorola, 64
moving average, 170
moving factories, 36, 56
MRP, 22, 24, 62, 67
MRP description tree, 122
MRP II, 24, 30
MRP tree, 112
MS-DOS, 23
MS-Excel, 40, 207
MS-Excel document, 181
MSN, 30
MTO, 62
MTS, 35, 64
Multi-Agent, 33
multi-agent framework, 80
multi-drop, 67
multi-level structure, 202
multiprocessor computer, 83
multi-purpose blocks, 101
multi-purpose simulator, 101
multi-user system, 81
Munich, 67, 101
Nagasaki, 17
naive method, 190, 191, 198, 199
naive method of reordering, 179
naive policy, 195
naive solution, 195
Name, 129
nanotechnology, 34
Naples, 5
NASA, 14
NATO, 31
NEC, 15
network of transports, 104
Neural Networks, 39
new product flexibility, 71
New York, 30
next-event, 3, 140
next-generation, 129
nodes, 56
Nodes, 156
NOKIA, 43
non-visibility, 208
NORMAL, 156
Normal distributed numbers, 203
Normal distribution, 72, 94, 119
Normal function, 54
normal node, 108, 131
NORMAL node, 157
nuclear energy, 21
Number of input bays, 120
number of lots, 192
numeric analysis, 66
Nuremberg, 5

Nyquist sampling law, 86
Object Linking Embedding, 41
Object Oriented Paradigm, 16, 39
Object Oriented Programming, 3, 32, 77, 138
Object::Draw(), 154
occupancy, 71, 112
ODBC, 78
OLE, 41
on demand, 128
OnDemand, 123
one-order-cost, 120
OOP, 32, 39, 41, 77, 78, 138
OOP languages, 138
Open Database Connectivity Link, 78
operational, 3, 58, 65
operational advantages, 190
operational level, 59
operational performance, 200
operations block, 138
Operations Management, 19
Operations Research, 13
operations theatre, 153
OPS, 20
optical-fibres, 33
optimistic methods, 80
order, 129
order management, 178
order-by-catalogue, 106
OrderByFile(), 164
order-by-Internet, 106
ordering by “Internet”, 178
ordering policy, 129, 177
Ordering policy, 118
oriental cultures, 73
out-bound logistics, 65
Output complexity, 85
output graphs, 174
output process, 125
outsourcing, 53
P products, 112
P+ product, 112, 122, 178
Panalpina, 58
paper tape, 14
parallel, 42, 74
parallel hardware, 83
parallel machines, 83
parallel processing, 83
parallel simulation, 84
Paris, 30
partners, 208
PASCAL, 28
path, 155
path to the client, 159
PAUSE, 156
pause node, 108, 131
PAUSE node, 157
PC, 28
PC compatible, 30
PCs, 79
PDP-8, 18
Peace & Love, 20
PeopleSoft, 30
Perfect Order Measurement, 70
Persian Golf, 28
Personal Computer, 22, 23
Personal Computing, 23
Petri nets, 2, 19, 29, 43, 44, 75, 81, 84, 101
Philips, 41
Physical Distribution, 12, 15, 19
physical processes, 57
Physics, ii, 33
PINK FLOYD, 21
pioneers, 28
pipeline management, 56
Pipeline Management, 38
pipeline-like flow, 60
pipelines, 58, 106
pizzas, 64
places, 75
PLAY button, 206
pointers, 137
POP, 20
POP culture, 17
popularisation of technologies, 198
Porto, 179
power spent in imbalance, 132, 210
predicting, 36
prediction of the future, 87
prediction of the past, 87
preliminary results, 185
Preparing bay time, 120
Prices and costs, 109
primer supplier, 53, 56, 134, 167
Process of delivery, 127
process of inputting, 120
Process of manufacturing, 128
process of outputting, 120
Process of purchase, 125
Process of stocking, 125
process of validation, 183
process strategies, 62
Process-focus, 62
process-interaction, 19, 77
process-time, 63
procurement, 113
procurement of additives, 179
Product, 162
product cost, 109, 200
product mixes, 71
product obsolescence, 38
product price, 109, 200
product-box, 116, 120, 129, 163
Product-box, 163, 164, 170
product-boxes, 116
Product-focus, 63
Production policies, 62, 65
production rate, 122, 128
production scheduling, 63
production section, 117, 121
Production-manager, 124
product-plan, 121, 165
Product-plan, 165
product-plan objects, 167
Products, 116
Professional Manager, 20
profitable demand, 72
Progressive Rock, 21
ProModel, 26, 32
propagation of uncertainties, 93
protocol of communication, 78, 138
Puebla, 191
PULL, 35
pull philosophy, 24
PULL system, 67
pulling the materials, 191
purchase, 117, 124
purchase cost, 182

purchase process, 125
purchasing, 104
purchasing costs, 109
Purchasing costs, 109
purchasing policies, 186
purchasing process, 110
PUSH, 35
push philosophy, 30
push system, 22
push-into-stock, 191
quantity arrived, 206
quantity to order, 118, 180
Quantum Computer, 42, 45
Quantum Mechanics, 33, 74
QUEST, 32
Queueing Systems Theory, 22
Radio Frequency Identification, 53
RAIL, 155
railways, 58, 106
RAND Corporation, 15
random generated, 203
Random Normal generator, 146
random number, 138
random number generator, 93
random steps, 204
random variation, 54
ratio between incomes, 199
ration of exports, 27
raw cycle time, 128
raw materials, 56, 63
raw process time, 122
reacting, 36
real data, 184
real demand, 189
real reasoning, 3
recycling, 38
reductionism, 73
Reference, 129
refill policies, 194
refinery of Porto, 78, 177, 179
regulated by time, 200
reorder cycles, 201
reorder level, 118
Reorder Point, 61
reorder policy, 193
reorder quantity, 203
reordering policy, 191
Repetitive-focus, 63
replenishment policy, 60
replications, 81, 194, 209
representation of the facilities, 181
resolution, 85
retailer, 54, 192, 194, 195
retailers, 53, 103
Return-On-Investment, 36, 70
Reverse Logistics, 38, 163
RFI, 53
Richard Saw, 95
right-mouse-button-click, 167
rigidity, 70, 72, 131, 177, 195
Rigidity, 70, 164
rigidity matrix, 133, 177, 209
rigidity of a Supply Chain, 70, 211
risk analysis, 73
risk/benefits analysis, 80
rndNormal(), 144
ROAD, 155
road vehicles, 121
roads, 106
ROI, 36, 70
Romans, 12
rooter, 141
ROP, 61, 182
Russia, 37
Saddam, 37
safety-stock, 118, 192, 196
Sampling rates, 86
sandstone, 58
SAP, 30
satellites, 33
satisfaction, 194, 211
scale sculpts, 15
scale sculpture, 51
scaled elements of the reality, 102
scenario, 187, 188
Schedule(), 140, 143, 145
sciences of management, 201
Scientific Management, 13
SCM, 27, 34, 38, 53, 78, 81, 88, 95, 116, 201
SCM training, 177
scSimulator, 82
SEA, 155
seasonality, 54
section of management, 117
seeds, 93
SEE-WHY, 20
Sensitivity analysis, 92
sequence of events, 141
sequential objects, 77, 78, 79, 82, 84, 89, 137
Sequential objects, 76
serialization, 169
Serialize, 154
Serialize(), 143, 144
serialized, 167
server, 204
service level, 186, 194, 195, 198, 199
Service level, 69
setup cost, 61, 110, 120
shared memory, 79, 137
ships, 58
short-term scheduling, 67
ShowFlow, 41
signal-to-noise ratio, 85
SIM_NEWDAY, 150
SIM_SHOW, 149
SIMAN, 26
SIMAN/CINEMA, 26
SIMCOM, 20
SIMCSRIPT, 77
SimDlg, 147
SimEntity, 142, 149, 161, 168
SimEntity(), 143
SimEvent, 141, 144
SIMFACTORY, 26, 29
SimJava, 32
SimMovie, 149
SIMON, 26
SIMPLE, 15
Simplex Method, 13
SIMPROCESS, 29, 32
SIMSCRIPT, 16, 20, 26
SimTimeMark, 150
Simul8, 77
SIMULA, 3, 16, 20, 25, 32
SIMUL8, 41
SIMULA-P, 26
simulation, 39, 74, 194

Simulation, 4, 12, 18, 20, 28
simulation applications, 26
Simulation Dynamics Inc., 78
simulation environment, 67
simulation languages, 20, 32
simulation replications, 178
simulation strategy, 185
simulation toolbar, 174
Simulation tools, 26
simulations, 42
Simulator, 139, 143, 144, 147
simulator object, 144
simulator structure, 137
Singapore, 79
single-drop, 67
SKU, 60, 70, 130, 163, 181, 185, 207
SKU/day, 70, 132, 210
SLAM, 20
SLAM II, 26
SMTP, 81
Society of Fantasy, 45
Society of Knowledge, 45
SOL, 20
Space Conquest, 17
Space Race, 23
Spain, 43, 191
spare parts, 58
SPECTRUM, 151, 164, 170
spreadsheets, 185
standard deviation, 146, 193, 206
standard deviations charts, 208
standard logistic measures, 195, 197
standard measure, 4, 220
standard metrics, 194
state function, 74
State transition diagrams, 19, 76, 101,149
States (activities), 167
static elements, 56
Statistical, 52
statistical inventory models, 60
Statistical Sampling, 13
step in demand, 210
step of demand, 133
step on the demand, 70
step response, 133
stochastic model, 87
stochastic variable, 90, 93
stock, 54, 117
stock breaking, 111
stock in motion, 199
Stock input function, 121
Stock Keeping Unit, 60, 207
stock level, 128, 190
stock operation costs, 55
stock operation times, 55
stock refill policies, 190
stock resource, 166
stocking, 104, 117, 124
stocking costs, 109
Stocking costs, 110
stocking process, 125
stocking system, 120
Stock-manager, 124
stockout, 111, 126, 169, 206
stockout ratio, 186, 194, 195, 198, 211
Stockout ratio, 69
stockouts, 61, 132, 179, 183, 192, 194, 196, 200, 203, 207
strategic, 3, 58
strategic guidelines, 60
strategic level, 59
STRATOVISION, 66
SuperMan, 13
supplier, 196, 202, 203
suppliers, 52, 166
Supply Chain, 1, 12, 15, 27, 29, 31, 32, 34, 38, 52, 53, 56, 62, 65, 104, 194, 202
Supply Chain Builder, 78
Supply Chain coordination, 33
Supply Chain Cycle Time, 70
Supply Chain flexibility, 68
Supply Chain game, 95
Supply Chain Management, 6, 27, 34, 53, 78, 81, 88, 95, 116, 201
Supply Chain management training, 208
Supply Chain measures, 152
Supply Chain metrics, 194
Supply Chain model, 102, 125, 172
Supply Chain modelling, 74, 79, 124
Supply Chain simulation, 77
Supply Chain Simulator, 41, 133, 191
Supply Chains, 200
suspected “terrorist” detection, 73
Sustainable Development, 34
synchronism, 200
synchronization, 37
synchronizing causality, 79
system, 51
System 2250, 17
System Dynamics, 15, 36, 74, 75, 84, 101
System Dynamics Simulation, 29
system products, 116, 128, 129
System Simulation, 15
system states, 140
tactical, 3, 58, 65
tactical constraints, 60
tactical level, 59
Taiichi Ohno, 72, 190, 198
Tandy Radio Shack, 23
tasks, 62
Taylor II, 32, 41
TCP/IP, 25, 81, 94, 96, 203, 205
TCP/IP protocol, 201
Technology, 120
Tecnomatix®, 33
Texas Instruments, 15, 16
TFS, 193, 194
The Beatles, 17
The customer, 54
The factory, 55
the Moon, 19
The retailer, 54
the sophisticated and the obvious, 191
The Stones, 17
The warehouse, 55
theatre of operations, 181
Theory of Scheduling, 19
ThinAgents, 43
Third Party Logistics, 28

third world, 57
thread, 148
three-phase, 77
three-phase approach, 19
Thunderbird, 13
Tim Berners-Lee, 27
Time for Full Satisfaction, 120, 182, 193, 194
time of imbalance, 132
time synchronization, 202
time to recover the equilibrium, 210
Time(), 144
time-dependent models, 138
time-slicing waiting for the slowest, 95
tokens, 75
Tokyo, 30
top-down, 5, 124
Total and global costs, 114
total costs, 194, 199
total demand, 61
Total Quality Management, 25, 34
Total time of operation, 71
Total variable costs, 69
Toyota, 24
Toyota Production System, 25
TPG/TNT, 58
TQM, 25, 34
Traditional metrics, 67
training, 201
trains, 58, 121
TRANSFER, 156
transfer node, 108, 131
transistor, 14
transitions, 75
transparency, 38
transport cost, 113, 182, 200
transport cost echelons, 183
transport node, 67, 106
transport path, 3, 58, 130
transport resources, 58, 66
transport system, 58
transportation effects, 186, 189
travelling salesman problem, 20
Treaty of Maastricht, 28
tree of the simulation, 44
trend, 54
TRS-80, 23
truck occupation, 189
trucking, 58
Trucks, 58
trust, 34, 35, 200, 211
TSF, 182
tunnels of products, 200
turbulence, 62
turbulent markets, 31, 34
turnover, 68, 116, 194, 195, 198, 199
Turnover, 69
Twin Towers, 37
Two-Bins, 62, 190, 192
TX-0, 14
Type, 129
typical information delay, 115
Typical load/unload time, 129
UK, 79
UKSIM, 18
ultimate customers, 53
UML, 43, 44, 101
UN, 34, 37
Uncertainties, 85, 211
uncertainty, 3, 93
Unified Modelling Language, 43
UN-Iraq conflict, 28
unitary product cost,, 184
unitary stockout cost, 184
United Nations, 31, 34, 37
United States, 31
Units per container, 129
UNIVAC, 14
Universal Weather and Aviation, 43
University of Aarhus, 43
University of Cranfield, 5
UNIX, 16, 21, 23, 81
UNLOAD activity, 125, 127
Unload time, 121
unprocessed materials, 56
UpdateStock(), 163
UPS, 36, 58
USA, 87
Usage, 71
US-military, 17
USSR, 16, 23, 26
utilities block, 138
utilization of resources, 62
V2, 14
vacuum tube, 14
validating an application, 179
validation, 6, 177, 185
Validation, 91
validation of a model, 179
value of the product, 109
VANDERGRAAF, 21
variability, 202
variable costs, 58, 186, 194
variables, 137
vehicles, 106, 128, 212
Vehicles, 159
veicData, 170
Veiculo, 159
verification, 5, 177, 189
Verification, 91
vertical integration, 53
Via::Draw(), 156
Vietnam, 15, 17
Vietnam War, 17
View, 144, 154, 169, 171
VIM, 29, 32, 39
vinyl discs, 21
Virgin Atlantic, 41
Virtual Enterprise, 33, 35, 42
virtual methods, 142
virtual reality games, 40
Virtual Supply Chain, 57
virtual world, 57
VIS, 20, 23, 26, 29
visibility, 35, 208, 211
Visual and Interactive Simulation, 20
Visual and Interactive Software, 23, 26
Visual Interactive Modelling, 29, 39
Visual modelling (Area), 153
VMS, 23
Volkswagen, 36, 41, 191
volume flexibility, 72
Volume flexibility, 71
volume of production, 62
Von Braun, 17
VW, 64

Wagner-Whitin, 61
waiting, 157
WAN network, 79
warehouse, 167, 192, 193, 194
warehouses, 53, 103, 202
warm-up times, 81
waterways, 58
Wave Fourier Analysis, 74
Web, 40, 42, 74
Web Based Beer Game, 202
Web-based, 42, 57, 74, 81
Web-based simulation, 42
Web-browser, 42, 82
Websters Dictionary, 11
Web-tutorials, 82
what-if, 60, 66
when to deliver, 66
Wikipedia, 42
Win Sockets, 78
Windows, 66, 204
WINDOWS 3.0, 28
WINDOWS 95, 30
Windows application, 171
Windows event handlers, 153
Windows message, 148
WinSockets, 94
WITNESS, 26, 41
wood, 58
WordStar, 23
workstations, 79
World Trade Centre, 37
World Trade Organization, 31, 37
World War II, 13, 53
WORLD WAR II, 21
Worlds Away, 30
Worldwide Web, 27
WTO, 31, 37
WWW, 27, 28, 33, 36, 38, 39, 42
XEROX, 21
XML, 40, 82
XTUNNEL, 155
ZARA, 26, 36, 58, 107
zero defects, 24
zooming, 154