



City Research Online

City, University of London Institutional Repository

Citation: Topintzi, E. (2001). System Concepts and Formal Modelling Methods for Business Processes. (Unpublished Doctoral thesis, City, University of London)

This is the accepted version of the paper.

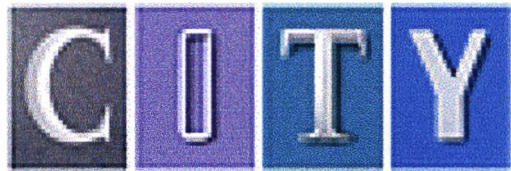
This version of the publication may differ from the final published version.

Permanent repository link: <https://openaccess.city.ac.uk/id/eprint/30844/>

Link to published version:

Copyright: City Research Online aims to make research outputs of City, University of London available to a wider audience. Copyright and Moral Rights remain with the author(s) and/or copyright holders. URLs from City Research Online may be freely distributed and linked to.

Reuse: Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.



City University
London

**SYSTEM CONCEPTS AND FORMAL
MODELLING METHODS
FOR BUSINESS PROCESSES**

BY

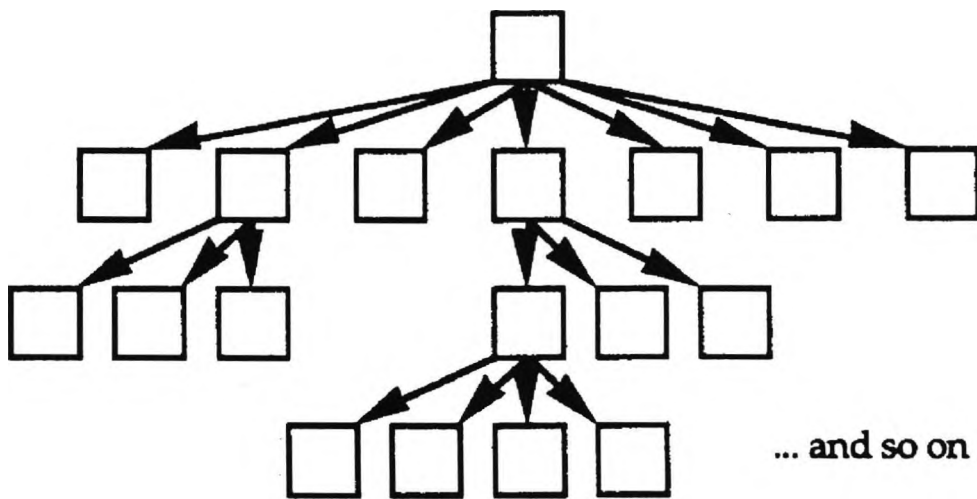
ERMIONI TOPINTZI

THESIS SUBMITTED FOR THE
AWARD OF THE DEGREE OF DOCTOR
OF PHILOSOPHY
IN
SYSTEMS ENGINEERING

CONTROL ENGINEERING CENTRE
DEPARTMENT OF ELECTRICAL, ELECTRONIC AND
INFORMATION ENGINEERING
CITY UNIVERSITY
LONDON EC1V 0HB

December 2001

SYSTEM CONCEPTS AND FORMAL MODELLING METHODS FOR BUSINESS PROCESSES



Are the world's processes really this neat and tidy?

To my beloved grandmother Nina

TABLE OF CONTENTS

| | |
|--|-------------|
| LIST OF FIGURES | IX |
| ACKNOWLEDGEMENTS | XIV |
| ABSTRACT | XVII |
| 1. INTRODUCTION..... | 19 |
| 2. PROBLEM AREA: BUSINESS PROCESSES AND EMERGING ISSUES | 29 |
| 2.1 INTRODUCTION..... | 29 |
| 2.2 BUSINESS PROCESSES | 29 |
| 2.3 BUSINESS PROCESS RE-ENGINEERING (BPR) | 35 |
| 2.4 THE INDUSTRIAL ENTERPRISE PARADIGM: PROBLEM FORMULATION AND GENERAL ISSUES | 36 |
| 2.5 STRUCTURE, BEHAVIOUR AND FUNCTION..... | 38 |
| 2.6 CLASSIFICATION OF BUSINESS PROCESSES..... | 40 |
| 2.6.1 <i>Taxonomy of Business Processes</i> | 40 |
| 2.7 RESEARCH FOCUS..... | 43 |
| 2.7.1 <i>The consulting firm's problem – a case study</i> | 45 |
| 2.8 CONCLUSION | 46 |
| 3. A SYSTEMS APPROACH IN THE CONTEXT OF GENERAL PROCESSES..... | 49 |
| 3.1 INTRODUCTION..... | 49 |
| 3.2 THE NOTION OF A SYSTEM..... | 49 |
| 3.3 THE NOTION OF OBJECTS | 50 |
| 3.4 THE NOTION OF COMPOSITE OBJECTS | 56 |
| 3.5 THE SYSTEM AND ITS ENVIRONMENT | 60 |
| 3.6 SYSTEM BEHAVIOUR, DYNAMICS AND QUALITATIVE SYSTEM PROPERTIES..... | 63 |
| 3.7 CONTROL CONCEPTS AND PRINCIPLES..... | 68 |
| 3.8 THE INTEGRATED ENTERPRISE: GLOBAL DIAGNOSTICS AND GLOBAL CONTROLLABILITY | 72 |
| 3.8.a <i>Description of the Integrated Operations</i> | 72 |
| 3.8.b <i>Modelling Classes in Process Operations</i> | 75 |
| 3.8.c <i>Organisational Issues of the Integrated Operations</i> | 76 |
| 3.8.d <i>Global Control and Measurement Issues</i> | 80 |
| 3.9 CONCLUSION..... | 84 |
| 4. GENERIC METHODOLOGIES FOR BUSINESS PROCESSES MODELLING AND DECISION MAKING: A REVIEW | 87 |

| | |
|---|-----|
| 4.1 INTRODUCTION..... | 87 |
| 4.2 MODELLING ISSUES OF THE GENERAL BUSINESS PROBLEM | 87 |
| 4.3 GENERIC METHODOLOGIES FOR BUSINESS PROCESSES MODELLING AND DECISION MAKING..... | 89 |
| 4.4 PART I: QUALITATIVE PHYSICS | 91 |
| 4.4.1 <i>Qualitative Reasoning</i> | 91 |
| 4.4.2 <i>History of qualitative reasoning</i> | 92 |
| 4.4.3 <i>Qualitative reasoning by Forbus</i> | 92 |
| 4.4.4 <i>Qualitative simulation</i> | 94 |
| 4.4.5 <i>Basic concepts</i> | 94 |
| 4.4.6 <i>Semi-Quantitative Reasoning</i> | 96 |
| 4.4.7 <i>Other representations</i> | 97 |
| 4.4.7.1 <i>Linking Petri Nets with Qualitative Reasoning</i> | 98 |
| 4.4.7.2 <i>Petri Nets as Qualitative models</i> | 99 |
| 4.4.7.3 <i>Advances</i> | 100 |
| 4.4.8 <i>Example:</i> | 100 |
| 4.4.9 <i>Dealing with complex systems</i> | 102 |
| 4.4.10 <i>Comments on Qualitative Reasoning</i> | 104 |
| 4.4.11 <i>Order-of-magnitude reasoning</i> | 105 |
| 4.4.12 <i>Basic Concepts</i> | 106 |
| 4.4.13 <i>Assignments, Constraints, and Rules</i> | 108 |
| 4.4.14 <i>O(M) Reasoning</i> | 109 |
| 4.4.15 <i>Related work</i> | 110 |
| 4.4.16 <i>Comments on Order-of-Magnitude Reasoning</i> | 111 |
| 4.5 PART II: FUNCTIONAL REASONING AND MULTILEVEL FLOW MODELLING..... | 112 |
| 4.5.1 <i>The significance of Functional Knowledge</i> | 112 |
| 4.5.2 <i>Linking Qualitative Reasoning with Functional Reasoning</i> | 112 |
| 4.5.3 <i>The Functional Representation</i> | 113 |
| 4.5.4 <i>Multilevel Flow Modelling</i> | 114 |
| 4.5.5 <i>Concepts of Goals and Functions</i> | 115 |
| 4.5.6 <i>Modelling using the MFM methodology</i> | 116 |
| 4.5.7 <i>Flow Structures</i> | 119 |
| 4.5.8 <i>Tools</i> | 121 |
| 4.5.9 <i>Comments on Multilevel Flow Modelling</i> | 122 |
| 4.6 PART III: GOAL-DRIVEN METHODOLOGIES | 123 |
| 4.6.1 <i>Modelling the intentional structure of a business process</i> | 123 |
| 4.6.2 <i>Redesigning the process</i> | 125 |
| 4.6.3 <i>A different approach on Goal Driven Business Modelling</i> | 127 |
| 4.6.4 <i>Driving Business Modelling with Goals</i> | 129 |
| 4.6.5 <i>Comments on Goal-driven methodologies</i> | 129 |
| 4.7 PART IV: BAYESIAN NETWORKS | 130 |
| 4.7.1 <i>Causal networks</i> | 131 |

| | |
|---|------------|
| 4.7.2 Bayesian networks..... | 132 |
| 4.7.3 Building models..... | 133 |
| 4.7.4 Modelling tricks..... | 134 |
| 4.7.5 Example:..... | 134 |
| 4.7.6 Junction trees..... | 135 |
| 4.7.6.1 Cluster trees..... | 135 |
| 4.7.6.2 Junction trees..... | 136 |
| 4.7.7 Actions..... | 139 |
| 4.7.8 Comments on Bayesian networks..... | 140 |
| 4.8 CONCLUSION..... | 140 |
| 5. CONCEPTUAL MODELLING AND THE SOFT SYSTEMS APPROACH..... | 144 |
| 5.1 INTRODUCTION..... | 144 |
| 5.2 THE SOFT SYSTEMS APPROACH..... | 144 |
| 5.2.1 The development of Soft Systems Approach..... | 144 |
| 5.2.2 A bird's eye view of Checkland Methodology..... | 146 |
| 5.2.3 The use of HARDY in the modelling of Soft Systems..... | 147 |
| 5.2.4 Guiding Soft Systems modelling through CommonKADS..... | 149 |
| 5.2.5 General comments on the Soft Systems Approach..... | 149 |
| 5.3 CONCEPTUAL MODELLING..... | 150 |
| 5.3.1 The mind and the origins of conceptual modelling..... | 150 |
| 5.3.2 Basic concepts..... | 152 |
| 5.3.3 Drawing the conceptual model for the consulting firm's problem..... | 158 |
| 5.3.4 General comments on Conceptual Modelling..... | 162 |
| 5.3.5 Ontologies and Conceptual Modelling..... | 164 |
| 5.4 CONCLUSION..... | 167 |
| 6. SYSTEM DYNAMICS..... | 170 |
| 6.1 INTRODUCTION..... | 170 |
| 6.2 BASIC CONCEPTS OF SYSTEM DYNAMICS..... | 171 |
| 6.3 POSITIVE AND NEGATIVE FEEDBACK..... | 172 |
| 6.4 METHODOLOGY AND APPROACHES..... | 173 |
| 6.4.1 A structured approach to System Dynamics..... | 173 |
| 6.4.2 Influence Diagrams..... | 175 |
| 6.4.3 Discussion on Simulation..... | 176 |
| 6.4.3.1 Representing delays in models..... | 178 |
| 6.4.4 Optimisation in System Dynamics..... | 179 |
| 6.4.4.1 Example:..... | 180 |
| 6.4.5 Time in System Dynamics..... | 183 |
| 6.5 TESTS FOR BUILDING CONFIDENCE IN SYSTEM DYNAMICS MODELS..... | 183 |
| 6.6 PARAMETER ESTIMATION IN SYSTEM DYNAMICS MODELLING..... | 186 |

| | |
|--|------------|
| 6.7 ASSUMPTIONS | 188 |
| 6.8 TOOLS..... | 189 |
| 6.9 DRAWING THE SD MODEL FOR THE CONSULTING FIRM'S PROBLEM | 190 |
| 6.9.1 <i>Simulating the model</i> | 195 |
| 6.10 GENERAL COMMENTS ON SYSTEM DYNAMICS..... | 203 |
| 6.11 THE DISCRETE-TIME MODEL OF THE CONSULTING FIRM'S PROBLEM..... | 205 |
| 6.11.1 <i>Dynamic Programming Implementation</i> | 209 |
| 6.12 CONCLUSION | 212 |
| 7. ROLE BASED MODELS: ROLE ACTIVITY DIAGRAMS..... | 216 |
| 7.1 INTRODUCTION..... | 216 |
| 7.2 BASIC CONCEPTS..... | 216 |
| 7.2.1 <i>Representing alternative courses of action</i> | 217 |
| 7.2.2 <i>Representing concurrent threads of action</i> | 219 |
| 7.2.3 <i>Interactions</i> | 221 |
| 7.3 MODELLING BUSINESS PROCESSES..... | 221 |
| 7.3.1 <i>Getting the "big picture" for the consulting firm's problem</i> | 222 |
| 7.4 DRAWING THE RAD FOR THE CONSULTING FIRM'S PROBLEM..... | 223 |
| 7.5 GENERAL COMMENTS ON RADs | 228 |
| 7.6 CONCLUSION | 229 |
| 8. PETRI NETS..... | 231 |
| 8.1 INTRODUCTION..... | 231 |
| 8.2 BASIC CONCEPTS OF PETRI NETS..... | 232 |
| 8.2.1 <i>Modelling with Petri Nets</i> | 234 |
| 8.2.2 <i>Subclasses of Petri nets</i> | 236 |
| 8.2.3 <i>Examples and Applications</i> | 238 |
| 8.3 EXTENSIONS OF PETRI NETS..... | 240 |
| 8.4 HIGH-LEVEL PETRI NETS AND THEIR ROLE IN THE MODELLING OF BUSINESS PROCESSES..... | 242 |
| 8.4.1 <i>Modelling business processes</i> | 244 |
| 8.4.2 <i>Triggering</i> | 245 |
| 8.5 ANALYSIS OF PETRI NETS..... | 247 |
| 8.6 TOOLS..... | 248 |
| 8.7 DRAWING THE PETRI NET MODEL FOR THE CONSULTING FIRM'S PROBLEM..... | 250 |
| 8.8 GENERAL COMMENTS ON HIGH-LEVEL PETRI NETS..... | 256 |
| 8.9 PETRI NETS AND THE OTHER METHODOLOGIES..... | 258 |
| 8.10 CONCLUSION | 262 |
| 9. CONCLUSIONS – (IDEAS FOR FUTURE WORK)..... | 264 |
| 9.1 GENERAL APPROACH OF THE THESIS | 264 |
| 9.2 FUTURE WORK..... | 269 |

| | |
|---|------------|
| REFERENCES: | 272 |
| APPENDIX | 303 |
| 1. QUALITATIVE REASONING | 303 |
| 2. ORDER-OF-MAGNITUDE REASONING..... | 308 |
| 3. MULTILEVEL FLOW MODELLING..... | 310 |
| 4. BAYESIAN NETWORKS..... | 313 |
| 5. SYSTEM DYNAMICS..... | 316 |
| 6. MATLAB PROGRAMS FOR THE DISCRETE-TIME MODEL..... | 320 |
| 7. PETRI NETS..... | 327 |
| LIST OF PUBLICATIONS | 330 |

LIST OF FIGURES

| | |
|--|-----|
| <i>Figure (2.1): Enterprise as an interconnection of business processes driven by a common goal</i> | 31 |
| <i>Figure (2.2): Systemic view of business process as an interconnection of activities driven by a particular goal, objective</i> | 32 |
| <i>Figure (2.3): Extended description of a business process [Poh & Chew, 1]</i> | 33 |
| <i>Figure (2.4): Embedding and Fractal Representation of the Overall Enterprise System</i> | 34 |
| <i>Figure (2.5): Interrelationships of the elements of a business process [Hammer & Champy, 1]</i> | 35 |
| <i>Figure (2.6): Embedding of the features of an Industrial Enterprise</i> | 37 |
| <i>Figure (2.7): Overall description of the problem area</i> | 38 |
| <i>Figure (2.8): The three types of business process ([Ould, 1])</i> | 42 |
| <i>Figure (2.9): PRIMA's Business Process Model [PRIMA, 1]</i> | 43 |
| <i>Figure (3.1): Object Environment Relation</i> | 51 |
| <i>Figure (3.2): Object orientation and the object's embedding into the environment</i> | 55 |
| <i>Figure (3.3): Objects interaction with environment representation</i> | 57 |
| <i>Figure (3.4): Composite object representation</i> | 59 |
| <i>Figure (3.5): The System and its Environment</i> | 61 |
| <i>Figure (3.6): System Overview</i> | 63 |
| <i>Figure (3.7): Open-loop Configuration</i> | 69 |
| <i>Figure (3.8): Closed-Loop, or feedback configuration</i> | 70 |
| <i>Figure (3.9): Diagram of the state estimation problem</i> | 71 |
| <i>Figure (3.10): General Control, Modelling, Estimation and Coordination Configuration</i> | 71 |
| <i>Figure (3.11): Functions for Operations of Process Plants</i> | 74 |
| <i>Figure (3.12): Overall Operational Hierarchy</i> | 79 |
| <i>Figure (3.13): Model Nesting in Process Control Hierarchy</i> | 79 |
| <i>Figure (3.14): Generic Function Model with Internal Structure</i> | 80 |
| <i>Figure (4.1): Conceptual Modelling and Development of the System ([Louc. & Zic., 1])</i> | 88 |
| <i>Figure (4.2): Modelling at quantitative levels of abstraction ([Dalle Molle et al., 1]).</i> | 94 |
| <i>Figure (4.3): Qualitative models are abstractions of differential equations ([Kuipers, 5])</i> | 95 |
| <i>Figure (4.4): Qualitative step responses for a system of two first-order processes in series ([Dalle Molle et al., 1])</i> | 96 |
| <i>Figure (4.5): The Supply and Demand curves ([Kuipers, 5])</i> | 100 |
| <i>Figure (4.6): Defining and perturbing the Supply and Demand curve equations ([Kuipers, 5])</i> | 102 |
| <i>Figure (4.7): The reference state (R) and the perturbed solution (T0) for the Supply and Demand curves, responding to increased XSF ([Kuipers, 5]).</i> | 102 |
| <i>Figure (4.8): Control of focus of attention ([Kuipers, 5])</i> | 104 |
| <i>Table (4.1): Primitive relations of the O(M) formalism ([Mavro. & Steph., 2])</i> | 107 |
| <i>Table (4.2): O(M) relations representing relations that are commonly used in engineering</i> | 108 |

| | |
|---|-----|
| <i>([Mavro. & Steph., 2])</i> | 108 |
| <i>Figure (4.9): Dimensions of representation in MFM ([Lind, 2])</i> | 115 |
| <i>Figure (4.10): Goals and functions of a circulation system ([Lind, 2])</i> | 116 |
| <i>Figure (4.11): Relations between goals, functions and equipment in the central heating system ([Lind, 2])</i> | 118 |
| <i>Figure (4.12): Flow structures for the central heating system ([Lind, 2])</i> | 120 |
| <i>Table (4.3): Elementary Acts ([VonWright, 1])</i> | 120 |
| <i>Figure (4.13): MFM model of the central heating system ([Lind, 2])</i> | 121 |
| <i>Figure (4.14): Rules expressed as activity descriptions ([Yu & Mylop., 1])</i> | 125 |
| <i>Figure (4.15): Issue Argumentation model supporting redesign of a goods acquisition process. ([Yu & Mylop., 1])</i> | 126 |
| <i>Figure (4.16): The Process Meta Model by ([Pohl et al., 1])</i> | 127 |
| <i>Figure (4.17): Integration of the functional view by into the PMM ([Jac. & Holt., 1])</i> | 128 |
| <i>Figure (4.18): Serial connection. When B is instantiated it blocks communication between A and C ([Jensen, 2])</i> | 131 |
| <i>Figure (4.19): Diverging connection. If A is instantiated, it blocks communication between its children ([Jensen, 2])</i> | 131 |
| <i>Figure (4.20): Converging connection. If A changes certainty, it opens communication between its parents ([Jensen, 2])</i> | 132 |
| <i>Figure (4.21): A Bayesian network and a corresponding cluster tree. Separators are in square boxes ([Jensen, 2])</i> | 135 |
| <i>Figure (4.22): Examples of singly connected and multiply connected DAGs ([Jensen, 2])</i> | 136 |
| <i>Figure (4.23): A singly connected DAG and its junction graph. By removing any of the links with separator F one gets a junction tree ([Jensen, 2])</i> | 137 |
| <i>Figure (4.24): Construction of a junction tree for a singly connected DAG ([Jensen, 2])</i> | 137 |
| <i>Figure (4.25): A DAG, the moral and triangulated graphs. The fill-ins are indicated by dotted lines ([Jensen, 2])</i> | 138 |
| <i>Figure (4.26): The junction graph for the triangulated graph in figure 4.25 and a junction tree ([Jensen, 2])</i> | 138 |
| <i>Figure (5.1): The advice transfer system at the highest level of abstraction ([Kingston, 3])</i> | 145 |
| <i>Figure (5.2): The advice transfer system expanded into individual processes ([Kingston, 3])</i> | 145 |
| <i>Figure (5.3): The process "Carry out work" is broken down to sub-processes ([Kingston, 3])</i> | 145 |
| <i>Figure (5.4): The top level of the Bid Management system ([Kingston, 3])</i> | 148 |
| <i>Figure (5.5): Representing the attributes of an entity</i> | 156 |
| <i>Figure (5.6): The global conceptual model for the consulting firm's problem</i> | 161 |
| <i>Figure (5.7): Part of the conceptual model of figure 5.6 that includes the time-varying objects</i> | 162 |
| <i>Figure (5.8): Structuring the competency question</i> | 167 |
| <i>Figure (6.1): The five stages approach ([Coyle, 2]).</i> | 174 |
| <i>Figure (6.2): The inquiry procedure in System Dynamics</i> | 175 |
| <i>Figure (6.3): The influence diagram of the central heating system ([Coyle, 2]).</i> | 176 |

| | |
|--|-----|
| <u>Figure (6.4):</u> The diagram of relationship between levels, rates and auxiliaries ([Coyle, 2]). | 177 |
| <u>Figure (6.5):</u> The different types of delays ([Coyle, 2]). | 178 |
| <u>Figure (6.6):</u> Influence diagram of the management's rationale for expansion into the speciality business ([Risch, Troy. & Ster., 1]) | 182 |
| <u>Figure (6.7):</u> Revised loop diagram with the unanticipated side effects on mill operations of greater product line breadth ([Risch, Troy. & Ster., 1]) | 182 |
| <u>Figure (6.8):</u> The overall system dynamics model for the consulting firm's problem | 192 |
| <u>Figure (6.9):</u> The SD model for the equivalent consultant workforce | 193 |
| <u>Figure (6.10):</u> The SD model for the hiring of Junior Consultants | 194 |
| <u>Figure (6.11):</u> Distribution of Consultants to supervisors and to those available for projects | 195 |
| <u>Figure (6.12):</u> A slider for the promo time jun to con | 195 |
| <u>Figure (6.13):</u> A first attempt to run the model | 196 |
| <u>Figure (6.14):</u> The personnel bar graph | 196 |
| <u>Table (6.1):</u> Table of values for Figure 6.13 | 197 |
| <u>Figure (6.15):</u> The best-case scenario | 198 |
| <u>Figure (6.16):</u> A closer to reality scenario | 199 |
| <u>Figure (6.17):</u> The graph of hiring when manpower discrepancy equals zero | 200 |
| <u>Figure (6.18):</u> The graph of the four levels of resources when manpower discrepancy equals zero | 200 |
| <u>Table (6.2):</u> Table of values when manpower discrepancy equals zero | 201 |
| <u>Figure (6.19):</u> The graph of the four levels of resources when manpower discrepancy is negative | 202 |
| <u>Figure (6.20):</u> The graph of hiring when manpower discrepancy is negative | 202 |
| <u>Figure (6.21):</u> The control equivalent form of the consulting firm's problem | 207 |
| <u>Figure (6.22):</u> Transition graph for a deterministic finite state system [Kerts., 1] | 210 |
| <u>Figure (7.1):</u> The state before a case refinement | 218 |
| <u>Figure (7.2):</u> The state after a case refinement in one case | 218 |
| <u>Figure (7.3):</u> The state after a case refinement in the other case | 218 |
| <u>Figure (7.4):</u> The state before a two-way part refinement | 219 |
| <u>Figure (7.5):</u> The equivalent state at the start of the part refinement | 220 |
| <u>Figure (7.6):</u> The state at the end of the part refinement | 220 |
| <u>Figure (7.7):</u> The equivalent after the part refinement | 220 |
| <u>Figure (7.8):</u> A summary RAD for the consulting firm's problem | 222 |
| <u>Figure (7.9):</u> The RAD for the HR Manager role | 223 |
| <u>Figure (7.10):</u> The RAD for the External Consultants role | 224 |
| <u>Figure (7.11):</u> The RAD for the Junior Consultants role | 225 |
| <u>Figure (7.12):</u> The RAD for the Consultants role | 225 |
| <u>Figure (7.13):</u> The RAD for the Senior Consultants role | 226 |
| <u>Figure (7.14):</u> The RAD for the consulting firm's problem | 227 |
| <u>Figure (8.1):</u> Concurrency. These two transitions can fire in any order. ([Peterson, 2]) | 235 |

| | |
|--|-----|
| <u>Figure (8.2): Conflict. Transitions t_j and t_k are in conflict since firing either will remove the token from p_1, disabling the other transition ([Peterson, 2])</u> | 235 |
| <u>Figure (8.3): An example of a state machine</u> | 237 |
| <u>Figure (8.4): An example of a marked graph</u> | 237 |
| <u>Figure (8.5): An extended free-choice net but not a free-choice one. Verify that $p_1 \bullet = \{t_1, t_2\}$ and $p_2 \bullet = \{t_1, t_2\}$, $p_1 \bullet \cap p_2 \bullet \neq \emptyset$, and $p_1 \bullet = p_2 \bullet$, but $p_1 \bullet \neq p_2 \bullet \neq 1$.</u> | 237 |
| <u>Figure (8.6): The producer / consumer problem modelled as a Petri net ([Peterson, 2])</u> | 238 |
| <u>Figure (8.7): A PERT chart of the construction of a house [Levy et al., 1]</u> | 239 |
| <u>Figure (8.8): A Petri net representation of the PERT chart of figure 8.7 ([Peterson, 2])</u> | 239 |
| <u>Figure (8.9): The basic activities (primitives) in workflow management ([van der Aalst, 4])</u> | 242 |
| <u>Figure (8.10): The four types of triggering ([van der Aalst, 6])</u> | 246 |
| <u>Figure (8.11): An example indicating the triggering mechanism ([van der Aalst, 3])</u> | 247 |
| <u>Figure (8.12): The complete Petri Net model of the consulting firm's problem</u> | 253 |
| <u>Figure (8.13): Introducing clustering in Figure 8.12</u> | 254 |
| <u>Figure (8.14): General description of the consulting firm's problem</u> | 254 |
| <u>Figure (8.15): Drawing the tree of the system</u> | 255 |
| <u>Figure (8.16): Classification of the various methodologies according to the nature of model description</u> | 259 |
| <u>Figure (9.1): Classification of the various methodologies according to the nature of model description and in the wider effort of deployment of conceptual and quantitative information</u> | 265 |
| <u>Figure (9.2): Nesting of formal modelling</u> | 271 |
| <u>Figure appendix (2.1): Strict interpretation of the relation $A r B$ ([Mavro. & Steph., 2])</u> | 308 |
| <u>Figure appendix (2.2): Constrained strict interpretation of the relation $A r B$ ([Mavro. & Steph., 2])</u> | 308 |
| <u>Figure appendix (2.3): Intervals used in the heuristic interpretation: (bottom) overlapping intervals; and (top) non-exhaustive intervals ([Mavro. & Steph., 2])</u> | 309 |
| <u>Figure appendix (3.1): MFM concepts and symbology ([Lind, 6])</u> | 310 |
| <u>Figure appendix (3.2): The symbol of decision function</u> | 311 |
| <u>Figure appendix (3.3): The symbol used for support functions</u> | 311 |
| <u>Figure appendix (3.4): The symbol used for management functions</u> | 312 |
| <u>Figure appendix (4.1): A way to introduce undirected relations between A, B, and C ([Jensen, 2])</u> | 313 |
| <u>Figure appendix (4.2): The general situation for noisy or. q_i is the probability that the impact of A_i is inhibited ([Jensen, 2])</u> | 314 |
| <u>Figure appendix (4.3): A_i is independent of $C_1, \dots, C_{i-1}, C_{i+1}, \dots, C_n$ given A and C_i ([Jensen, 2])</u> | 314 |
| <u>Figure appendix (4.4): A_1 and A_2 are divorced from A_3 and A_4 by introducing variable C ([Jensen, 2])</u> | 315 |

| | |
|---|-----|
| <i>Figure appendix (5.1): The analysis of the problem of the central heating system through the List Extension method ([Coyle, 2])</i> | 316 |
| <i>Figure appendix (5.2): The diagram of management consultants recruits ([Coyle, 2])</i> | 317 |
| <i>Figure appendix (7.1): A Petri net structure</i> | 327 |
| <i>Figure appendix (7.2) : A Petri net graph equivalent to the structure of figure 7.1 ([Peterson, 2])</i> | 328 |
| <i>Figure appendix (7.3): A marked Petri net to illustrate the firing rules. Transitions t_1, t_3, and t_4 are enabled ([Peterson, 2])</i> | 328 |
| <i>Figure appendix (7.4): The marking resulting from firing transition t_4 ([Peterson, 2])</i> | 328 |
| <i>Figure appendix (7.5): The marking resulting from firing transition t_1 ([Peterson, 2])</i> | 329 |

ACKNOWLEDGEMENTS

This thesis would not have come about without the invaluable assistance, advice, criticism, and encouragement of many colleagues, friends, and family. I owe an immeasurable debt to my supervisor Professor Nicos Karcianas who has guided me all the way from my admission to the Department of Electrical, Electronic and Information Engineering to the completion of my thesis. He has not only provided sage advice and constructive criticism, but also his probing questions, insightful observations, and knowledge of systems theory have led me to the completion of my goals. His dedication and passion in relentlessly proving to his students the challenging beauty of research have taught me that “its not the final destination that is of value, but the voyage itself”.

Only recently had I the privilege of working with a very knowledgeable academic, Dr. George Halikias whose knowledge in the field of control and dynamic programming has introduced me to invaluable literature and ideas, and has supported the development of my research.

My research as well as my life activities would not have been possible without the support of my loving family: First, my parents for their unconditional love and support when the need was there. My sister Nina for her tenderness and the many conversations we had during difficult and frustrating moments. My partner, Dr. Manos Nistazakis who without his support, love, intellectual and technical help this thesis would not have been possible. I will always be grateful to him for his constant encouragement whenever disillusionment took over.

My colleagues at the Control Engineering Centre and the Machine Vision Group of City University have given me an enjoyable working environment and have helped me immensely to overcome technical problems arising throughout this research work. Names may be unlisted here, but are very well registered in my memory. Special thanks to Tania, who has patiently shared with me the anxiety and the inevitable difficulties of long-lasting research.

Special acknowledgement is due to Colin Manders (former director at NAI Gooch Webster) for his support and advice, his ‘obsession’ with delivering high-standard work, his guidance in developing presentation skills and his help in countless other ways. Special thanks to Tasos Kyprianidis (Planet - Ernst & Young), who helped

me to understand the consulting firm's problem in depth and develop its conceptual model, Rania Vrettou (Planet - Ernst & Young) and Mamina Amourgi (Planet - Ernst & Young), who provided valuable data for my research.

Finally, thanks are due to Planet - Ernst & Young, Greece and NAI Gooch Webster, London for the financial support that they have provided.

DECLARATION

The University Librarian of the City University may allow this thesis to be copied in whole or in part without any reference to the author. This permission covers only single copies, made for study purposes, subject to normal conditions of acknowledgements.

ABSTRACT

The major quality breakthrough of the 1980s was the realisation by management that business and manufacturing processes are the key to customer service and organisational performance. This thesis is concerned with the overall problem of modelling of business processes. Of special interest is the study of business processes through an interdisciplinary approach that cuts across the boundaries of management and information technology. The overall effort is placed on being able to move from a purely conceptual level of describing a business process to a more formal one, enabling decision making, and driving the analysis away from experience, intuition, and informal debate. The extended review and presentation of the various modelling methodologies given here, serve as a guide to their basic concepts and capabilities. A particular case-study – the management of the human resources in a consulting company – has been used in this thesis to enable the evaluation of the modelling techniques. Hence, models have been produced, as well as simulation results to indicate the limitations, the advantages and the information gained. Through this application, the understanding of requirements for modelling analysis and decision making of business processes was acquired.

Particularly, two very important techniques were investigated. System Dynamics and Petri nets provide the answers when process models are geared to deliver not only qualitative but also quantitative results. However, Petri nets provide the mathematical notation and the plethora of analysis tools needed for the validation, verification, and performance analysis of the model. Additionally, two different simulation software packages were used, based on these methodologies; *Ithink*[®], which is based on System Dynamics, and *Alpha/Sim*[®], based on Petri nets theory. The model produced in the case study depicts perfectly the capabilities of the two techniques. Petri nets is not the total business modelling solution, it can be complemented by other methods, such as System Dynamics and discrete-time modelling as shown in Chapter 6. The feasibility of all these modelling techniques lies entirely on the analyst, who should use them alternately to satisfy the requirements of the problem.

Chapter 1

INTRODUCTION

1. INTRODUCTION

Developing models of business processes is a wide and complicated area of research. At a suitably high level of abstraction, any process can be made to look neat and tidy. But what happens when someone tries to decompose the process? Real-world processes are complex, muddled and messy; in a very few instances, a process will indeed be a neat hierarchy. There have been a number of “movements” in the 1980s and early 1990s that have made people recognise that they have processes, and that these processes are what the enterprise is about [Croft & Lefk., 1], [Dijk., 1], [For., 1], and [For., 2]. In Europe particularly, the emergence and development of the ISO 9000 series of standards has led to an increased concern in how an organisation ‘does’ its business in a way that ensures quality in the products or services that it delivers to its customers [Ould, 1]. The current desire for greater flexibility, higher efficiency, cost reduction and shorter cycle times, together with concern for the environment, quality and safety, demand an integrated approach encompassing all types of activity, from high level strategy to business operation [Karc. *et al.*, 1]. In addition, central to standardisation, key processes should be defined in such a way so that they are repeatable, measurable, and improvable.

Once a model of a business process is developed, one may use the model to explore the properties of the process itself, and answer questions of quantitative nature, for example, “what is the average cycle-time?”, “where are the bottlenecks?” or “can this process deadlock?”, and of qualitative nature, for example, “are the right decisions made at the right level in the organisation?” or “is the division of tasks between people involved the optimum?” [Ould, 1]. This kind of analysis drives the process of better understanding, and thus, improves the organisation. People might be driven to change the ordering of activities, increase or decrease the amount of parallel activity, restructure functions to align them better with the process, change the scheduling mechanisms and so on [Ould, 2], [van der Aalst & Hee, 1].

The current practice of treating business, operational and design issues independently, without taking into account the existing interactions, and relying on testing for the evaluation of alternatives is time consuming, expensive and rarely leads to satisfactory results [Karc., 1]. The need for an integrated approach that breaks the traditional boundaries between technical and managerial disciplines, as well as between

operational and design issues is becoming very strong. Global enterprises have to be able to respond to sudden changes in market demands, and this implies that they have to be able to propagate high-level decisions throughout the organisation down to the lowest level, and, in turn, be able to perceive and react to changes at the lowest level.

The study of engineering and technological systems has benefited immensely by the use of a system's framework, which, when supported by formal modelling tools, enables the emergence of analysis and design methodologies. System's concepts have been employed at the 'design of the system' stage, as well as when formulating control decision-making strategies for the final system. So far, system theory has evolved based on paradigms linked to physical or engineering systems, and on simple classes of mathematical models that permit the development of formal methodologies for analysis and synthesis-design. Major challenges emerge nowadays in problem areas associated with new paradigms, where the rich modelling tools of the engineering domain do not always apply. Hence, developing a successful conceptual framework and efficient methods for these alternative areas is an important challenge. An effort to develop a formal approach requires the definition of a generic system framework, which is of the conceptual type, and selection / development of relevant modelling tools that will support analysis and design.

The area of business processes has been already identified as a subject that can benefit from a unifying system framework and deployment of formal modelling tools [Karc., 1], [Theod. *et al.*, 1]. Early attempts in developing such a framework have been characterised by two trends: The first, referred as Soft Systems approach [Check., 1], has provided useful conceptual models of problems but no links to formal modelling methods. The second trend is expressed by mathematical system theory, where an abstract mathematical setting, motivated by simple paradigms, is used. However, links to the specifics of classes of systems are difficult to establish. Engineering approaches, which have been successfully applied to classes of technological problems, for example Multilevel Flow Modelling and Petri Nets, provide alternative possible routes for analysis and design. Nevertheless, they too lack a "mature" system framework within which the analytic capabilities of formal modelling techniques can develop to generic formal methodologies.

In this thesis, a general systemic framework, suitable for discussing a number of important issues arising in the analysis and design of business processes, is

introduced. In addition, it has the ability to integrate formal analysis approaches and tools. Furthermore, control theory concepts are used to support decision-making.

The aim of this research is to consider the overall problem of modelling of business processes and understand the systemic issues in their analysis and design. This aim encompasses the specification of modelling issues for understanding behavioural aspects of business processes. Also, it includes the definition of the system and control aspects, which may affect the development of a formal approach to analysis, decision making and redesign of business processes. More analytically, the main objectives of this thesis include:

- (a) The development of a general conceptual system's framework that can support analysis and decision-making problems for general processes, and is capable of integrating formal analysis methods of different types.
- (b) The understanding of requirements for modelling analysis and decision making of business processes. The analyst should, at all times, be able to identify the goals that need to be achieved, the 'essential business entities', i.e., the things that are the central subject matter of the business, their interrelationships and the business rules, which drive the process modelling exercise [Yu & Mylop., 1], [Louc. & Zic., 1].
- (c) A review and evaluation of generic methodologies and tools that are relevant to modelling of business processes. Their basic concepts, as well as their modelling logic are explained.
- (d) The application of business-oriented methodologies in a specific human resources management problem and their evaluation. Specifically, we deal with the integration of concepts and aspects of conceptual modelling, System Dynamics and Petri Nets in the context of this application.
- (e) The demonstration of the use of quantitative methods of Control theory for decision-making.
- (f) The classification of modelling methodologies based on the objectives of the modelling activity. These, after all, define the description of the process, i.e., static or dynamic, quantitative or qualitative description.

The wider theoretical and technological horizon of this research contains many more issues (that are beyond the scope of the current thesis) and this includes:

- (i) The development of the generic formal capabilities of the general system's framework

- (ii) The integration of generic formal analysis methods within a “mature” system’s framework
- (iii) The creation of a system’s framework and analysis methodology that can support behavioural aspects and their representation at the level of data systems

A distinctive feature of the research, is the study of business processes through an interdisciplinary approach that cuts across the boundaries of management and information technology. The overall effort will be placed on moving from the purely conceptual level of describing a business process to a more formal one [Theod. *et al.*, 1], [Bronowski, 1]. Business process models produced can be as detailed as the modeller wants them to be and his/her viewpoint of a process can vary as his/her goals/objectives vary. Completeness and relevance is in the eye of the modeller; the model is “right” if it helps to reveal things or can be analysed quantitatively to predict outcomes, or can be adjusted to test proposed changes, or simply aids understanding. In the long term, the understanding gained could be translated into practical tools. We focus on those aspects, which enable the propagation of business level policies, as control strategies through the different levels of the operational hierarchy. This is a prerequisite of the overall effort to develop global control and optimisation for the integrated enterprise.

In this thesis, we identify our models as models for process analysis; the latter includes structural and behavioural analysis. Their purpose falls between descriptive process models, which provide a medium for human communication about the business process, and the process support models, which an IT system (a simulation software package) runs to support the process in real time [Dicesare *et al.*, 1], [Coyle, 2], [Stev., 1], [Moore & Bren., 1], and [Abdel-Hamid, 2]. In this way, the modeller has the ability to run and test different policy scenarios, trace structural and logical deficiencies, predict outcomes when sudden changes occur, and evaluate the overall behaviour of the process. Business process modelling leads to further discussion on the efficiency of the process and to possible enhancements of it (Business Process Re-engineering) [Hammer & Champy, 1], [Ould, 2].

So far, the current trend for business consultants is to draw elegant pictures of the processes that the organisation encompasses [Check., 1], [Kingston, 3], [Naug., 1]. The analysis of the business problems is done in a rather heuristic way, mostly based on the consultant’s experience. Although it has been over thirty years now that Peter Checkland came up with his ‘Soft Systems’ methodology for modelling business

processes [Check., 1], consultants still rely on it to provide the client with an overall graphical view of the problem. Only Conceptual Modelling, as created and used by database developers and software engineers, has managed to move the analysis a step further by introducing a formalisation of the modelling process [Douglas, 2], [Chen, 1], and [Louc. & Zic., 1]. But still, there are a lot of gaps to be filled in, in order to be able to provide clients with a rigorous model of a business process. Behavioural and structural issues are too important to be omitted from the analysis.

This research work attempts to provide a natural path for moving from a conceptual level to a formal one, a brief guideline on how to go about modelling a business process. Particular emphasis is given on the objectives of the modelling exercise, which after all, define the description of the process, i.e., static or dynamic, quantitative or qualitative description (Figure 1.1). However, this work does not attempt to provide guidelines on who to interview, what data to gather, pitfalls to avoid, nor serve as a quick reference to the consultant. The aim is to evaluate various modelling techniques – and of course, we do not claim we exhaust the list of available techniques – and then, make use of them at different stages of analysis. The classification of these modelling techniques is made in Figure 1.1. The figure clarifies the process of how to build models of business processes. Historically and conceptually, Soft Systems form the initial attempt of the model developer to gather data and structural information for the process under observation. Elicitation of knowledge is based mainly on the goals and objectives of the modelling effort, and on these will all the methodologies mentioned in this thesis, rely to drive the development of business models.

Conceptual Modelling adds to the representation of a process by creating semantic models that can suppress all the irrelevant details and emphasise only the essential ones. The abstraction mechanisms used are aggregation, grouping, classification, and generalisation. However, neither Soft Systems approach nor Conceptual modelling provides a modelling support for the behaviour of systems. Additional modelling techniques provide behavioural models, that are able to express how and when changes occur to entities and relate with one another. Role Activity Diagrams produce extremely detailed qualitative representations of the process and enhance communication. Petri Nets are very good at modelling well-structured processes, where as System Dynamics can overcome fuzziness and propose a better interlinking of the entities and processes involved, when the description of the problem

is vague. The integrated framework shown below could be incorporated into the repertoire of a business process modeller.

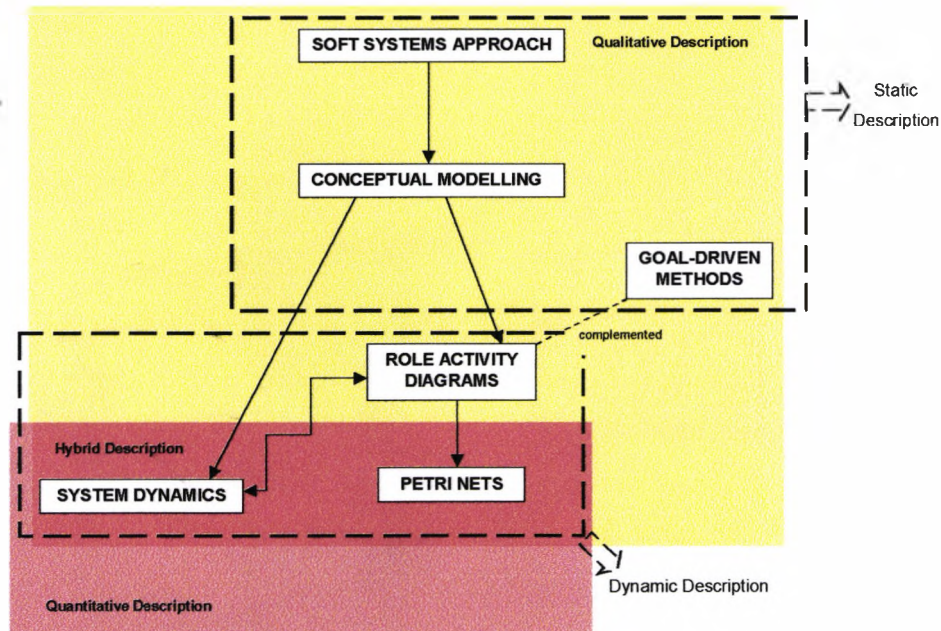


Figure (1.1): Classification of the various methodologies according to the nature of model description

The evaluation of the various modelling techniques is accomplished through their application to a very specific problem. It is a problem of human resources management in a consulting firm, or, in order to be more precise, it is a problem of resource allocation so that the company is able to meet the increasing demand for projects. The behaviour of such a system is time varying. The system is of dynamic nature, based on an information-feedback system. There are time delays between the creation of policies, decisions and actions. These time delays are based on the flows of information and resources. There is a need to create an exploratory model to help organisations not only to forecast the ramifications of policy and environmental changes, but also to give decision-makers information about the possible effects of policy choices. In this thesis, the development of such a model is introduced and the application of the proposed framework (Figure 1.1) has provided us with the understanding of requirements for modelling analysis and decision-making for business processes. The advantages and the limitations of the alternative modelling techniques have been addressed and the type of models that these techniques are suitable to provide is discussed. However, the intention of this thesis is to evaluate modelling

methodologies for business processes, rather than to create a "perfect" model that would serve the needs of a real company.

The structure of this thesis is the following: In chapter 2, some of the emerging issues when dealing with the modelling of business processes are presented. Beforehand, the problem area is specified and the need for an interdisciplinary approach, that deploys qualitative and quantitative methods and techniques, is clearly pointed out. Further important issues, such as the categorisation of processes and the definition of basic modelling concepts such as *function*, *behaviour* and *structure*, are considered. The chapter completes with the narrative description of the case study on which the evaluation of the different modelling tools will be based.

Chapter 3 provides a general conceptual framework for systems that handles issues such as basic objects, processes' topology of interconnections, variables, relations and behaviours for objects and systems. Additionally, an introduction to the notions of equilibrium and open, close loop control problems is given. The chapter concludes with a description of the Integrated Operations defined on an industrial establishment and a discussion on the organisational and measurement issues that this paradigm entails.

Chapter 4 lists some of the modelling methods relevant to the business process issue. A fairly detailed account of the potentials of Qualitative Reasoning, Functional Reasoning, Goal-driven methodologies, and Bayesian Networks is provided. These modelling approaches are referred here to invoke further work on their capabilities and links to the manipulation of the business process modelling problem. The chapters to follow form a detailed account of some other modelling techniques, such as Conceptual Modelling, the Soft Systems Approach, System Dynamics, Role Activity Diagrams, and last but certainly not least, Petri Nets.

In Chapter 5, the ability of the Soft Systems approach to draw together disparate approaches to a process is outlined; the main concepts of this approach as well as of Conceptual Modelling are presented, and their linkage is discussed. The development of the conceptual model of the consulting firm's problem is then elaborated and the power of conceptual models unravels soon after.

Chapter 6 provides a very detailed presentation of System Dynamics modelling methodology. The chapter is structured as following: First, a discussion on the basic concepts of System Dynamics and the influence of positive and negative feedback on the behaviour of the model is given. Then, moving on to the structural approach of System Dynamics, we extend the discussion on influence diagrams, the simulation

technique, the role of optimisation in System Dynamics and the issues raised when dealing with time. Having presented the general methodology of System Dynamics, we come up with some criteria of quality that a System Dynamics model should satisfy and dwell on the assumptions made when using this methodology. Soon after a brief summary of the various tools that are based on the SD approach, the SD model for the consulting firm's problem is drawn. By using this model to test a number of policy scenarios, the power and the flexibility of the method unravels. The chapter also contains a paragraph on the possible manipulation of the problem using discrete-time equations and optimal control. Finally, the limitations and difficulties, that arise when using System Dynamics in the modelling of complex systems, are discussed.

Chapter 7 presents the Role Activity Diagrams (RADs) and their basic concepts. Here, the focus of the modelling process is on the 'organisational behaviour' of the process being modelled and computational detail is not considered. This method is then used to model the consulting firm's problem. The advantages and the disadvantages of using RADs to model business processes are explicitly presented.

The journey to finding a modelling technique with hybrid characteristics, i.e., one that is able to provide quantitative and qualitative description, as well as dynamic and static description of the problem, depending on the goals to be achieved, ends with Petri Nets on Chapter 8. Petri Nets is a graphical and mathematical modelling tool designed to describe and study systems, characterised as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. As a mathematical tool, Petri Nets can be used to set up state equations, algebraic equations, and other mathematical models, as well as simulation models. As a graphical tool, Petri Nets provide a visual modelling technique where tokens are used to simulate the dynamic and concurrent activities of systems. In this thesis, we focus on a specific class of Petri Nets (PNs) suitable for the representation, validation, and verification of business processes. It will be shown that it is not difficult to map a process onto a Petri net. This special class of PNs is called High-Level Petri Nets, which are based on the classical Petri net model introduced by Petri, but are extended with "colour", "time", and "hierarchy". These extensions allow for the representation and study of complex business processes and, at the same time, inherit all the advantages of the classical Petri net, such as the graphical and precise nature, the robust mathematical foundation, and the abundance of analysis methods. The application of Petri Nets to the consulting

firm's problem is also discussed; an account of their modelling capabilities is also provided.

Finally, Chapter 9 provides the opportunity to summarise the main findings and point out the open issues and challenges. The latter draw a horizon of development of a research programme beyond this thesis.

There are also a number of appendices; in them, a more detailed account of the semantics and the capabilities of some of the methodologies mentioned in the thesis, is given. The appendix on System Dynamics also includes the mathematical equations as formed during the manipulation of the consulting firm's problem through the Ithink[®] software. Additionally, an appendix with two Matlab algorithms for the discrete-time model developed in Chapter 6 is included.

Chapter 2

PROBLEM AREA: BUSINESS PROCESSES AND EMERGING ISSUES

2. PROBLEM AREA: BUSINESS PROCESSES AND EMERGING ISSUES

2.1 Introduction

Every organisation, whether it is a firm or an industrial enterprise, has a number of *processes* that it carries out in order to achieve its business objectives, for example, handling orders for goods, recruiting staff, or designing new products. Each such process is some kind of activity within the organisation where people and machines work together to achieve some desired outcome [Ould, 1].

Nowadays, companies are attempting to manage – or even eliminate – the gap between strategic vision and operational execution by using business process modelling tools and techniques. These tools and techniques provide both a blueprint for analysing how processes are executed today as well as a road map for creating the most efficient processes for new business initiatives.

Some of the emerging issues when dealing with the modelling of business processes are presented in the following pages. Beforehand, the problem area is specified and the need for an interdisciplinary approach, that deploys qualitative and quantitative methods and techniques, is clearly pointed out. The chapter includes a narrative description of the case study on which the evaluation of the different modelling methodologies will be based. The approach presented here, involves a conceptual system framework within which the issues of re-engineering and decision-making can be addressed.

2.2 Business Processes

A *business process* is essentially that chain of activities, which has to take place to achieve a corporate goal. An example might be ordering and paying for equipment, from initial decision to buy, through to receipt and acceptance of the goods, and subsequent payment. [Crowe *et al.*, 1] define business processes as *sequences of linked functional-level activities, which take inputs and produce outputs*. [Hammer & Champy, 1] define a business process as *a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer*. '[Daven., 1]'s

subsequent definition of a process is also close to that of [Hammer & Champy, 1]. A process as defined by [Daven., 1] is *a specific ordering of work activities across time and place, with a beginning and an end, and clearly identified inputs and outputs: a structure of action*. Here, we must emphasise the fact that the description of a business process specifically does not address the way the process transforms its inputs into outputs- it describes what the process does, not how it does it. Finally, in [SOTAS, 1]'s view, business processes are *the activities, which a commercial organisation performs in order to carry out its business*.

As we can see, a business process involves *activity*. It also generally involves more than one person or machine: a process is about *groups*; it concerns *collaborative* activity. And a process has a *goal* [Ould, 1].

Simply by modelling the process, one discovers:

- What the organisation is trying to achieve with the process: the business *goals* for the process
- What constraints the organisation puts on what people can do and how they should operate: the business *rules*
- What individuals *do* to achieve the goals
- How processes are divided over *roles*, and
- How individuals within groups *interact* to work collaboratively in order to get the job done.

Therefore, the set of business processes in an organisation essentially defines what the organisation does. All organisations depend for their competitiveness on the efficiency with which their business processes operate. In addition, using a process view highlights the fact that there are compelling reasons for moving from traditional, hierarchical, organisational structures to those where the importance of business processes is recognised. These reasons include:

- the ability to deliver a better service to customers
- the creation of increased flexibility of operations
- improved job satisfaction by providing a goal-oriented focus and enabling business transformation to flatter more effective structures.

Throughout the thesis, we will adopt the following definition for a business process:

Definition (2.1): A *business process* is a set of interlinked activities within an establishment, which are organised around a common goal and deliver a certain functionality to the overall system.

The above definition of the business process is consistent with the systemic viewpoint of seeing an enterprise as an organised form of functionalities working to achieve a set of goals. A business process is then identified as a functionality, or a basic element in this system and the enterprise is the organised form of such functionalities, basic elements. A representation of the enterprise may thus be expressed as follows:

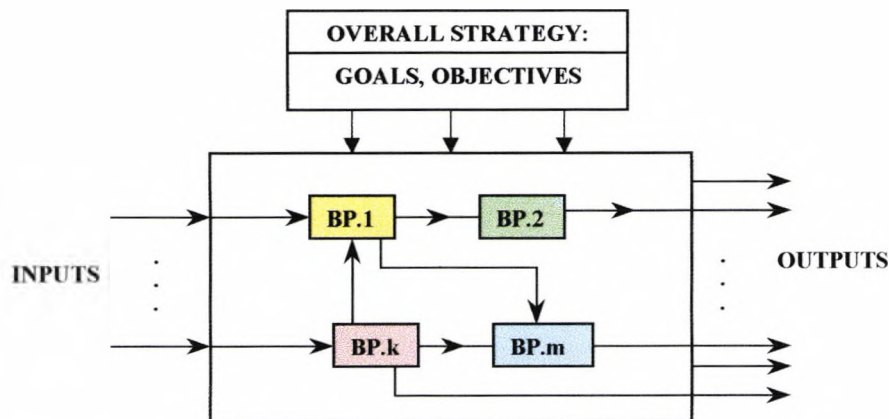


Figure (2.1): Enterprise as an interconnection of business processes driven by a common goal

In this setup, an elementary business process (BP) is the cell, the fundamental element and the enterprise is represented as the interconnection of such elements that are driven by the general goal, the objectives of the enterprise. A formal definition of the ingredients of the general system – process - will be provided in the following chapter, where a unifying introduction to general systems will be given. It is worth mentioning here, that the term *inputs* is interpreted in the standard way (and include also, external non-assignable effects, such as disturbances) and *outputs* refer to deliverables, as well as observations of internal activities (simple or composite indicators). The terms *goals, objectives* are also used in the standard way.

The important feature of the above setup is that the basic elements of the enterprise, the business processes, are organised, that is, they, themselves, are viewed as interconnected elements with a similar structure as the enterprise, and may be represented in an input-output configuration as shown in Figure 2.2.

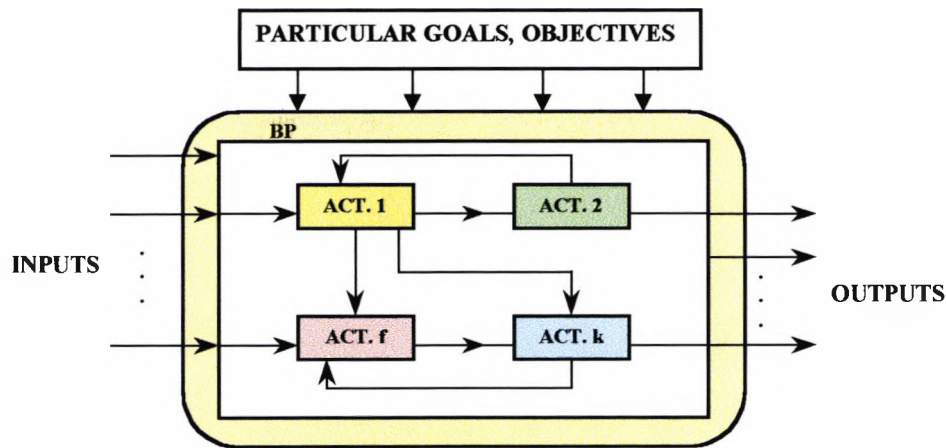


Figure (2.2): Systemic view of business process as an interconnection of activities driven by a particular goal, objective

In this figure, we see that the basic elements are now simpler blocks referred to as activities (ACT) and interpreted in the standard way. The concepts of organisation, interconnection, and connectivities are fundamental in this visualisation of enterprise and its processes. They express a natural graph-type topology, which contrary to the physical systems description, is not always fixed and is usually driven by the goals and objectives. The man-based nature of such systems, organisations, gives them a distinct character from the engineering systems, where interconnections are fixed under normal circumstances (exceptions are faults, and structural changes under variability of operational conditions). From this viewpoint, such systems are closer to those of biology, genetics, where external stimuli may lead to changes of the organisation form.

A more detailed view of the business process that pays attention to its internal structure is given below in Figure 2.3. Here, we see that there are four levels of interactions in the framework: At the bottom level, the *manpower level*, where there are people of an organisation. At the next higher level, the *internal system and structure level*, there are job definitions, organisational structure, beliefs and values, measurement and management systems, and any other concepts or systems that determine the relations between people. The third level is then the *process level*, which consists of activities or actions, which transform several inputs to an output. Finally, the fourth level is the *objective level*. The objectives of an organisation can include such things as the improvement in the critical measures of performance (cost, quality, capital, service, speed) from the organisational as well as the clients' perspective.

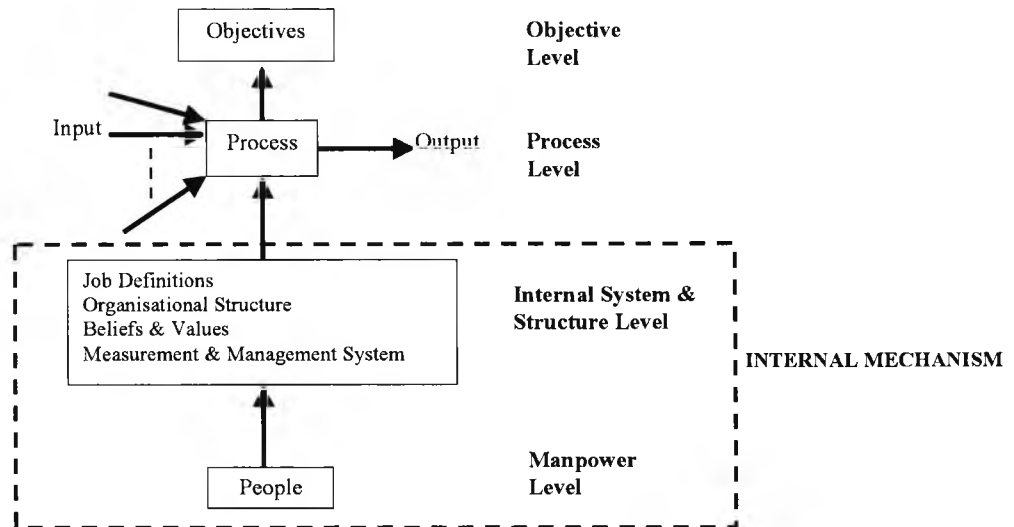


Figure (2.3): Extended description of a business process [Poh & Chew, 1]

The above framework provides an additional dimension to our understanding of the business process by introducing the notion of the *internal mechanism*; this refers to the content of the various activities associated with the process and these are classified as the *Procedural layer* (higher layer, i.e., Internal System and Structure level) and the *Physical layer* referring to people, i.e., the Manpower level. There is a natural hierarchy associated with this configuration, which is part of the overall structure of connectivities of the business process and it is due to the fact that we are considering business processes where the human element is an essential ingredient. The box referred to in the diagram as *People* is one of the activities of a business process and itself can also be a business process. By associating variables with all the activities of a business process, we create the notion of the *state vector* of the process. The latter is viewed as the overall set of all internal variables associated with the particular process and is fundamental in defining the time evolution and behaviour of the process as a function of inputs and objectives – goals. The proper setup of the input, state, output mechanism of a business process will be considered in chapter 3 and relevant models for describing evolution and behaviour will be considered in chapter 4.

The overall emerging picture of the enterprise, business processes, activities, is of an embedded organisation of processes containing different layers, with certain organisation at every layer. The distinguishing feature of this organisation is that the elements of every layer have a fractal structure similar to those of the elements at a higher or a lower level. Understanding such structures is essential, not only for setting

up mechanisms for understanding behaviour (time and event-driven), but also, in developing specifications and requirements for the software system and IT that underpins such investigations.

The embedded form of the overall organisation, which also expresses a hierarchical organisation, is described in Figure 2.4.

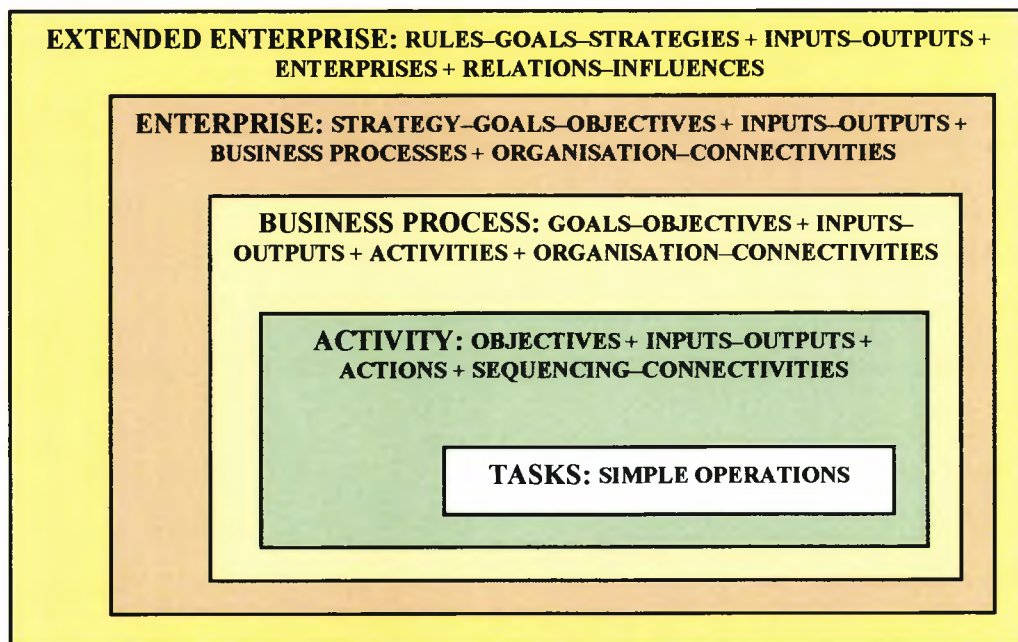


Figure (2.4): Embedding and Fractal Representation of the Overall Enterprise System

The figure summarises the embedding and the generic fractal representation of the overall enterprise and its processes. In this diagram, the terms used are interpreted in the standard way as previously discussed.

The important distinguishing feature of such systems is the significance of the strategy, objectives and goals in the shaping of inputs, in the selection of outputs, as well as in the shaping of organisation and connectivities. Strategy, goals and objectives certainly do affect the forces imposing specific forms of organisation and derive from different areas, including economic forces, social, cultural, political, geographic, governmental, and legal factors, as well as technological and competitive forces [Poh & Chew, 1]. The study of such frameworks is important in understanding the behaviour and the evolution of behaviour and characteristics of processes and enterprises. A diagrammatic representation of such framework is indicated below [Hammer & Champy, 1].

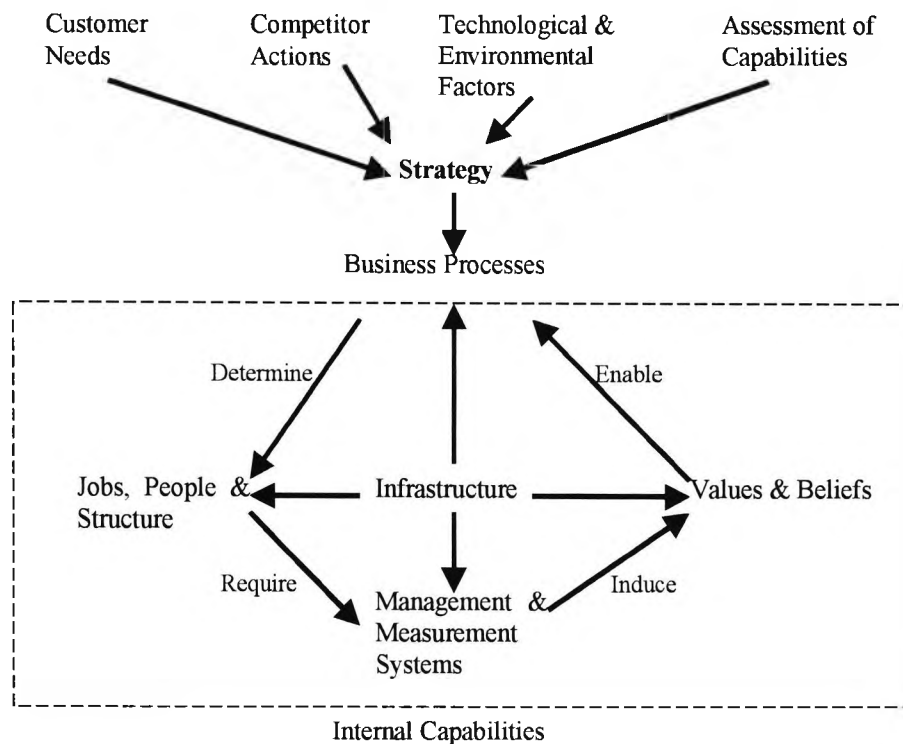


Figure (2.5): Interrelationships of the elements of a business process [Hammer & Champy, 1]

Figure 2.5 can be explained as follows:

- The external forces (customer needs, competitor actions, technological and environmental factors) and internal factors (assessment of internal capabilities) influence the formulation of the organisation strategy, which in turn determines the design of the business processes.
- The business processes then interact with (enabled by and determine) the jobs, people, organisational structure, management and measurement systems, values and beliefs, and infrastructure, to determine the internal capabilities of the organisation.

2.3 Business Process Re-engineering (BPR)

When talking about business processes, it would be a terrible omission not to refer, even briefly, to the latest trend characterising modern organisations. Companies turn to help from consultants, to modelling tools, and re-engineering techniques to analyse business goals and improve the efficiency of their business processes.

Indeed, the practice of Business Process Re-engineering (BPR) is not at all new. What is new is the jargon of Business Re-engineering believed to have been coined and

popularised by Michael Hammer, an MIT computer science professor turned to management consultant [Poh & Chew, 1]. Ever since the 'invention' of re-engineering, there has been an exponential increase in the number of researchers and consultants working in this field.

There are many and different definitions of BPR. [Poh & Chew, 1] listed all the different definitions and came up with a definition given by a framework based on the understanding of the processes as described in Figure 2.3. BPR hence involves the redesign/re-planning of each of the four levels on an on-going basis. A formal definition for BPR [Poh & Chew, 1] can be reworded as follows:

Definition (2.2): *Business Re-engineering* is a total re-planning and redesign of manpower, internal system and structure, and processes in direct or indirect response to external forces, in order to achieve some objectives, which are usually not easy to achieve given current conditions of an organisation.

Re-engineering is a process that requires a good understanding of the enterprise and of its business processes. The reference to BPR will end here. Further details will not be provided, since this is not the scope of this thesis. Nevertheless, what is of interest is the Process Level as mentioned in [Poh & Chew, 1].

2.4 The Industrial Enterprise Paradigm: Problem formulation and general issues

The study of business processes and of the associated enterprise issues is of general interest to both industrial and non-industrial enterprises (Retailing, Finance, Commercial, etc). Regarding applications and special features, we are mostly concerned with the case of the Industrial Enterprise, which has the additional feature of a strong interrelationship between General Business, Operational issues and Engineering Design problems for processes and products. The strong linking of the Business–Operational–Design issues of the Industrial Enterprise makes it a very important paradigm and this is indicated briefly by the following diagram [Karc. *et al.*, 1].

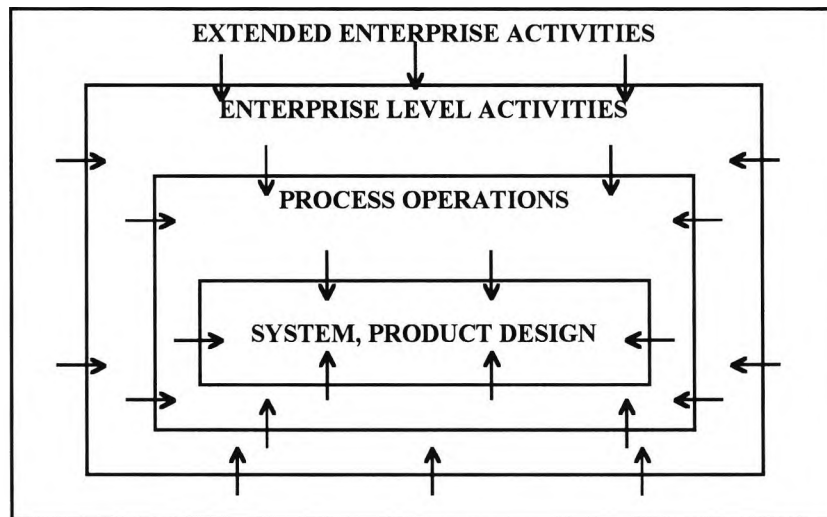


Figure (2.6): Embedding of the features of an Industrial Enterprise

The study of business processes from the top level of industrial hierarchies is complex and demands a multidisciplinary approach that cuts across the boundaries of management and information technology [Karc. *et al.*, 1]. An overview of the problem under study is given in Figure (2.7), where the identified generic blocks associated with a given Business Process are:

- i. Process Measurements
- ii. Data Modelling and Data Management
- iii. IT applications and Simulation
- iv. Generic Modelling and Support Methodologies
- v. Management, Knowledge Management and Decision Making

Process Measurements are essential as the basis of the information system (database) and for the extraction of knowledge required in modelling the information system. Data management and IT applications form the two parts of the information system that supports the monitoring, the management, and the decision-making of business processes.

The analysis of the properties-behaviour of the business processes is supported by the Generic Modelling Methodologies, which in turn can also provide the means for developing tools and techniques for management and decision-making.

In Figure 2.7, we consider business processes, which are based on industrial manufacturing systems, but our results may well apply to other cases, for example consultancy companies. The fact that industrial enterprises operate nowadays in a global market environment, implies that requirements of higher product quality, higher

efficiency, greater flexibility in production, rapid development of new products, while at the same time operating under stricter environmental and safety regulations become more prominent. If the fundamental aspects of the enterprise activities are integrated, then such set of frequently conflicting requirements can be met [Karc. *et al.*, 1]. In fact, business level requirements have to be readily translated to production performance criteria and the latter to design, or redesign requirements of process units. Conversely, technical issues of design and operational aspects must always be interpreted at the business level and their role in the business level performance indices has to be established. This property expresses the notion of strong coupling between processes at the different layers of the hierarchy, which are the distinguishing features of the industrial enterprise.

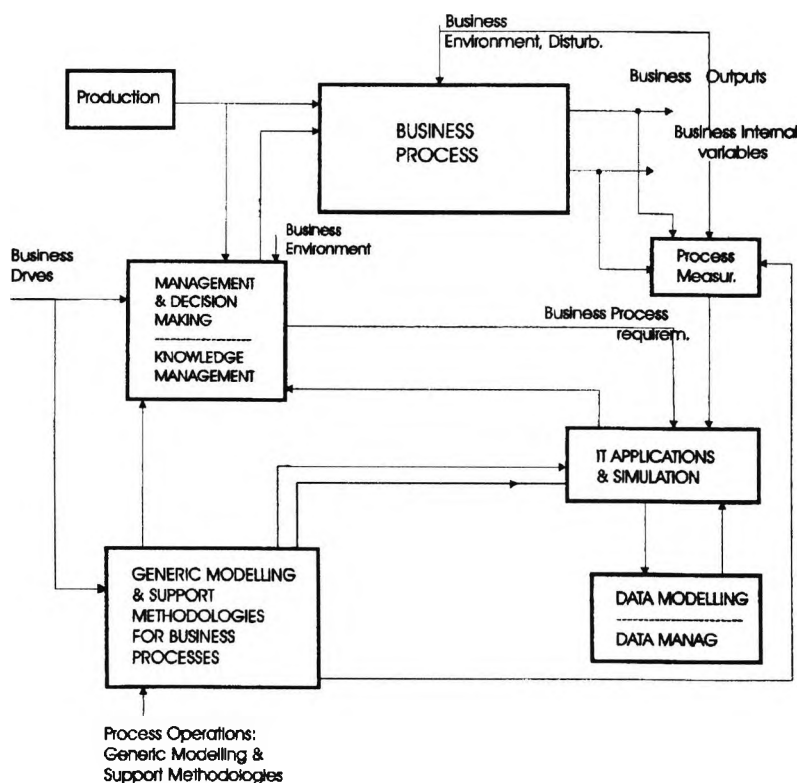


Figure (2.7): Overall description of the problem area

2.5 Structure, Behaviour and Function

There is quite a lot of ambiguity when it comes to using terms such as *function*, *behaviour* and *structure*. Since a soft system, such as a business process, is described in

terms of goals, functions, behaviour, and physical components, it is compulsory to have a clear understanding of what each of these concepts encompasses.

Structure indicates the arrangement and relationship of the components of a device model [Keuneke, 1]. [Kuipers, 4] describes the structure of a system in terms of a collection of continuous parameters and constraints among them. A differential equation would represent the structure of the system [Kuipers, 5].

Behaviour is described in terms of changes to position and direction in qualitative quantity spaces. Following [Kuipers, 5] analysis again, the solution to the differential equation would provide the behaviour of the system. [Keuneke, 1] says that behaviour specifies how, given a stimulus, the *function* is accomplished. In other words, behaviour is the causal sequence of transitions of partial states.

Some define function to be equivalent to behaviour. [Chandr. & Kaindl, 1] state that function expresses an intention of a designer or a user, that is an abstraction of behaviour. [Chandr., 1] in a recent proposal on the definition of function, has argued that a *function* or *role* of an object is an effect it has on its environment. It has been recently defined in [Keuneke, 1] that the function of a device is its intended purpose. So, the *function* specifies the result or goal of a component's activity. A more formal definition in abstract system terms of all the above will be considered in chapter 3.

It is worth stressing out that our models of interest, i.e., the process models, concentrate on dynamic behaviour. In an information system [Rumbaugh *et al.*, 1] identifies three perspectives: the data or informational perspective, the functional perspective and the behavioural perspective. The data perspective focuses on the data, manipulated by the system. Conceptual modelling embodies this perspective. The functional perspective focuses on the transformations which the system performs – System Dynamics offer this aspect –, while the behavioural perspective focuses on when things happen, – Petri Nets is probably one of these techniques. In our process models, the emphasis is on the behavioural perspective. The models are created to ensure the smooth running of the process rather than the computation of a result. Note that the three perspectives defined above are not independent; they are rather embedded in a natural nesting and they are thus interrelated.

2.6 Classification of business processes

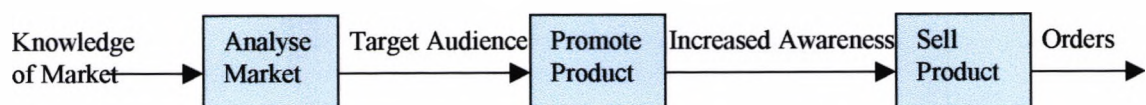
Literature provides us with an abundant set of classification of processes. Broadly, processes can be split into two groups: highly structured, repetitive, often high-volume processes; and flexible, dynamic processes, where the structure of the process can vary markedly between occurrences (e.g. bid management). References [Gulla & Lindl., 1], [Porter & Millar, 1] point out that activities in a business process can be considered as one of three types:

- a. *Production activity*: this type produces results such as information or materials.
- b. *Co-ordination activity*: this type controls the performance of the production activity without directly adding value to the product.
- c. *Information Storing activity*: this type corresponds to the recording of events and relevant information and is linked to the Data Formation System.

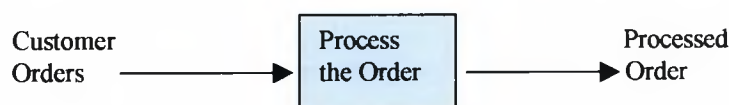
[Crowe *et al.*, 1] develop a generic set of Strategic Business Unit (SBU) processes, and a decision support system to reduce the risks while maintaining or improving the rewards. In all, eight major business properties are identified. The taxonomy seems to represent the main business processes for any firm, but it is not necessary that every firm have all the eight processes mentioned here. The taxonomy developed is presented below:

2.6.1 Taxonomy of Business Processes

1. *Knowledge of Market to Orders*: the process of taking preliminary knowledge of a product market, analysing it to find the target audience, increasing the awareness of this audience through product promotion for the purpose of winning orders.



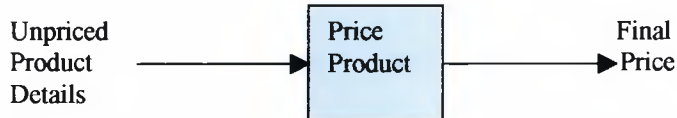
2. *Customer Order to Processed order*: the process in which a customer order is received and all of the necessary paper work, etc. is processed so the order can be filled.



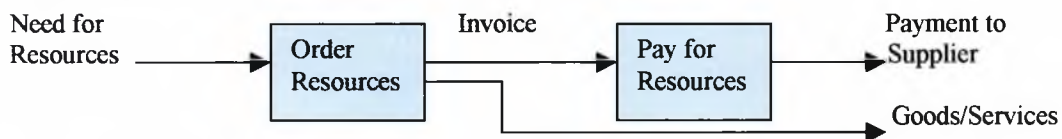
3. *Concept to Successful Design/Redesign*: the conversion of a product idea (or product improvement idea) into a complete set of verified product plans.



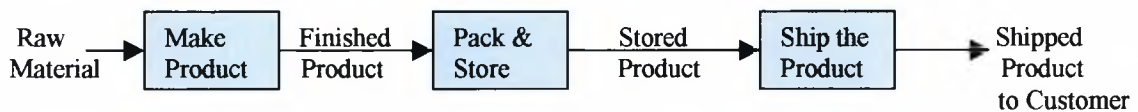
4. *Unpriced Product to Final Price*: to compile all of the direct and indirect factors used that add cost to a product to determine how much the product should sell for.



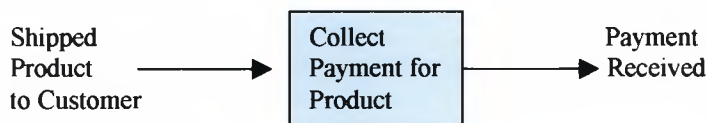
5. *Need for Resources to Payment*: the process of acquiring all goods and services for production, and sending payment to the suppliers.



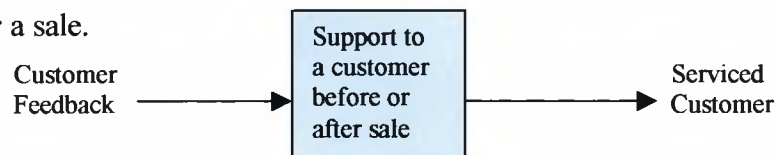
6. *Raw Material to Shipped Product*: the conversion of raw material into a finished product, which is then packaged, stored, and then shipped to a customer.



7. *Shipped Product to Payment Received*: the process of collecting payment for all products that have been shipped to a customer.



8. *Customer feedback to Serviced Customer*: to provide support to a customer before or after a sale.



[Crowe *et al.*, 1] have also developed a decision model (software package called DPL), which contains both deterministic and stochastic elements, based upon the concepts of decision tree analysis and influence diagramming. Through this model, the business processes can be ranked in the order of importance for the firm to re-engineer. American Productivity and Quality Center's International Benchmarking

Clearinghouse, [APQC, 1], has also come out with their own Process Classification Framework, which pretty much agrees with the work of [Crowe *et al.*, 1]. They also divide processes into two large categories, the *Operating Processes* and the *Management & Support Processes*. [Ould, 1] proposes another way of dividing business processes into three broad types (see Figure (2.8)):

- *Core processes*: they concentrate on satisfying external customers. They directly add value in a way perceived by the customer of the business. They respond to a customer request.
- *Support processes*: they concentrate on satisfying internal customers. They might add value to the customer indirectly by supporting a core business process, or they might add value to the business directly by providing a suitable working environment.
- *Management processes*: they concern themselves with managing the core processes or the support processes, or they concern themselves with planning at the business level.

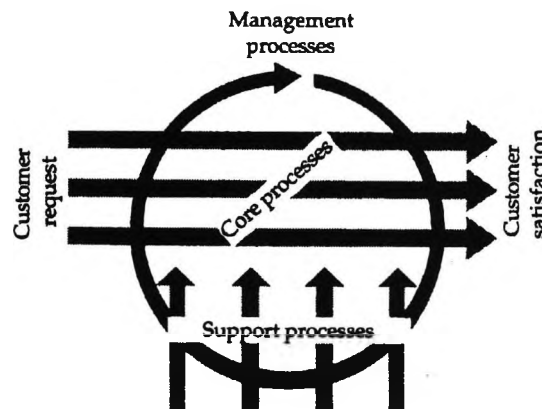


Figure (2.8): The three types of business process ([Ould, 1])

Finally, the PRIMA project matches our view of a business process as a functionality and categorises processes by analysing their function. Below are three classifications of business process [PRIMA, 1].

- *People processes*: these take place when a management team identifies the business objectives and key performance indicators that will be used to run the business. The details of these business processes change frequently and are not readily analysed. They require very little IT support for the process itself but very good quality information as input.

- Computer processes: these can be fully automated. They are required to successfully operate a computerised system. The computer system will require a mechanism to prevent sensitive information being given to unauthorised users, load monitoring, back-up and transaction logging will be required to maintain the integrity of the information.
- Shared processes: this third set of business processes requires a direct link between the functionality of the business process and the functionality of the computer system.

PRIMA project also introduces the *business process model* that has six high level processes. The processes are in three pairs. There is an exploitation layer (top green), which looks at the short term issues and the investment layer (bottom blue), which looks at the longer term strategic direction. The processes are paired together as they focus on:

- Products and Processes
- Cash and Finance
- Production and Maintenance

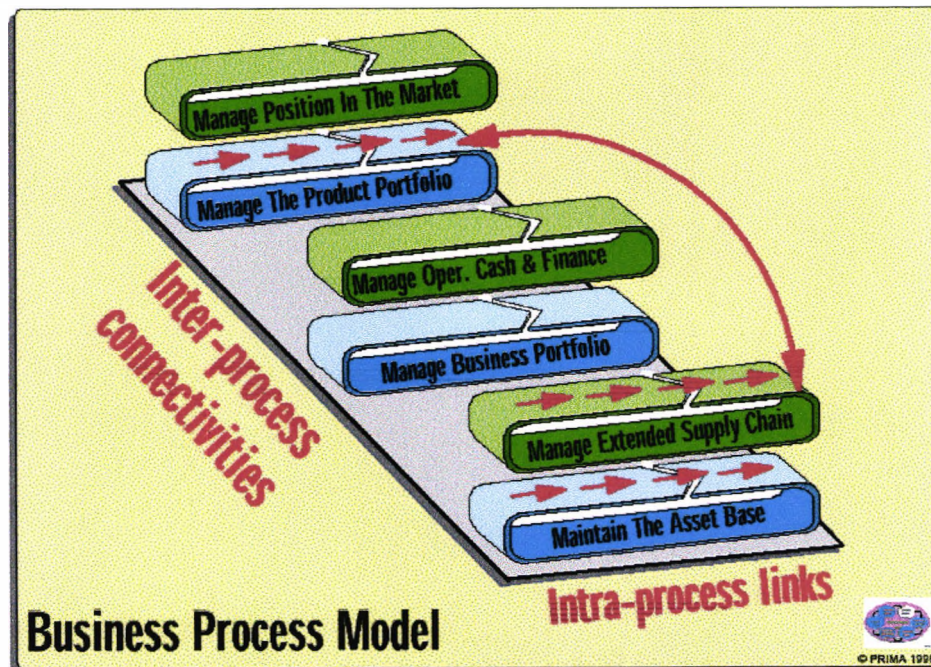


Figure (2.9): PRIMA's Business Process Model [PRIMA, 1]

2.7 Research Focus

The focus of this work is on the modelling requirements of an information system, and in general of a *soft system* (by this, we mean a system which involves

human activity or human judgement, such as an activity of business management). We will try to explore the abilities as well as the limitations of the individual modelling techniques and methodologies. Our aim is to evaluate modelling methodologies and choose the ones that better suffice to be incorporated into the repertoire of a business process modeller. The above work requires a unifying framework of system theoretic nature for the study of Enterprise and Business Processes. Such a framework is essential for the development of control and diagnostics framework, which is independent of the particulars of a business process.

A specific case study has been chosen to provide the basis on which the analysis and the evaluation of the different methodologies, as well as of the overall framework will take place. The Human Resource Management System of a consultancy firm is studied here. This system captures the hiring, training, assimilation and transfer of the firm's workforce. Such actions are not carried out in a vacuum but are driven by the company's estimated turnover, which is an indication of demand for services. Consultancy services are dynamic in nature and are based on an information-feedback system. In system models, variables are organised in a circle, or loop of cause and effect relationships. The behaviour of such systems is time varying. There are time delays between the creation of policies, implementation decisions and actions.

The understanding of the dynamics and general principles underlying the structures and patterns of behaviour of the consultancy service delivery systems is crucial to being able to plan and maximise service in an ever changing environment, as well as, to optimise the use of the existing services and to design new training and recruitment policies. The general components of the consultancy services delivery system consists of feedback relationships involving the identification of discrepancies in the demand of services, and the ability to meet those demands. The challenge for managers and planners is to identify and understand the causal relationships. By creating a model of the particular system, through the use of certain modelling techniques, we can derive a recruitment policy. After all: "...Model experimentation is now possible to fill the gap where our judgement and knowledge are weakest – by showing the way in which the known separate system parts can interact to produce unexpected and troublesome overall system results" [For., 2].

2.7.1 The consulting firm's problem – a case study

The management of a consultancy firm has been concerned about the firm's seeming inability over the past few years to grow with the market for consultancy services. The firm also seems never to have the right numbers of qualified consultants and junior consultants to offer a balanced service on existing jobs and to obtain new business.

In the case under survey, we assume two different types of consultants, the *internal* consultants and the *external* ones. The internal consultants are further divided down to other levels, according to working experience and professional duties. Starting from the internal consultants first, the firm recruits new staff, who require a specific time of supervision - training before they are considered qualified (the *junior* consultants). This is called an "orientation phase". The orientation process has both technical as well as social dimensions [Abdel-Hamid, 1]. On the technical side, (newly hired) personnel often require considerable training to become familiar with an organisation's unique mix of software packages and project methodologies. As for 'social orientation': "... it refers to the processes of teaching the new recruit how to get along in the organisation, what the key norms and rules of conduct are, and how to behave with respect to others in the organisation. The new recruit must learn where to be at specified times, what to wear, what to call the boss, whom to consult if he/she has a question, how carefully to do the job, and endless other things which insiders have learned over time", [Schein, 1]. Here, we assume that the firm hires and generally supervises juniors for a period of 5 years, before becoming full-fledged consultants. Junior consultants receive supervision by consultants and senior consultants of the firm. At the end of the training, a proportion of the junior consultants leave, for various reasons, the rest becoming qualified *consultants*. The average time for promoting from consultants to *senior* consultants is 7 years, during which some of the consultants may leave the firm.

Senior consultants tend to stay with the firm for quite a long time (usually, the senior consultants' quit fraction is the lowest), before they leave to pursue other careers. For every 7 junior consultants, a qualified consultant acts as tutor/supervisor, and supervisors and junior consultants are able to work on projects, though they are less productive compared to the rest of the qualified personnel, who are not acting as

supervisors. Consultants and senior consultants can move freely between acting as supervisors, or working on projects.

The consultancy workforce of the firm also includes the external consultants, who are employed for a short period of time. Their number, as well as their longevity in the firm depend on the market demand for consultancy services (number of projects per annum). Here, it is assumed that externals are hired for a specific project and they leave the company only when the project finishes. On average, each project requires a certain number of qualified consultants, or their equivalent in juniors and supervisors, but, if the actual number of equivalent consultants is less than the number required for the projects undertaken, jobs tend to take longer than normal, and vice versa. There is also a company standard practice that each project requires a certain number of qualified consultants who are not at the same time acting as supervisors.

The prediction of external demand, of course, requires measurement. The rate of obtaining new projects is governed by an exogenous demand for consultancy services that grows slowly and tends to rise and fall roughly in line with the business cycle. The number of projects currently being worked on, as well as the requirements they impose, govern the number of consultants (external and internal) and equivalents the firm should have. The discrepancy between that and the actual number is a component of the desired recruitment rate. Finally, the company is assumed to have a policy of not dismissing staff on the grounds of redundancy, hence, junior consultants are hired if the manpower discrepancy is positive or, in order to cover the overall quitting of staff.

2.8 Conclusion

Modern business is taking place against a background of unprecedented change. The keys to competing in today's environment are quality, speed, and flexibility. What this means, in other words, is that companies must examine their business processes very closely to understand how the performance of those business processes can be optimised. We have seen that a business process is a sequence of activities that achieve a business result, and have presented a first taxonomy of types of processes. The main focus of this research work is to model business processes in the best possible way, and that would imply moving from the purely conceptual way of modelling to a formal level of knowledge representation.

The distinguishing feature of the current approach is that it is based on a generic system theoretic framework and related tools that guarantee independence of the particulars of the corresponding process. Such a framework is closely linked to the specification, development of the IT, Data Information System, but goes further. This is achieved by relying on a variety of generic behavioural models that may be used for the particular subprocesses, by recognising the significance of interconnection structure – connectivities, by using an integrated framework for the study of Control (Decision-Making), Observation (Monitoring, Evaluation, Measurement) and Modelling aspects. The overall effort is a multidisciplinary task, which sets to specify, first, the overall problem area and, then, develop concrete concepts and tools for particular subprocesses.

Chapter 3

A SYSTEMS APPROACH IN THE CONTEXT OF GENERAL PROCESSES

3. A SYSTEMS APPROACH IN THE CONTEXT OF GENERAL PROCESSES

3.1 Introduction

The study of business processes is an interdisciplinary task that relies on various disciplines such as finance, psychology, software engineering, measurement, operational research, and mathematical modelling, as well as on methodologies for decision-making. As such, it requires a holistic approach and thus a systems type framework. The development of a systems framework for general systems is not a new activity [Tak. & Takah., 2], [Mes., 2], [Gog., 1] and [Gog., 2]. However, such developments have been influenced predominantly by the standard engineering paradigm and as a result they failed to cope with new paradigms such as those of the business processes, data systems, biological systems, and so on. The development of a general system framework is well beyond the scope of this thesis. Nevertheless, a first attempt to identify the areas and issues where abstract system theory requires extensions and modifications to cope with the challenges of new paradigms, such as that of business processes (BP) is made here.

From another perspective, this chapter aims to provide a terminology that will be used in the following chapters. The effort will be placed on the definition of a system and its basic associated concepts. Finally, a discussion on some fundamental concepts linked to business processes, such as control, organisation and management will be provided.

3.2 The notion of a system

The definition of a system, given here, is rather general and aims to encompass many paradigms (including the traditional engineering and business ones).

Definition (3.1): A *system* is an interconnection, organisation of objects that is embedded in a given environment.

□

Such a definition is very general and uses as fundamental elements: *objects*, *connectivities – relations* (topology), and *environment*. It can be symbolically denoted as:

| |
|--|
| SYSTEM \longleftrightarrow OBJECTS + TOPOLOGY OF RELATIONS + ENVIRONMENT |
|--|

Remark (3.1): This definition of a system is suitable for the study of ‘soft systems’. It does not make use of notions such as causality, input-output orientation, definition of goal, behaviour, and so on. Quite a few systems do not involve these features and thus they have to be introduced as additional properties of certain families.

It should be noted that the notion of a system is conceptual and involves fundamental notions that are the result of our experiences, previous knowledge. Hence, this notion is abstract and may take concrete form as soon as the notion of system model is introduced. We consider first the meaning of the basic ingredients of the system’s definition.

3.3 The notion of Objects

In this thesis, *objects* are considered to be the most primitive element, allowing them to be almost anything. By not restricting the definition to any particular class, we allow freedom to construct any type of system that is of interest.

Definition (3.2): An *object* is a general unit (abstract or physical) defined in terms of its attributes and the possible relations between them.

□

As such, an object is defined in terms of a collection of observations of some selected characteristics and of the relations between these attributes, expressing in this way, a form of organisation or a degree of knowledge. The relations between the attributes may be functional, linguistic, structural, and so on. They express knowledge that stems from past history of the object or the environment in which it has been operating. Thus, a collection of observations and the possible relations between them are considered in order to define an object.

There are different types of objects that may be classified as:

- atomic versus composite
- neutral versus relational

The consideration of objects, which are *atomic*, implies our inability or lack of desire to decompose them into simpler elements. The emergence of objects, which are themselves interconnections of other more basic objects, leads to the notion of *composite* forms. *Neutral* objects are simply characterised by their attributes, whereas *relational* objects involve both attributes and their relations. The latter indicate objects of richer nature that may also be identified by such relations.

Definition (3.3): For a given object, we define its *environment* as the set of objects, signals, events, structures, which are considered topologically external to the object, and are linked to the object in terms of relations with its structure, attributes.

□

The essence of this definition is that for a given object, a boundary around the object that includes all structures and attributes associated with it, is identified. The object under consideration may be related to other objects that are, however, considered to be external. The existence of the objects' environment implies crossings of the imaginary boundary and such crossings indicate the connectivities of the object to objects in its environment. This provides a local view of the environment and the interconnection of it with the other objects of the system. The previous statement may be indicated by the following Figure 3.1.

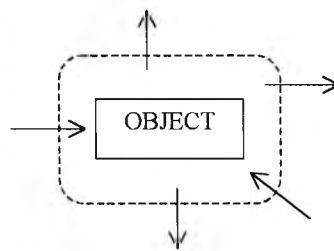


Figure (3.1): Object Environment Relation

The linking of objects to an environment provides a classification of objects, i.e., *autonomous* and *embedded* objects. An *autonomous* object has no link to its environment and thus its attributes are not affected by external stimuli. An *embedded* object has active link to its environment and thus its attributes are affected by external stimuli, events. An autonomous object will be called *static*, when its attributes are not time-dependent and *active* otherwise. The topology of the linking of an object to its

environment defines the *local interconnection structure*. This structure may be *fixed*, or *evolving* (time or event dependent) and introduces an additional classification for embedded objects. Clearly, an object may function as autonomous, or embedded and active, or static; the term *mode* will be used to refer to the different ways one considers an object. The characterisation of objects uses the notion of attribute and this is formally defined below:

Definition (3.4): An *attribute* for an object is an identifiable and possibly measurable characteristic of the object.

□

In the current context, identifiability implies the ability to associate a label, tag with the object, whereas measurability requires ability to quantify it by associating a set of values, or functions. Attributes, which are determined on an object in the autonomous static mode, will be referred as *parameters* and those that are defined in the active, or embedded mode will be called *variables*. Note that such a classification may depend on the different stages of the lifecycle of the object. The determination of the parameters of an object may be the outcome of direct observations, a priori knowledge, or the outcome of experiments at some time in the past.

We shall denote an object by B and by $P(B), V(B)$ the corresponding set of parameters and variables. Assuming the object B to be in the general embedded (or autonomous) mode, we may define *relations*, with the standard mathematical meaning of the term, on the set $V(B)$; these may be trivial (identity mappings), or nontrivial and this set will be denoted by $R(B)$ and referred as the *full relations set* of B . A subset of $R(B)$ may be defined when the object is considered in the autonomous active mode; this is denoted by $R_a(B)$ and referred as *autonomous relations set* of B . The subset of $V(B)$ on which there are no trivial relations will be referred as *implicit states* and denoted by $\mathfrak{I}(B)$. The subset of nontrivial relations will be denoted by $\tilde{R}(B)$ and provide some form of identity for the object B . The absence of trivial relations guarantees some form of independence for the resulting set of variables in $V(B)$. We may further classify $\mathfrak{I}(B)$ as follows:

Definition (3.5): All variables of $V(B)$, which have nontrivial relations and are defined when the object is in the autonomous mode, will be called *extended states* and the set is denoted by $\hat{\mathfrak{I}}(B)$. Any subset of $\hat{\mathfrak{I}}(B)$, with the additional property that the elements are independent (in some algebraic sense) and describe completely $\tilde{R}(B)$ will be referred as *states* of B . The latter are denoted by $\tilde{\mathfrak{I}}(B)$.

□

Such a definition is intuitive and rather conceptual, and for classes of objects, an appropriate mathematical formalism is required in order to provide answers to issues such as:

- (i) Definition of independence of variables and relations
- (ii) Expression of dependent variables in terms of an independent set of variables

Clearly, for general dynamical systems and linear systems in particular, the vector space algebraic setup is adequate for addressing the above. For general classes of objects a more general framework is needed.

The set of implicit states, extended states and states associated with an object may be ordered into a vector form. Such representations may be denoted as $\underline{\xi}_B$, \hat{x}_B , \tilde{x}_B respectively and $\mathfrak{I}(B) = \{\underline{\xi}_B\}$, $\hat{\mathfrak{I}}(B) = \{\hat{x}_B\}$, $\tilde{\mathfrak{I}}(B) = \{\tilde{x}_B\}$ provide an explicit representation of the corresponding elements. To every variable in $V(B)$ there is an associated set of values, indexed by time or events, and this is called the *range set* of the particular variable. The ordered set of all range sets of $V(B)$ is denoted by $T_B = \langle V(B) \rangle$ and it is called the *space* of $V(B)$. The corresponding spaces for $\underline{\xi}_B$, \hat{x}_B , \tilde{x}_B are denoted respectively by Z_B , \hat{X}_B , \tilde{X}_B and shall be referred as *implicit-, extended-, state-space* respectively of B . The space Z_B expresses all knowledge one has about the object at a given time and for a set of events that have occurred. Such a space, however, does not necessarily imply minimality, independence, as far as the information contained. Note that for the above spaces the following set relationship holds:

$$\tilde{X}_B \subseteq \hat{X}_B \subseteq Z_B \subseteq T_B$$

An important classification of objects is based on the cardinality μ_B of the set $V(B)$. During the object's lifecycle, certain variables may vanish and new ones may appear together with the possible changing of values of the variables that are preserved.

There are two extreme cases associated with the object: The first corresponds to the disappearance of all variables at a certain instance and it is referred as the *death* of the object. The second characterises the emergence of an object when a number of variables associated with it emerge; this is then referred as *birth* of the object. During the lifecycle of a process, objects may have variables disappearing, or emerging and this leads to the following definition.

Definition (3.6): An object B , for which its cardinality μ_B of $V(B)$ remains fixed throughout the object's lifecycle, will be called *fixed*; otherwise, it will be called *evolving*. □

Remark (3.2): The above definition is motivated by the need to consider the system's evolution throughout its lifecycle. Indeed, objects in systems may have variable cardinality and some of them may be born, or die throughout the system's lifecycle. The fact that we refer to $V(B)$ varying cardinality implies that variables linked to the interconnection topology of objects may also emerge or disappear. The latter implies that the interconnection topology is also subject to variability.

The relations defined on $V(B)$ of an object provide a characterisation of it in an explicit way. Objects for which the full relations set $R(B)$ contains at least one nontrivial element will be called *relational* objects. If $\underline{\xi}_B$ is the implicit vector, then the existence of nontrivial relations amongst the components of $\underline{\xi}_B$ may be expressed in a functional form as

$$f_B(\underline{\xi}_B) = 0 \quad (3.1)$$

For traditional dynamical systems, the above expression has a standard functional interpretation. For more general objects, there is a need to develop an appropriate algebraic setup within which such relations may be interpreted.

The embedding of an object to an environment implies that the vector $\underline{\xi}_B$ may be partitioned as

$$\underline{\xi}_B = \left[\underline{w}_B^t; \hat{\underline{x}}_B^t; \underline{v}_B^t \right] \quad (3.2)$$

where \underline{w}_B , \underline{v}_B denote vectors expressing *generalised outputs*, *generalised inputs* respectively, that is variables feeding through to other objects, associated with external objects respectively that express the interactions of the object with its environments locally; $\underline{\hat{x}}_B$ denote the internal variables or extended state of the object. The vectors \underline{w}_B , \underline{v}_B are manifestations of the *local structure* of the object, that is the way the given object is embedded in its environment. The vector \underline{w}_B may be partitioned as

$$\underline{w}_B = \left[\underline{y}_B^t, \underline{\hat{w}}_B^t \right] \quad (3.3)$$

where \underline{y}_B denotes the vector of *local measurements*, observations and $\underline{\hat{w}}_B^t$ are the remaining variables, which are not necessarily measured, but with \underline{y}_B together, become inputs to other objects. Similarly, \underline{v}_B may be partitioned as

$$\underline{v}_B = \left[\underline{u}_B^t, \underline{\hat{v}}_B^t \right] \quad (3.4)$$

where $\underline{\hat{v}}_B$ is a vector with variables associated with other objects and \underline{u}_B is a vector with variables *arbitrarily assignable* and referred as *local inputs*. The definition of partitioning of the implicit vector $\underline{\xi}_B$, as defined above, describes explicitly the embedding of the object to the environment, specifies the local structure and it is referred as the *orientation* of the object with respect to its environment. This orientation is described diagrammatically as

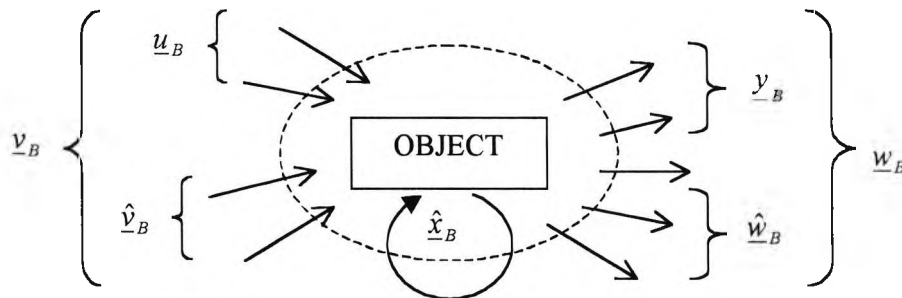


Figure (3.2): Object orientation and the object's embedding into the environment

The spaces corresponding to \underline{v}_B , $\underline{\hat{v}}_B$, \underline{u}_B , \underline{w}_B , \underline{y}_B vectors will be denoted by

V_B , \hat{V}_B , U_B , W_B , Y_B respectively and referred as:

V_B : input influence space

W_B : output influence space

U_B : control input space

Y_B : output measurement space

\hat{V}_B : internal input space

and characterise the nature of the object's embedding during its lifecycle.

An interesting class of relational objects are those for which (3.1) expression may lead to explicit oriented relations of the type

$$\underline{y}_B = \Phi_B(\underline{u}_B, \hat{\underline{v}}_B) \quad (3.5)$$

$$\hat{\underline{w}}_B = \hat{\Phi}_B(\underline{u}_B, \hat{\underline{v}}_B) \quad (3.6)$$

Definition (3.7): A relational object B for which the implicit relation (3.1) can be expressed in the form (3.5), (3.6) will be called *strongly oriented* and Φ_B , $\hat{\Phi}_B$ will be referred as *transfer-* and *co-transfer-relations*.

□

It is worth noting that if the object environment is structured, then the object orientation is partially structured (some freedom may exist in the selection of \underline{u}_B , \underline{y}_B). If, however, the environment is not explicitly defined, then the orientation of the object may be the result of specific design (of input-output structure) and thus, not uniquely defined. In the case where only (3.5) may be explicitly defined, the object will be called *weakly oriented*.

Objects have been identified with the set of variables and relations associated with them. For a given variable of an object x , its values evolve as a function of time and/or events. The time and possibly event evolution of a variable is called *behaviour* of the variable and it is a notion that extends to the implicit and to other vectors associated with the object. Relations on an object, which involve behaviours of variables, will be called *behavioural relations* and, in general, they comprise a proper subset of the relations defined on an object; such relations are crucial in describing the lifecycle evolution of an object.

3.4 The notion of Composite Objects

The essence of a system is the organisation of objects in a specific way and in general terms this is what describes the notion of a composite structure. Oriented

objects, in the form described by Figure 3.2, interact in a certain manner via a given topology of influences, and this produces composite objects exhibiting properties that in general are different to those of constituent objects. Composition of objects has two main aspects:

- (i) Definition of an object-based, local, interconnection structure
- (ii) Rules for interconnecting objects

The definition of an orientation, as described in the previous section, involves a partitioning of the implicit vector $\underline{\xi}_B$ as:

$$\underline{\xi}_B = [\underline{y}_B^t, \hat{\underline{w}}_B^t, \hat{\underline{x}}_B^t, \hat{\underline{v}}_B^t, \underline{u}_B^t]^t \quad (3.8)$$

and such a partitioning leads to the following definition:

Definition (3.8): Given an orientation of the implicit vector of an object B , as in (3.8), we define:

- (i) The pair $(\underline{y}_B, \underline{u}_B)$ together with the relations R_y, R_u of $\underline{y}_B, \underline{u}_B$ with the implicit vector $\underline{\xi}_B$ is defined as the *decision structure* of the object; (\underline{y}_B, R_y) defines the *measurement* and (\underline{u}_B, R_u) the *control structure* of the object.
- (ii) The pair $(\hat{\underline{w}}_B, \hat{\underline{v}}_B)$ together with the relations $R_{\hat{w}}, R_{\hat{v}}$ of $\hat{\underline{w}}_B, \hat{\underline{v}}_B$ with the implicit vector $\underline{\xi}_B$ is defined as the *interconnection structure* of the object; $(\hat{\underline{w}}_B, R_{\hat{w}})$ defines the *output connection structure* and $(\hat{\underline{v}}_B, R_{\hat{v}})$ the *input connection structure*.
- (iii) The vector $\hat{\underline{x}}_B$ together with its relations $R_{\hat{x}}$ with the implicit vector $\underline{\xi}_B$ defines the *internal structure* of the object.

□

An oriented object enters relations with other objects and the general environment as a shell in the way described below:

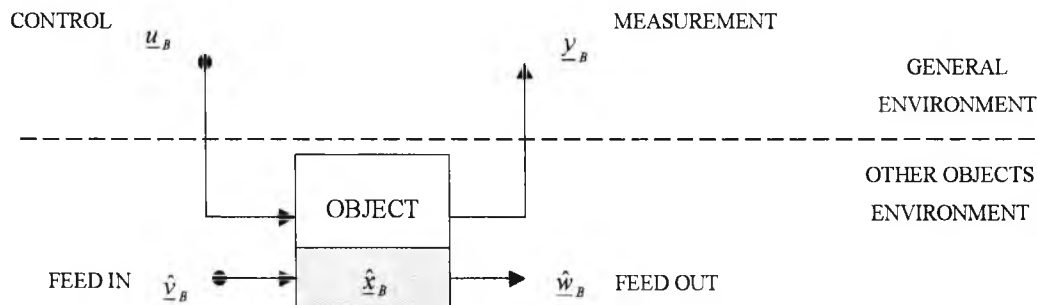


Figure (3.3): Objects interaction with environment representation

The vectors $\underline{u}_B, \underline{y}_B$ are elements of the *global control* and *measurement structure*, that will be defined later, whereas the pair $(\underline{\hat{w}}_B, \underline{\hat{v}}_B)$ define the available variables that may enter relations with other objects and thus are elements of the interconnection structure that is defined below:

Definition (3.9): Consider a set of objects $\{B_i, i \in \rho\}$ with interconnection vectors $(\underline{\hat{v}}_i, \underline{\hat{w}}_i)$, $i \in \rho$ and let \hat{V}_i, \hat{W}_i be the corresponding spaces of $\underline{\hat{v}}_i, \underline{\hat{w}}_i$ respectively. If $\hat{V} = \hat{V}_1 \oplus \dots \oplus \hat{V}_\rho$, $\hat{W} = \hat{W}_1 \oplus \dots \oplus \hat{W}_\rho$ denote the aggregate spaces, that is $\forall \underline{\hat{v}} \in \hat{V}$ and $\underline{\hat{w}} \in \hat{W}$:

$$\underline{\hat{v}} = [\underline{\hat{v}}_1^t, \dots, \underline{\hat{v}}_\rho^t] = \underline{\hat{v}}_1 \oplus \dots \oplus \underline{\hat{v}}_\rho, \quad \underline{\hat{w}} = [\underline{\hat{w}}_1^t, \dots, \underline{\hat{w}}_\rho^t] = \underline{\hat{w}}_1 \oplus \dots \oplus \underline{\hat{w}}_\rho \quad (3.9)$$

then we define as an *interconnection structure* any map F such that $F: \hat{W} \rightarrow \hat{V}$.

□

It is clear that the nature of objects and their associate variables define the nature of the interconnection structure. Such structures may not necessarily be fixed but they may change as a result of discrete events. Interconnection structures where some objects are evolving and/or the structure may change as a result of events will be called *evolving*; otherwise, they will be called *fixed*. Evolving interconnection structures appear in many areas of engineering and non-engineering type systems and correspond to faults, redesign of processes, changes of requirements, etc. Typical forms of interconnection structure are those defined by graphs. Having defined the composition structure we may now define a composite object or a process as follows:

Definition (3.10): Let $\{B_i, i \in \rho\}$ be an ordered set of objects and F be an interconnection structure defined on them. The object defined by the given set $B_a = \{B_1, \dots, B_\rho\}$ will be also denoted by $B_a = B_1 \oplus \dots \oplus B_\rho$ and will be called the *aggregate object* of the set. If $*$ denotes the action of the interconnection structure on B_a , then the resulting object is called *composite object*, or *process* and it is represented by

$$P = [\{B_i\}, F] = B_a * F \quad (3.10)$$

□

A composite object may be represented in a similar way to that of an object and as it is indicated by Figure (3.4). A composite object will be called a *process* where there is a goal, functionality associated with it. The extended state vector \hat{x}_ρ is the aggregation of the corresponding vectors $\{\hat{x}_{B_i}, i \in \rho\}$, i.e., $\hat{x}_\rho = \hat{x}_{B_1} \oplus \dots \oplus \hat{x}_{B_p}$, but the relation $R_{\hat{x}_\rho}$ is in general different than those defined by the set $\{R_{\hat{x}_{B_1}}, \dots, R_{\hat{x}_{B_p}}\}$, since it is affected by the composition rule F . Regarding control and measurements, we have that

$$\underline{u}_\rho = \underline{u}_{B_1} \oplus \dots \oplus \underline{u}_{B_p}, \quad \underline{y}_\rho = \underline{y}_{B_1} \oplus \dots \oplus \underline{y}_{B_p} \quad (3.11)$$

The set of variables in $\{\hat{v}_{B_i}, i \in \rho\}$, $\{\hat{w}_{B_i}, i \in \rho\}$ are external for the constituent objects but they may, or may not enter the relations implied by F and thus they may become internal for the composite object P ; the variables entering the relations defined by F are called *active*, whereas those not affected by F are referred as *neutral*. Active variables enter relations and thus become internal for the object, whereas neutral variables remain unaffected and thus external for the composite object. By collecting all neutral components from the sets $\{\hat{v}_{B_i}, i \in \rho\}$, $\{\hat{w}_{B_i}, i \in \rho\}$ respectively we define vectors \hat{v}_ρ , \hat{w}_ρ which are external to the composite object. In fact, the components of \hat{v}_ρ express variables that are externally specified (either as control variables, or outputs from another process) and those in \hat{w}_ρ express products of some processing which may, or may not be measurable completely. The composite system is thus represented as shown below

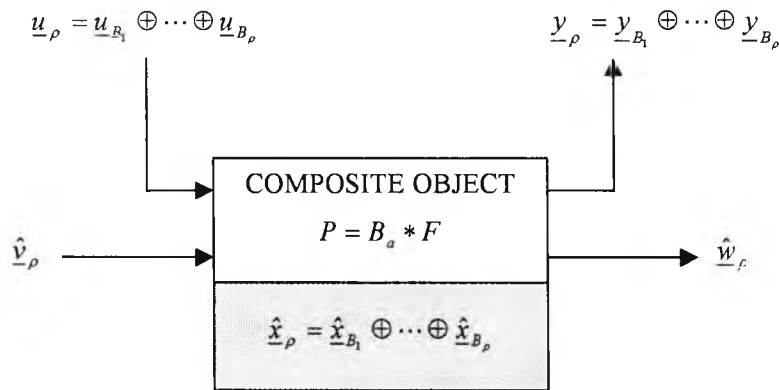


Figure (3.4): Composite object representation

The relations R_y, R_u, R_w, R_v, R_x associated with the composite object are shaped by the interaction of F with the aggregate object B_a and thus the resulting behaviours of the corresponding vectors are in general different to those simply defined on the aggregate object. The role of the interaction structure map F is thus crucial in determining the nature of the composite object P .

The nature of the process we are dealing with determines the specific properties of F and leads to the following general classification of them:

- (a) Natural maps
- (b) Design maps
- (c) Flexible maps

Such a classification is not related to the mathematical structure of F , but to the process that leads to the formation of the composite object. Composition of objects occurring as a result of phenomena beyond the control of a 'designer', or which emerge as unique solutions to design problems have a map F which is *natural*. In such cases, the properties of F are beyond the control of any designer and thus, issues of modification of them cannot be addressed due to lack of control mechanisms. This case includes physical, chemical, biological processes that occur in nature and cannot be influenced by human interaction. The crucial feature of this case is the lack of accessibility of the structure, that is ability to modify parts of it and thus its properties. A number of processes are entirely manmade and therefore, the corresponding map F is the subject of a design process; such maps are referred as *design* maps and issues related to their synthesis are significant. Many processes exist, either natural or designed, where there is some ability to modify them to improve properties without altering the original functionality, purpose of the process; we shall refer to them as *flexible* and for such processes redesigning F within a set of constraints is an important issue under consideration.

3.5 The system and its environment

Composite objects may be combined, according to some rule defined by an overall interconnection topology F' , and this leads to the definition of a system that may be represented in a general form as shown in Figure 3.5. If $S_a = \{p_1, \dots, p_v\}$

denotes the aggregate of processes interconnected under F' , then the system S is the interconnection of S_a and the F' topology and $\underline{x} = \hat{x}_{p_1} \oplus \dots \oplus \hat{x}_{p_v}$ is the system *state vector*, whereas vectors $\underline{u} = \underline{u}_{p_1} \oplus \dots \oplus \underline{u}_{p_v}$, $\underline{y} = \underline{y}_{p_1} \oplus \dots \oplus \underline{y}_{p_v}$ are respectively the input and output vectors, which are expressed as aggregates of those of the p_i subprocesses. The vector \underline{x} represents the complete knowledge on the internal mechanism of the system and its components express variables which may be identified down to the object level.

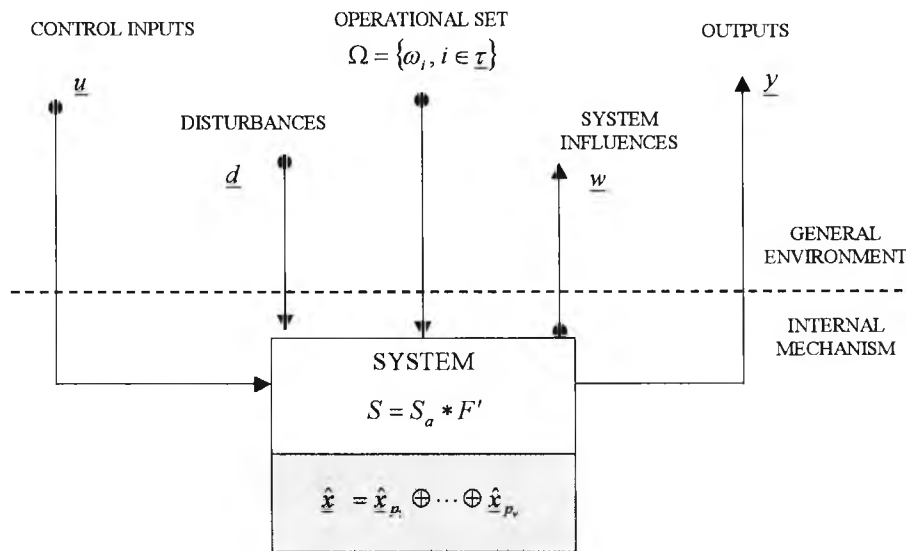


Figure (3.5): The System and its Environment

The interconnection topology is instrumental in determining the behaviour of \underline{x} . The space of all values of \underline{x} is referred as *state-space* of the system and is denoted by \mathcal{X} . There exist relations between the states of the system and this map is denoted by r and referred as the *internal relations map*. Such a map describes the evolution of the internal behaviour and it is affected by the behaviour of the processes in the aggregate system, the topology of interconnections F' and the coupling of the internal mechanism to the environment.

The vector \underline{u} represents the set of all external variables which affect the objects and processes in the system through some interface and which can be arbitrarily assigned externally. This vector will be called *input vector* and the space of all values of \underline{u} will be denoted by U and referred as *input space*. Similarly, \underline{y} represents all measurements performed on objects of the system and will be called *output vector*. The

space of all values of \underline{y} is denoted by Y and is referred as *output space*. The coupling of U to \mathcal{X} and \mathcal{X} to Y is expressed respectively and referred as *input-*, *measurement-* *maps* distinctively. The nature and properties of g, h express part of the interaction of internal mechanism and the environment and manifest the desire of the system designer to control, influence the system behaviour, as well as measure it.

The interaction of internal mechanism with the environment has also two other signal, event components. The external vector \underline{d} represents inputs, which affect the behaviour of the system states. However, the values of the components of \underline{d} cannot be arbitrarily assigned but are predetermined by other external processes. These variables take values from a set of signals, events, which are generated by some process, or may be unknown. We shall use the term *disturbances* for such vectors and the set of all values of \underline{d} is denoted by D and referred as *disturbance space*. The coupling of disturbances to the internal mechanism is denoted by relations, or map δ and it is referred as *disturbance map*. Note that the disturbances generated by known processes express the embedding of the given system in a wider context of interconnected systems and they will be referred as *loading disturbances*.

The variables that are measured express the knowledge extracted from the system. There exists, however, a vector of variables, which may be measured, or not measured and which affect other systems (they may become loading disturbances for other systems). These are denoted by a vector \underline{w} , which will be referred as *influence vector* and the set of values is a space denoted by W and called the *space of system influences*. The coupling of W to \mathcal{X} is expressed by a relation, map μ and it is called the *influence map*. Determining the nature of the map μ requires knowledge of the system, as well as of the environment within which the system is embedded.

A set of externally imposed rules, which represents objectives, performance indices, constraints, operational instructions is represented by $\Omega = \{\omega_i, i \in \underline{\tau}\}$ and will be referred as the *operational set* of the system. This represents higher-level functionalities, which affect the system behaviour, but not in a direct signal, or event way. The functionality of the system, as this is represented by higher-level goals, crucially depends on the nature of Ω set. The set Ω may contain rules, which affect the behaviour of individual processes, may alter the topology of interconnection to guarantee an alternative operational scheme, or may change to objectives, goals of the

system operation. This set is linked to the lifecycle aspects of the system and its elements and their functionality are defined at a higher level. In general Ω may be seen as the goal-setting governor of the system, which introduces the lifecycle aspects and impose alternative operational modes, goals and stimulate needs for redesigning the system. An overview of this system is given in Figure 3.6.

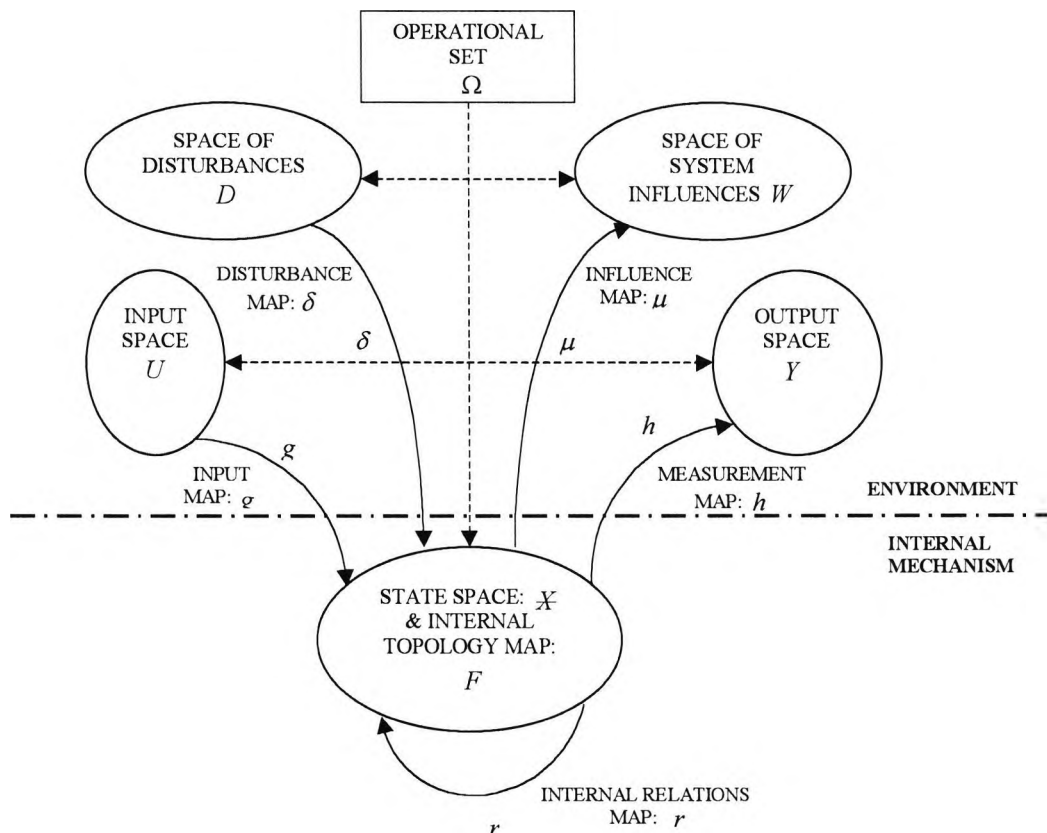


Figure (3.6): System Overview

3.6 System Behaviour, Dynamics and Qualitative System Properties

We consider the simplified system description of Figure 3.6, where we restrict our attention to the spaces (U, X, Y) and the corresponding maps $g: U \rightarrow X$, $r: X \rightarrow X$, $h: Y \rightarrow X$. Our interest here is to examine a richer subclass of the general family of systems, which is characterised by properties of the behaviour of the states, inputs and outputs associated with the system. Using properties of the behaviour, referred to the totality of variables, enables the introduction of important system properties, which in turn allow the introduction of systems' classification.

The behaviour of a variable is the set of all possible values of the variable obtained under a given initial value and expresses the time evolution of the values of the variables. The behaviour of the implicit states of an object is referred as *implicit behaviour*, or *implicit trajectory*. For relational objects the function f_B for which

$$f_B(\underline{\xi}_B) = 0 \quad (3.12)$$

expresses a relation that holds true for all implicit behaviours that may be associated with the object. An orientation on the relational object associated with the partitioning

$\underline{\xi}_B = [\underline{y}'_B; \underline{x}'_B; \underline{u}'_B]$ may lead to reduction of (3.12) into

$$r_B(\underline{x}_B, \underline{u}_B) = 0 \quad (3.13)$$

$$\underline{y}_B = h_B(\underline{x}_B, \underline{u}_B) \quad (3.14)$$

Such orientations will be referred as *solvable* and for them the notion of *state*, *input* and *output trajectory* are introduced as partitionings of implicit trajectory. If conditions (3.13) and (3.14) can be solved and produce the expressions

$$\underline{x}_B = \rho_B(\underline{u}_B, \underline{x}_B(0)) \quad (3.15)$$

$$\underline{y}_B = \sigma_B(\underline{u}_B, \underline{x}_B(0)) \quad (3.16)$$

where ρ_B , σ_B are uniquely defined and $\underline{x}_B(0)$ denotes the value of \underline{x}_B at some initial time, then the solvable orientation will be called *regular*. The relation ρ_B will be called *state transition* and σ_B as the *output transition*, or *$\underline{x}_B(0)$ -transfer function*.

In the following, we shall consider objects or systems for which there exists at least a regular solvable orientation. Such systems may be classified as:

Definition (3.11): An object B , or system Σ with a regular solvable partitioning, will be called *dynamic* if relations (3.15), (3.16) contain explicitly $\underline{x}_B(0)$; otherwise, they will be called *static*.

□

The meaning of a dynamical object or system is that the time evolution of the trajectories \underline{x}_B , \underline{y}_B are dependent not only on the \underline{u}_B trajectory, but also on the past history of the object state, as this is expressed by $\underline{x}_B(0)$. On the other hand, static objects or systems are characterised by relations (3.15), (3.16), which are independent from the initial condition $\hat{\underline{x}}_B(0)$ and thus they are of instantaneous nature. With

dynamical objects, systems, a crucial notion is that of equilibrium, which is defined below:

Definition (3.12): For a dynamic object B , or system Σ , a pair of constant vectors $(\underline{x}_B^*, \underline{u}_B^*)$ defines an *equilibrium*, if for $\underline{x}_B(0) = \underline{x}_B^*$

$$\underline{x}_B^* = \rho_B(\underline{u}_B^*, \underline{x}_B^*) \quad (3.17)$$

Furthermore, if $\underline{u}_B^* = \underline{0}$, the equilibrium will be called *free*, and if $\underline{u}_B^* \neq \underline{0}$ it will be called *forced*. □

The above definition characterises the equilibria as the fixed points of the state transition. This implies that equilibria correspond to pairs $(\underline{x}_B^*, \underline{u}_B^*)$ which when they are considered as inputs to state transition result in no movement at all. In the following, the trajectory that results from zero input, i.e., $\underline{u}_B = \underline{0}$, will be called *free motion*, whereas when $\underline{u}_B \neq \underline{0}$, the resulting motion as described by (3.15), (3.16) will be called *forced motion*. An important qualitative property of trajectories with respect to equilibria is that of stability, which characterises the behaviour of trajectories with respect to an equilibrium point. Introducing this notion requires the introduction of some topology on the spaces \mathbb{X}, U, Y , which may be general sets. Boundedness is reduced to a distance problem for each of the variables with a domain of values that may be a general set (signals, sequences, events, general sets). Similarly, defining regions for a point requires the definition of a distance function. Objects, systems, such that a metric topology may be defined on \mathbb{X}, U, Y , will be called *metric* objects, systems. For such systems, we define:

Definition (3.13): For a metric object B , or system Σ with a static equilibrium point \underline{x}_B^* we consider two spheres centred at \underline{x}_B^* with radii r, R , $\Omega(\underline{x}_B^*, r)$, $\Omega(\underline{x}_B^*, R)$ such that $r < R$. We may classify equilibria in the following way:

- (i) \underline{x}_B^* will be called *state bounded*, if for any $\underline{x}_B(0) \in \Omega(\underline{x}_B^*, r)$ the free trajectory $\underline{x}_B = \rho_B(\underline{0}, \underline{x}_B(0)) \in \Omega(\underline{x}_B^*, R)$ for all time.
- (ii) \underline{x}_B^* will be called *asymptotically stable* if it is *state-bounded* and $\underline{x}_B = \rho_B(\underline{0}, \underline{x}_B(0)) \rightarrow \underline{x}_B^*$ as $t \rightarrow \infty$.

- (iii) \underline{x}_B^* will be called *unstable* if no matter how small r is selected, there exists at least one $\underline{x}_B(0) \in \Omega(\underline{x}_B^*, r)$ such that for some time $t > \tau$ the free trajectory $\underline{x}_B = \rho_B(0, \underline{x}_B(0))$ escapes $\Omega(\underline{x}_B^*, R)$.

□

The above notions are expressions of the standard definitions of internal stability of dynamical systems expressing notions of Lyapunov stability, instability. In this more general setup, however, the selection of appropriate metric topology is crucial in defining the notions. Such topologies have to be natural and be linked to the specific characteristics of the object under consideration. The definition above may be readily extended to a characterisation of stability of forced motion when \underline{u}_B is a fixed input and we consider variations in the initial conditions $\underline{x}_B(0)$. An alternative notion of stability based on input-output properties is defined below:

Definition (3.14): For a metric object B , or system Σ , we may define alternative notions of stability as shown below:

- (i) The system is *Bounded-Input, Bounded-Output stable*, or simply *BIBO-stable*, if for all bounded inputs \underline{u}_B and for $\underline{x}_B(0) = \underline{0}$, the forced output trajectory $\underline{y}_B = \sigma_B(\underline{u}_B, \underline{0})$ is bounded.
- (ii) The system is *totally stable*, or simply *T-stable*, if for any bounded input \underline{u}_B and any $\underline{x}_B(0)$ bounded (within a given set), the state and output trajectories $\rho_B(\underline{u}_B, \underline{x}_B(0))$, $\sigma_B(\underline{u}_B, \underline{x}_B(0))$ are bounded.

□

The characterisation of such properties depends on the nature of the system and the selected metric topology. Different types of criteria may be derived for classes of models representing families of systems. The notions of stability have been presented in an abstract way aiming to cover all families of metric objects. Characterising such properties in terms of criteria requires use of models for the different types of systems and it is beyond the scope of this work.

For families of solvable partitionings of (3.13), (3.14) type, it may be possible to eliminate \underline{x}_B from (3.14) and derive a uniquely defined relationship between \underline{y}_B and

\underline{u}_B , that does not involve \underline{x}_B . Then (3.13), (3.14) may be represented in an equivalent manner as

$$r_B(\underline{x}_B, \underline{u}_B) = 0 \quad (3.13)$$

$$\underline{y}_B = \Phi_B(\underline{u}_B) \quad (3.18)$$

and this description has been referred as *strongly oriented* in the input-output sense and Φ_B is defined as a *transfer relation*. The description (3.18) on its own does not necessarily provide a complete representation of the object behaviour. We may classify such descriptions in the following way:

Definition (3.15): A strongly oriented description will be called *complete*, if there is a procedure of reconstructing the relationship (3.13) from (3.18). Otherwise, it will be called *incomplete*. □

Completeness, thus refers to the ability to recover a relationship between internal variables, states, from input-output or transfer relationship. Assessment of presence of such property requires use of specific features of the particular objects. For complete objects Φ_B is adequate to describe the object in the input-output sense. Objects for which the initial state $\underline{x}_B(0) = \underline{0}$ are referred as *relaxed*. An important feature of the dynamic behaviour that can be discussed in terms of the transfer relation is defined below:

Definition (3.16): Consider the strongly oriented object, or system represented by the transfer relation Φ_B . It will be called *causal* or *non-anticipatory*, if the output of the system at time t does not depend on the input applied after time t ; it depends only on the input applied before and at time t . Otherwise, it will be called *non-causal* or *anticipatory*. □

Causality, in short, implies that the past affects the future, but not conversely. Hence, if a relaxed object is causal, its transfer relation can be written as

$$\underline{y}_B(t) = \Phi_B(\underline{u}_B(-\infty, t)), \quad t \in (-\infty, \infty) \quad (3.19)$$

The output of a non-causal system depends not only on the past input, but also on the future value of the input. This implies that a non-causal system is able to predict the

input that will be applied in the future. For real physical systems, this is impossible. However, for processes involving human operators, or some form of intelligence, non-causality may be a naturally observed property.

Two important properties related to the family of state, output and input trajectories in relation to the spaces \mathbb{X}, Y and U , are those expressing ability to transfer the object state between two points of \mathbb{X} by some appropriate input, and the ability to reconstruct the initial state of the object by knowledge of the input and output trajectory. These properties are defined below.

Definition (3.17): Consider a dynamic object B , or system Σ , with state, output trajectory families as in (3.15), (3.16) defined for all possible inputs \underline{u}_B . We define:

(i) The object or system as *reachable*, if given any two points $\underline{x}_B^1, \underline{x}_B^2 \in \mathbb{X}$, there exists an input trajectory $\underline{u}_B \in U$ such that

$$(ii) \quad \underline{x}_B^2 = \rho_B(\underline{u}_B, \underline{x}_B(0) = \underline{x}_B^1)$$

and this occurs in finite time. If there exists a pair $(\underline{x}_B^1, \underline{x}_B^2)$ for which this property does not hold true, then the system will be called *non-reachable*.

(ii) The object or system is *reconstructable* if knowledge of the input and output trajectories over a finite time allows the reconstruction of the state trajectory and thus, also the initial state of the system; otherwise, the system is called *unreconstructable*. □

The characterisation of these properties in terms of specific criteria requires the consideration of particular classes represented by specific families of models.

3.7 Control Concepts and Principles

The behaviour of a system under external inputs and disturbances has to fulfil certain objectives that are set externally by the designer or user of the system. To achieve these objectives, actions are required and this is the task of the Control Design. For the system described diagrammatically in Figure 3.7, the Control Problem is defined as follows:

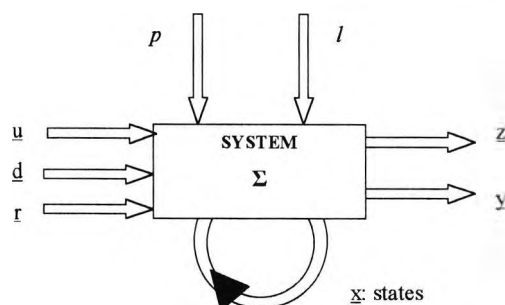


Figure (3.7): Open-loop Configuration

Open-Loop Control Problem: Given a set of objectives p , representing certain performance indices criteria, a set of process constraints l , a disturbance vector \underline{d} , a reference or target vector \underline{r} , design a control input vector \underline{u} such that the controlled variable vector \underline{z} follows the specified reference vector \underline{r} , whereas the states, inputs and measured variables satisfy the constraints l and the performance indices p are optimised in some sense.

The above is clearly a decision-making problem and selecting \underline{u} amongst all possible inputs has to take into account the understanding we have about the system model (used in the selection process) and requires exact knowledge of the disturbances over the lifecycle of operations of the system. The model and disturbance uncertainties make the above open-loop configuration rather unrealistic, and this has led to the following configuration, referred as closed-loop, which involves the use of feedback.

Closed-Loop Control Problem: Given the set of objectives p , constraints l , disturbance vector \underline{d} and target vector \underline{r} the design of the required control input \underline{u} is to be achieved by using the alternative feedback configuration shown below (Figure 3.8). In this configuration, the desired input is produced as the output of a new dynamical system, defined as the *Controller*. The controller is using as inputs the command signal \underline{r} and the real measurement of the command variable vector. The controller acts on the error signal and this generates the required input vector \underline{u} .

The use of feedback, that is measurement of required response and comparison with ideal response, enables the overcoming of difficulties associated with model uncertainty and lack of knowledge on the lifecycle behaviour of disturbances. In this closed-loop configuration, the design task now becomes a procedure for synthesis-design of a new system, the controller. Systems Theory provides the analysis

concepts and tools, whereas Control Theory deals with the solvability of certain types of Control Problems, which are posed in the context of specific types of models used to represent system behaviour. Control Design uses both Systems and Control Theory and deals with the derivation and implementation of control schemes which achieve the overall control objectives.

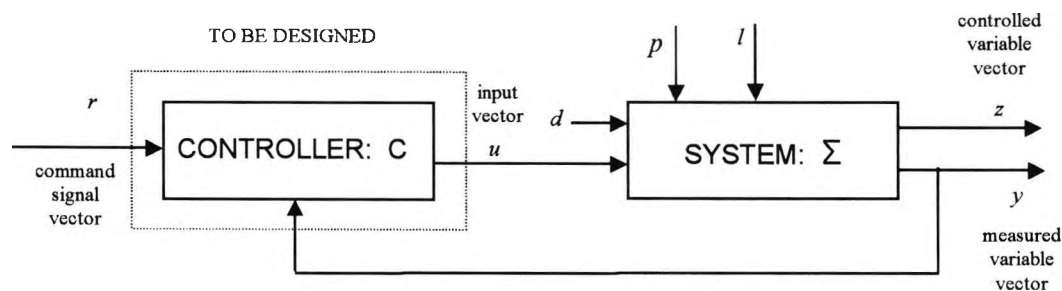


Figure (3.8): Closed-Loop, or feedback configuration

Measurement of the system variables for monitoring and/or control purposes may not be a straightforward process and this may require some additional design task described below:

Estimation Problem: Given the system, which is driven by a control input vector \underline{u} , measure all physically possible output variables \underline{y} through a measurement system. This measurement system produces measurements \underline{z} , which are defined by \underline{y} , but they may be corrupted by disturbances, or noise signal vectors \underline{d} . The problem of estimating the true system states then becomes a design problem where a new dynamical system is to be constructed, called the *estimator*, which produces estimates of the true states in an asymptotic sense. The estimator is provided with information about the system model, measured outputs \underline{z} and driving control signal \underline{u} and produces the estimated state vector $\hat{\underline{x}}$. The overall configuration is denoted in Figure 3.9.

State estimation is part of the overall problem of measurement and diagnostics, where certain aspects of the internal behaviour of the system have to be estimated, or quantitative functional characteristics have to be computed (for instance, quality parameters, critical values for behaviour of other processes etc). There is a classical duality between the Control and Estimation problem, which is linked to the duality between Reachability and Reconstructability on a given system.

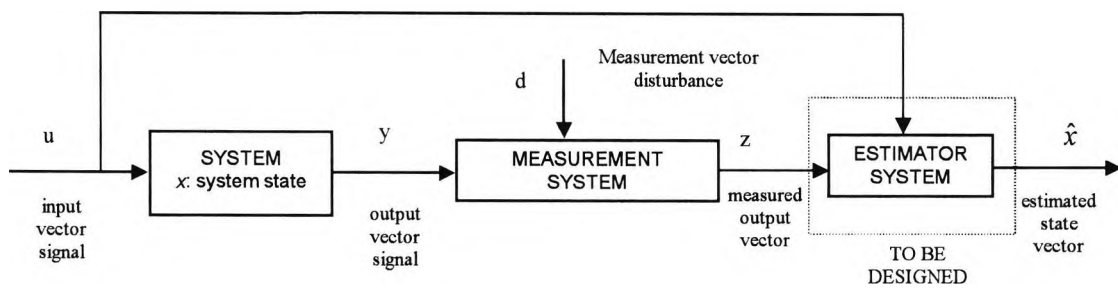


Figure (3.9): Diagram of the state estimation problem

A more general configuration representing the Control, Estimation, modelling tasks and higher strategies (embedding of a system whose control structure is a hierarchy of decision making functions) is described below:

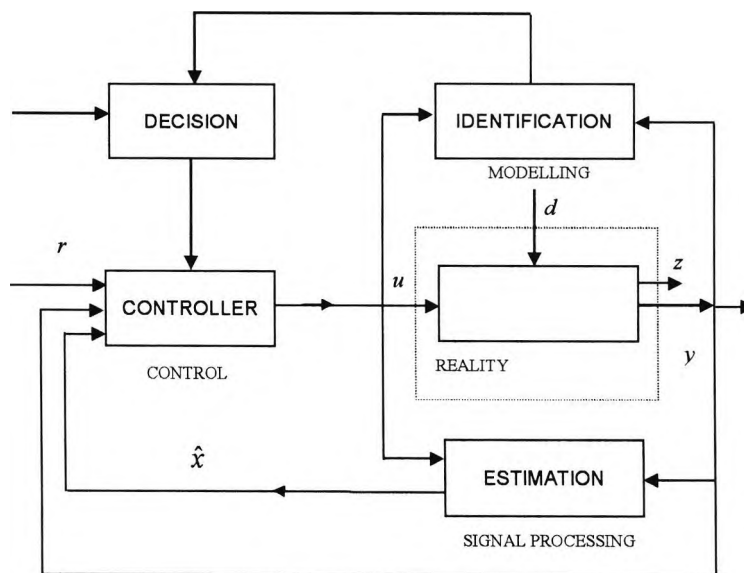


Figure (3.10): General Control, Modelling, Estimation and Coordination Configuration

The additional elements in this configuration are the tasks of *Identification* of models of the system and the *Decision-Making* process. Identification of models is linked to model derivation based on structural and input, output behaviour information and it is a task within the overall modelling exercise. The Decision-making block is the aggregate representation of overall higher layer coordination (see next section on hierarchies) and produces the higher level originating goal, objectives driving the control action. The controller in this extended configuration is now provided also with

inputs from Design and Coordination, as well as real and estimated variables behaviour. The above scheme is typical for engineering type problems, but provided with appropriate descriptions for the key blocks, also defines a general set up for decision, measurement, diagnostics and control of general processes.

3.8 The Integrated Enterprise: Global Diagnostics and Global Controllability

The study of multicomponent and multifunctional systems requires methodologies for managing complexity and this is essential for both analysis and design purposes. The discussion of the issues in this area is eased by the use of a paradigm. We shall use here the paradigm of Integration of Operations defined on an industrial establishment (in this case a continuous production process) and this is briefly described below [Karc., 3]. Following this, we proceed to the discussion of specific issues.

3.8.a Description of the Integrated Operations

The operation of production of the types frequently found in the Process Industries relies on the functionalities, which are illustrated in Figure 3.11. Such general activities may be grouped according to certain criteria described below (see also [Morari *et al.*, 1]):

- (a) Enterprise Organisation Layers
- (b) Monitoring functions (i.e. measurement, assessment) providing information to upper layers.
- (c) Control functions setting goals to lower layers.

Note that the process unit with its associated Instrumentation (sensors and actuators) is the primary source of information. However, processing of information (definition of diagnostics) can take place at the higher layer. Control actions of different nature are distributed along the different layers of the hierarchy (control and decision problems). The functions shown in Figure 3.11 are of the following type [Rijnsdorp, 1]:

1. Operations Planning: This refers to activities such as feedstock negotiation and acquisition, customer orders, resource planning etc.
2. Production Scheduling: This is concerned with the optimal timing of different operations runs and involves the combination of feedstock types and specification of the required type/quality of end-products from all production locations
3. Load Allocation: This involves the setting of the loads of the processing and utility plants of the overall production unit, such that they satisfy the production scheduling constraints.

The above three activities are known as Logistic type of activities and deal with general issues of production. As such they are also present in other industrial, or commercial activities, apart from continuous processes. In the latter case, however, such functions are strongly connected with the technical operations described below:

4. Recipe Setting/Initialisation/Correction: This is the higher layer of supervisory activities and deals with the co-ordination of the “mode” of operation defined as the set of conditions required for producing the desired products.

The above activity is technical, it is referred as *Desired Operation*, and it is a set of technical procedures required to produce the desired product. Although the procedure is determined in a general way (a priori knowledge), there is room for the fine setting of parameters and this involves some higher-level control action. The main layer of technical supervisory control functions involves the following group of functions:

5. Quality Analysis and Control: This involves the measurement, estimation of the important quality variables and attributes and then the initiation of corrective actions when product quality deviate from the set standards.
6. State Assessment, Off-Normal Handling and Maintenance: This set of activities is linked to the estimation of the actual “state” of the process based on all available information. In case of detection of off-normal process conditions there is a need to implement procedures to remedy the situation. In the case of emergency, the Emergency Shut Down Systems provide Emergency protection. The set of all these activities are linked to maintenance and in particular predictive maintenance activities.

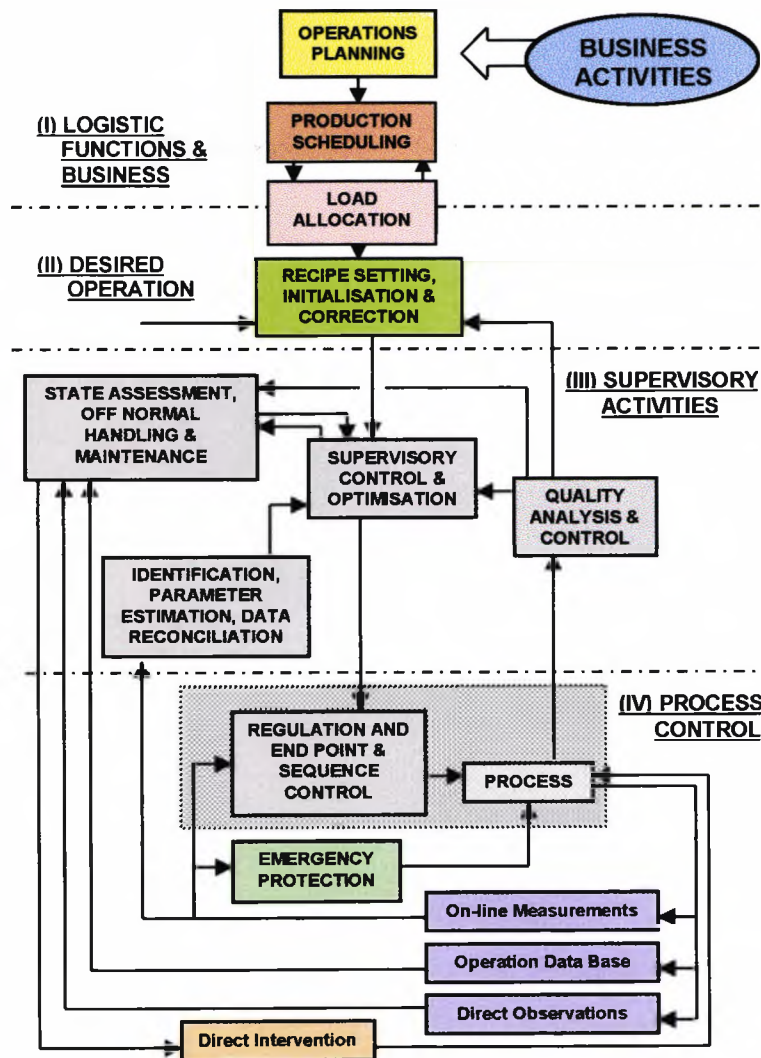


Figure (3.11): Functions for Operations of Process Plants

7. Supervisory control and Optimisation: Integrating the results from desired operations, quality analysis, state assessment and the general business objectives (coming from the higher business layers of the hierarchy), as well as taking into account the operational constraints (physical) and regulatory constraints (safety, environment etc) to produce an optimal policy, are the aims of the current task. This activity produces as output the optimal set points for the physical operation of the process.
8. Identification, Parameter Estimation, Data Reconciliation: The off-line and on-line control activities require models and relevant data that can lead to the identification of such models. Part of the supervisory activity, in collaboration with the design team, is the selection of the data, their validation, and then the identification of model parameters. Such an activity provides links with design, as well as

model-based diagnostics. This area is part of a wider activity referred as Data Management.

The above activities are of supervisory nature and refer to the Control room; the role of the process operator is to supervise and integrate all such activities. The automated part of the physical process is referred here as Process control and involves:

9. Regulation, End Point and Sequence Control: This refers to the regulating control loops, usually embedded in the Process Control and Data Acquisition (PCDA) systems (i.e., DCS). Direct intervention on the process from the Control room is also included here.
10. Emergency Protection: This refers to the Emergency Shut Down Systems.
11. Process Instrumentation and Information System: This refers to the overall system for on-line Measurements, Creation of the System Data base and may involve direct Observations, Data Storing and Management.

It is apparent that the complexity of operating the production system is very high. A dominant approach as far as organising such activities is through a Hierarchical Structuring [Mes. *et al.*, 1]. However, other forms of organisation are emerging at the moment, [MacFarlane, 2], but their full potential has not yet been evaluated in the context of process systems. The study of systems and modelling issues depends on the organisational form that is adopted. Here, we will restrict ourselves to the Hierarchical organisation paradigm [Mes. *et al.*, 1].

3.8.b Modelling Classes in Process Operations

The study of Industrial Processes has as main aspects those related to: (a) Design of Processes and (b) Operation of Processes. For the study of both areas we require models of different type, which are tuned to the respective goals of the particular function, design phase. The borderlines between the families of Operational Models (OM) and Design Models (DM) are not always very clear and frequently the same model may be used for some functions. Models linked to design are "off-line", whereas, those used for operations are either "off-line", or "on-line". In the strict sense these two types are not linked; however, in automated processes, on-line data can be used for revalidation and updating of off-line models and thus the usual distinction between off-line and on-line models tends to become blurred. For process type applications, models are now classified

into two main families referred as "line" and "support" models [Rijnsdorp, 1]. Line models are used for determining desired process conditions for the immediate future (set points for regulatory control etc.), whereas support models provide information to control models (i.e. parameter values), or they are used for simulation purposes.

Another major classification of models are those referred as "black" and "white" models [Rijnsdorp, 1]. White models are based on physical, chemical and/or biochemical principles and their development requires a lot of process insight and knowledge of physical/chemical relationships. Such models can be applied to a wide range of conditions, contain a small number of parameters and are especially useful in the process design, when experimental data is not available. Black models on the other hand are based on standard relationships between input and output variables containing many parameters, require little knowledge of the process and are easy to formulate; however, such models require appropriate process data and they are only valid for the range, where data are available. Black models can be turned to grey ones [Rijnsdorp, 1], if we know the ranges of process variables; hybrid, "White/black" models also may arise, when part of the model is white whereas difficult parts (such as chemical reactions etc.) are modelled as black models.

3.8.c Organisational Issues of the Integrated Operations

The overall problem of Process Operations is multidisciplinary and it is characterised by a high degree of complexity. The natural way of handling high complexity is through aggregation, modularisation and hierarchisation [MacFarlane, 1], and this is what characterises the overall OPPCP structure described in Figure 3.11. To be able to lump a set of components, subsystems together and then treat the aggregate, composite structure as a single object with a specific function, the sub-systems must effectively interact. Modularisation refers to the composition of specific function units to achieve a composite function task. Aggregation and modularisation refer to physical composition of subsystems, set of subsystems through coupling, and it is essentially motivated by the needs of design of systems with dedicated operational function. Hierarchisation on the other hand, is related to the stratification of alternative behavioural aspects of the entire system and it is motivated by the need to manage the overall

information complexity. The production system may be viewed as an information system and thus, notions of complexity are naturally associated with it [MacFarlane, 1].

Hierarchisation has to do with identification of design and operational tasks, as well as reduction of externally perceived complexity to manageable levels of the higher layers. At the highest level of the hierarchy, we perceive and describe the overall production process as an economic activity; at this level we have the lowest complexity, as far as description of the process behaviour. At the next level down, we perceive the process in terms of set of interacting plant sections, each performing production functions that interact in such a way so as to give an object - the economic unit activity - the higher level of description. We can describe how the process at the lower level of logistic functions area works, if all the production units at this level of description effectively interact. At this level of description we will use similar amounts of complexity in describing the particular functions of different production functions. At the next level down we are concerned with specification of desired operational functions for each unit in a plant section and so on we can move down to operation of units with quality, safety etc. criteria and further down to dynamic performance etc. In an effectively functioning hierarchy, the interactions between systems, or units at lower level, is such as to create a reduced level of complexity at the level perceived above [MacFarlane, 1]. The hierarchisation implies a reduction of externally perceived complexity successfully, as we proceed up the hierarchy till the top level.

The natural way to specify the different types of models needed for OPPCP is to link them with the operational functions and thus follow the hierarchy described by Figure 3.11 and try to identify a clearer stratification of basic operational functions. A simpler representation of the overall operational hierarchy of Figure 3.11 is as shown in Figure 3.12, which is an extension of a standard process control hierarchy [Backx, 1] that incorporates all functions described in Figure 3.11. Each of the above levels is characterised by different activities and has the following modelling requirements [Vasil., 1].

0-level: (Signals, Data Level). Physical variables, Instrumentation, Signal processing, Data Structures.

1-level: (Primary Process Control). Time responses, simple linear SISO models.

2-level: (Dynamic Multivariable Control). Linear, Nonlinear Multivariable Dynamic models.

3 level: (Supervisory Control Level). Process Optimisation Models, Statistical Quality Models, (SPC, Multivariate, Filtering–Estimation), Fault Diagnosis Models, Overall Process State Assessment Models (Heuristics, Neurofuzzy, Qualitative, etc.).

4 level: (Plant Operation and Logistics). Nonlinear Static or Dynamic Models for Overall Plant, Operational Research Models (Queuing etc.), Discrete Event Models (Petri Nets, Languages, Automata).

5 level: (Global Production Planning Level). Production Models, Planning, Forecasting, Economic Models, Operational Research, Game Theory Models.

6 level: (Business Level). Enterprise, Business Modelling, System Dynamics, Forecasting, Structural, Graph Models, Economic Models, etc.

The overall range of models mentioned above may be classified into the following main classes [Rijnsdorp, 1]:

- (i) Models for Logistic and Recipe Functions (MLRF)
- (ii) Models for Quality and Regulatory Control, and for Process Information (MQRCPi)
- (iii) Models for the Supervisory Functions (MSF)
- (iv) Models for Data Reconciliation (MDR)

A fundamental problem in modelling is understanding the way the different functional models are derived as well as how they are interfaced. We shall refer to this problem as the *Functional Model Derivation and Interfacing* (FMDI). The different types of models in the above groupings are interrelated. For each of these families detailed models on the unit level are simplified and aggregated to models on the plant level and then on the production site, business unit and possibly the enterprise level. Model composition accompanied by simplification is the dominant feature of models specified above. The latter classification is referred as functional and the Process Control Hierarchy implies a nesting of models.

The diagram in Figure 3.13 indicates that at the level of the process we have the richest possible model in terms of signals, data, full dynamic models. Then, as we move up in the hierarchy, the corresponding models become simpler, but also more general since they then refer not to a unit but to a section of the plant etc. The operation of extraction of the simpler models is some form of projection, whereas wider scale models are obtained by using plant topology and aggregations. The mechanisms of projection depend on the particular function the model addresses and they are not

always well understood. These models, although of different nature and scope, are related, since they describe sections, or aspects of the same process. Dynamic properties of subsystems are reflected on simpler, but wider area models, although not in a straightforward way. This is what we may refer to as *Embedding of Function Models (EFM)*.

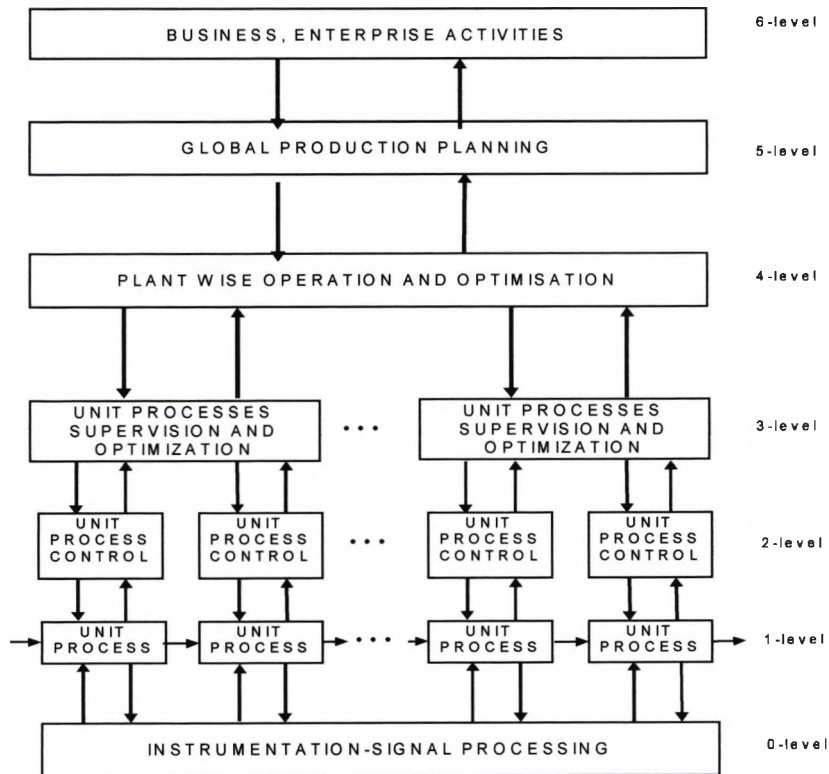


Figure (3.12): Overall Operational Hierarchy

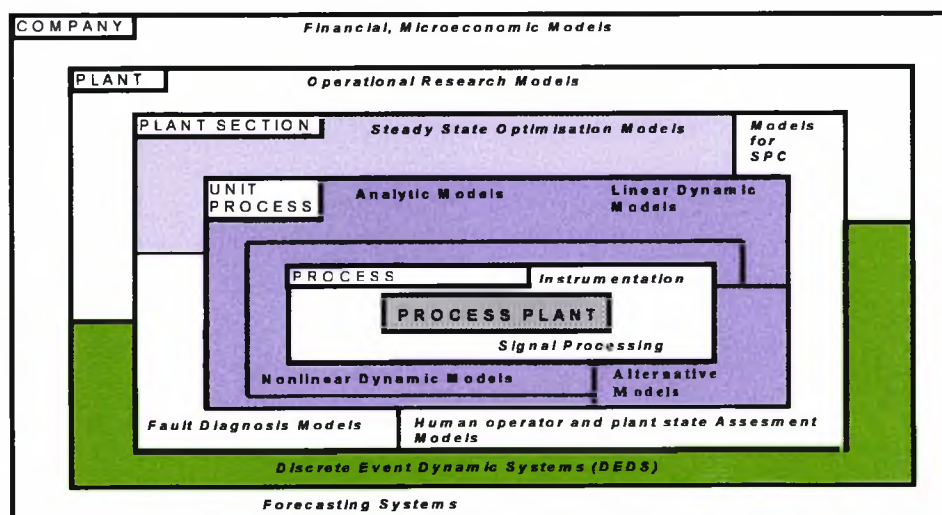


Figure (3.13): Model Nesting in Process Control Hierarchy

3.8.d Global Control and Measurement Issues

The hierarchical model of the Overall Process Operations involves processes of different nature expressing functionalities of the problem. Such processes are interlinked and each one of them is characterised by a different nature model. We adopt an input-output description of each of the subprocesses, with an internal state expressing the variables involved in the particular process and inputs, outputs expressing the linking with other processes. Such a model is generic enough to be used for all functionalities described in Figure 3.11 and can take a specific form determined by the nature of the specific process. We may adopt a generic description for the various functions as shown in Figure 3.14, where u_1, \dots, u_p denote independent manipulated variables of the function model, called system inputs; y_1, \dots, y_m are the independent controlled variables that can be measured and they are called the system outputs d_1, \dots, d_q are the exogenous variables which cannot be manipulated, but they express the influence of external to the particular function variables and they are called disturbances. A model describing the relationships between the vectors, \underline{u} , \underline{d} , \underline{y} is expressed as $y = H(u;d)$ where H expresses a relationship between the relevant variables, and it is called an input – output model. The construction of such a model is a major problem and involves a number of issues which may be classified as:

- (i) For a given function, establish a conceptual model for its role in the operational hierarchy.

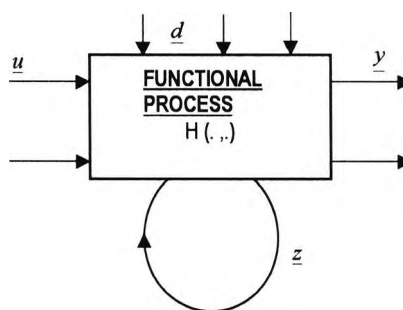


Figure (3.14): Generic Function Model with Internal Structure

- (ii) Define the vector of internal variable \underline{z} associated with a given problem and determine its relationships to input-output vectors by using any physical insight that we may possess about the functioning of the internal mechanism.

- (iii) Establish the relationships that exist between the alternative vectors \underline{z} associated with problems of the operational hierarchy.
- (iv) Define the appropriate formal model (using inductive and/or deductive modelling methodologies), to provide an adequate description for the H functional model.

The above generic steps are providing an approach, which however, involves many detailed modelling tasks. Typical problems here are issues such as classification of variables to inputs, outputs, disturbances, internal variables [Karc., 2], specification of formal description for H, definition of performance indices etc. When the classification of internal variables is completed, the key issue is the establishment of relationships between such variables; such relationships may be classified to implicit and explicit (oriented) forms respectively as:

$$M(u, y, d; z) \begin{cases} F(\underline{z}, \underline{u}, \underline{d}) = 0 & \text{(IMPLICIT)} & (3.12) \\ y = G(\underline{z}, \underline{u}, \underline{d}) & \text{(ORIENTED)} & (3.13) \end{cases}$$

The nature of variables and the type of problem under consideration determines the nature of the F, G, functions. This model structure also shows how constraints $F(\underline{z}, \underline{u}, \underline{d})$ can be propagated from higher to lower levels. The selection of \underline{z} implies that the modelling exercise, expressed as an attempt to specify F, G includes the modelling of the interface of higher level operation to the level defined by \underline{z} . The model $M(\underline{u}, \underline{y}, \underline{d}; \underline{z})$ in (1), (2) will be referred as a *z-stage model*. The selection of the operational stage (i.e., logistics, scheduling, steady state optimisation, quality control, dynamic process control, state assessment, maintenance etc.) determines the nature of the internal vector \underline{z} and thus also of the corresponding \underline{z} -stage model. The dimensionality and nature of \underline{z} depends on the particular functionality under consideration. Describing the relationship between different stages internal vectors is an important problem and implies an understanding of interfaces between functions; this is closely related to the problem mentioned before Hierarchical Nesting, or Embedding of Function models. The fundamental shell of this hierarchical nesting architecture is described in Figure 3.15.

Note that the vector reference image \underline{r}_{i+1} of operational objectives of the (i+1)-stage is defined as a function of the ith-stage internal vector \underline{z}_i . A scheme such as the one described above, is general and can be used to describe the meaning of the

hierarchical nesting. Furthermore, such a scheme can be extended to describe relations between models associated with functions at the same level of the hierarchy, extend upwards to business level activities and downwards to the area of the physical process (Production Stage).

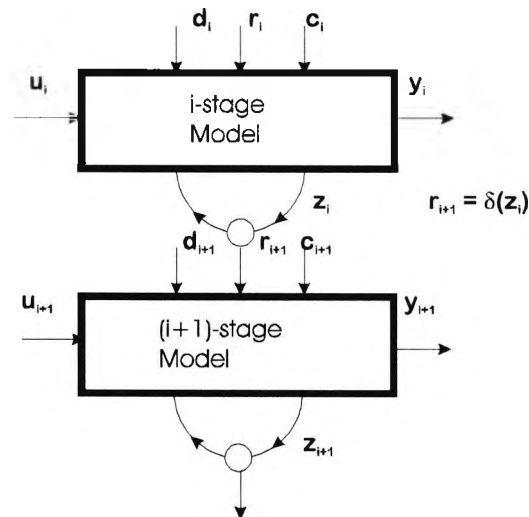


Figure (3.15): Nesting of models in the operational Hierarchy

The fact that each stage model in the hierarchy is of different nature than the others, makes the overall system of hybrid nature. It is clear that the theory of hybrid systems [Antsaklis *et al.*, 1], [Ner. & Kohn, 1] is crucial in the study of the control problems defined on the overall process hierarchy. Most of the work in the hybrid area has been concerned with two types of models; the characteristic of the present paradigm is that we have a multilayer hybrid structure. On this multilayer structure we have two fundamental problems:

- (i) Global Controllability Problem
- (ii) Global Observability Problem

The first, refers to the fundamental issue of whether a high level objective (possibly generated as the solution of a decision problem at a high level) can be realised within the existing constraints at each of the levels in the hierarchy and finally, at lowest level, where we have the physical process (production stage). This is a problem of *global controllability*, or alternatively may be seen as a problem of *realisation of high-level objectives* throughout the hierarchy. This open problem requires development of a multilevel hybrid theory and it can take different forms, according to the nature of the particular stage model. The Global Controllability problem described above, is

central in the development of *top-down* approaches in the study of hierarchical organisations, such as the process operations considered here.

The second problem refers to the property of being able to observe certain aspects of behaviour of the production layer of the hierarchy by appropriate measurements, or estimation subprocesses which are built in the overall scheme. This is a *global observability* property and is related to the ability to define *model based diagnostics* that can predict, evaluate certain aspects of the overall behaviour of the manufacturing process. It is assumed that the observer has access to the information contained at all stages of the model apart from the production layer, where only external measurement provides the available information. The Global Observability problem, is intimately linked to the *bottom-up* approach in the study of hierarchical organisations. The measurements, diagnostics defined on the physical process are used to construct the specific property functional models (quality, overall state assessment etc.) and thus, global observability (ability to observe all types of behaviour of the physical process), is linked to the quality of the respective functional model.

Integration of Operations requires study of fundamental problems such as Functional Model, Global Derivation and Interfacing, Model Embedding of Function Models, Global Controllability and Global Observability of the Process Hierarchy. These problems have links between themselves and establishing such links is also a challenging problem that may be referred as *Process Operations Design (POD)*. These problems have been hardly addressed from the Systems viewpoint so far, with the only exception the recent work on hybrid systems, which covers only partially some of the issues raised in the above problems. Of course, Process operations are based always on a physical system, process. Establishing the links between Operational criteria (desirable goals) and Engineering Design Objectives – criteria, is a major challenge and it is referred as *Operations–Design Interface (ODI)* problem. When operational objectives cannot be realised on the existing physical process, then the problem of Process Redesign arises and this is a problem that addresses together problems of Process Operations and Integration of Design simultaneously and can be considered within the current framework.

In summary, the area of Integrating Operations involves a number of fundamental problems of the Systems, Modelling, Control and Measurement type, which may be described as:

- (O.P.1)** Formulation of individual Process functionalities as dynamic decision making problems (particular problem aspects).
- (O.P.2)** Study of alternative forms of organisation of the Overall (extended) Process Operations and Business environment.
- (O.P.3)** Multimodelling aspects of the Integrated Extended Operations hierarchy and multilevel Hybrid Systems.
- (O.P.4)** Global Controllability of the Integrated Extended Hierarchy and realisation of policies, strategies.
- (O.P.5)** Global Observability of production process and Model based Diagnostics.
- (O.P.6)** Integrating design aspects of alternative process operations.
- (O.P.7)** Interfacing Operational issues and Engineering Design of the production process.

(O.P.1) involves the formulation of individual function studies in the standard control framework, whereas (O.P.2) deals with the alternative forms of organisation, rather than the traditional Hierarchy adopted here. The areas (O.P.3)-(O.P.7) have been already discussed. The term “extended operations” refers to that in the traditional manufacturing production operations we also include those linked to the business environment.

3.9 Conclusion

This chapter encapsulates the basic terminology upon which the theory of general processes can be built. The effort has been placed on the definition of the elementary concept of object as the fundamental element for the definition of composite objects, systems and their interconnections. In the overview of the system, as presented in Figure 3.6, we have included a special set of externally imposed rules, called the operational set, in order to introduce objectives, constraints, goals and performance indices to the description of the system. This creates a framework capable of addressing the issues emerging from the modelling of business processes.

The discussion continues on the dynamic and qualitative properties of the system, as well as on the major control concepts. Finally, the paradigm of Integration of Operations is presented here to address the issue of complexity in multicomponent

systems as well as the need to find appropriate methodologies that are able to manage complexity.

Chapter 4

GENERIC METHODOLOGIES FOR BUSINESS PROCESSES MODELLING AND DECISION MAKING: A REVIEW

4. GENERIC METHODOLOGIES FOR BUSINESS PROCESSES MODELLING AND DECISION MAKING: A REVIEW

4.1 Introduction

Developing a business information and decision-making system is a complex design task in which the final specifications cannot be known in advance. Requirements of modelling and analysis are characterised by informality and uncertainty. In addition, the modelling of dynamic behaviour of different types of business processes is underdeveloped and decision-making is mostly of heuristic nature. The quality of a conceptual scheme describing the overall process and, ultimately, that of the information system depends largely on the ability of the developer (designer) to extract, understand, and manipulate relevant knowledge. The methodologies available for describing the dynamic aspects of behaviour come from many and diverse areas.

This large number of approaches, methodologies of the structural type have been developed, having in mind, specific application areas [Karc. *et al.*, 1]. Frequently, their technology differs, but there is a common core and complementarity between the alternative approaches. Integrating these methods by identifying the common core and interpreting the special features is an important step for decision making for “soft systems” and business type problems.

This chapter lists some of the modelling methodologies and techniques, relevant to the business process issue. A fairly detailed account of the potentials of Qualitative Reasoning, Functional Reasoning, Goal-driven methodologies, and Bayesian Networks will be provided. These modelling approaches are referred here for suggesting further work on their capabilities and links to the manipulation of the business process modelling.

4.2 Modelling Issues of the General Business Problem

The modelling of the enterprise is clearly a multidisciplinary task. This task includes two key aspects: (i) the process of capturing large bodies of knowledge about the enterprise and (ii) the subsequent manipulation and abstraction of such knowledge into a “formal” description.

The first activity is known as *conceptual modelling* and it is a prerequisite for the development of subsequent formal models. Figure 4.1 depicts an abstraction of the activity of developing an information and decision making system that recognises the importance of first developing models that are oriented towards the understanding of a particular process.

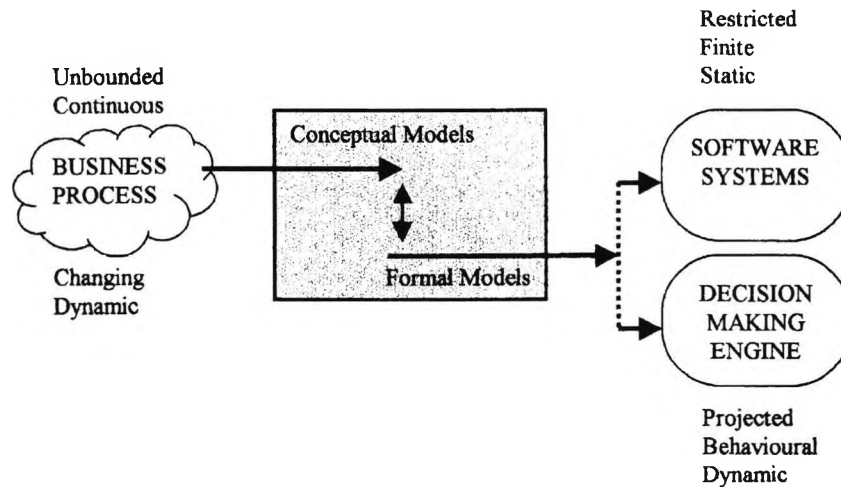


Figure (4.1): Conceptual Modelling and Development of the System ([Louc. & Zic., 1])

The development of formal models is considered as a following step that is achieved through a series of transformations and is always guided by design and decision making objectives. Figure 4.1 is based on a diagram describing the role of conceptual modelling in the development of software systems [Louc. & Zic., 1]. [Karc. *et al.*, 1] have added the Decision Making Engine (DME), which is viewed as a new important task. The purpose of modelling of the DME is to embody knowledge of the dynamic behaviour of the process and represent it in such a way so as to enable a decision-maker to reason about this knowledge, to communicate it with others, and modify the model accordingly.

Formal models are concerned with the abstraction, structure and representation of the observed process in a way that enables the reasoning about the system. A basic feature of complex systems is that each has structure associated with it. That structure might be quite obvious, as in the managerial organisation of a corporation, or less obvious, as for example in the value structure of a decision-maker. It is much more effective to deal with systems, whose structure is well defined.

4.3 Generic methodologies for business processes modelling and Decision making

Some of the methodologies, relevant to the modelling of soft systems, are:

(i) Conceptual Modelling: [Mylop., 1] This is an activity for formally describing some aspects of the physical and social world for purposes of understanding and communication. Such descriptions are often referred as *conceptual schemata*, and require the adoption of a formal notation, which is frequently called a *conceptual model*. Conceptual modelling has an advantage over the natural language or diagrammatic notations in that it is based on a formal notation, which allows one to “capture the semantics of the application”. It also has an advantage over the formal (or mathematical) notations because it supports structuring and inferential facilities that are useful for human communications. The development of conceptual models has benefited from contributions from fields such as Databases, Artificial Intelligence, Software Engineering [Louc. & Zic., 1], and Chemical Engineering [Douglas, 1]. Much of the work in these fields has progressed almost in an independent way, but many similarities exist amongst these models.

(ii) Soft Systems Theory: A number of efforts to extend classical notions of system theory to non-engineering problems have been made. The different approaches ([Klir, 1], [Vemuri, 1], [Skyt., 1], and [Check., 1]) in this area are closer to the problem formulation, but lack mathematical rigour and analytical capabilities. They are though significant at the conceptual level of modelling. Chapter 5 gives a rather detailed account of soft systems theory and their link to conceptual modelling.

(iii) System Dynamics Approach: For systems with totally unknown meta-laws the Forrester methodology [For., 2], referred as System Dynamics, allows the incorporation of any physical insight, that we possess about the functioning of the process, in the model. The class of methods here, includes concepts such as influence, or causal loop diagrams [Coyle, 1], structure diagrams [Karc., 1] etc. Its basic thesis is that several interconnected feedback loops can easily produce counter-intuitive behaviour, and therefore simulation can greatly increase our understanding of “soft” systems. Chapter 6 dwells more on the potential of System Dynamics.

(iv) Role-based models: In some business, and software process modelling approaches emphasis is placed on the organisational perspective. In Role Activity Diagrams

(RADs) [Ould, 1], [Ould, 2] the main unit of process structuring is the role. A process is a number of *roles*, and there is a distinction between *activities* which are local to a single role and those activities, called *interactions*, which involve the participation of several roles. Interactions represent those points where the roles need to co-ordinate their actions. For further analysis refer to chapter 7.

(v) Structural Methodologies: One of the simplest types of models-which carry the most basic form of knowledge-is that expressing physical or information connectivity. Essentially, these are models expressing topology, but many variations exist which combine topological together with some other additional properties. Linear graphs are the most common models, but the class is much larger and contains models such as Bayesian Networks [Jensen, 2], cause and effect diagrams, Petri Nets [Peterson, 2], Bond graphs etc. Structural modelling is well developed and is assisted by the powerful tools of graph theory. Such modelling is significant even for other forms of modelling based on behaviour. See chapter 8 for further discussion on Petri Nets.

(vi) Qualitative Physics: Qualitative physics aims to understand how humans model the behaviour of physical systems and thus, seeks to automate human reasoning about the physical world. The approach for analysing the system behaviour is based on a coarse, qualitative, structural description of the real physical system. Recent work focuses on expert reasoning about scientific and engineering domains and a good account of its status may be found in [Kuipers, 5]. The area uses AI concepts and has been deployed in the study of problems of process engineering [Mavro. & Steph., 2]. This translation between numerical and qualitative data is equivalent to a data reduction procedure. This area is primarily a topic within AI and contains important techniques known as qualitative simulation.

(vii) Functional Modelling: Given the ability to generate behavioural sequences for various subsystems and knowing the connectivities (structure), it is often possible to put together an account of the function of subsystems and their relationship to the system structure. This leads to functional modelling; note that although the behaviour can be the function, generally, functional specification involves teleology, i.e., an account of the intentions for which the process is used [Milne, 1]. The derivation of functional models may require the abstraction of behaviour to a level higher than that at which the subprocesses are specified. This abstraction process often involves a hierarchical organisation of representation of the relation between function and structure. In the area of Process Control, a method called Multilevel Flow Modelling [Lind, 5] has been

developed, aiming to handle issues of representation of goals and functions of complex industrial plants. The main idea behind this methodology is the use of functional concepts to represent a plant at multiple interrelated levels of abstraction. The mainstream development comes, however, from the area of AI and software engineering.

(viii) Goal-driven methodologies: One step to a more systematic approach of designing business processes is to develop models that provide appropriate representations of the knowledge that is needed for understanding and reasoning about business processes. Business modelling and goal modelling can be regarded as top-down methods, which successively decompose and satisfy single modelling states. Here, we will introduce an approach that uses goals as means to drive and manage the business modelling process [Yu & Mylop., 1].

The different modelling and analysis methods listed above offer powerful means of modelling specific aspects of the overall business-operations set up. In [Karc. *et al.*, 1], we find a far more expanded view of the different methodologies that can be applied. Specifying exactly what is the relevance of each particular methodology to the overall problem and how different approaches can integrate to provide methodologies that cover the needs of larger integrated parts, is a major challenge.

4.4 Part I: Qualitative Physics

4.4.1 Qualitative Reasoning

Qualitative Reasoning builds the conceptually unified theoretical framework for building and simulating qualitative models of physical systems, where knowledge of those systems is incomplete [Kuipers, 5].

Qualitative models are better able than traditional models to express states of incomplete knowledge about continuous mechanisms. *Qualitative simulation* guarantees to find all possible behaviours consistent with the knowledge in the model. It is a promising technique for design and analysis, particularly in model-based reasoning systems. Its purpose is to explain process observations by reasoning from physical descriptions to behavioural descriptions. Qualitative simulation has the ability to yield partial conclusions from incomplete knowledge of the process [Dalle Molle *et al.*, 1]. It

draws on a wide range of mathematical methods to keep a complete set of predictions tractable, including the use of partial quantitative information.

The *structure* of a system is described in terms of a collection of continuous parameters and constraints among them. *Behaviour* is described in terms of changes to position and direction in qualitative quantity spaces [Kuipers, 4].

4.4.2 History of qualitative reasoning

Several researchers have introduced domain-specific qualitative models into their expert system to enhance its performance ([Rieger & Grinb., 1], [Moh. & Sim., 1] and [Pan & Tenen., 1]). Other investigators have developed more general methods for qualitative reasoning about physical systems. The most noted of these methods are *Qualitative Physics based on Confluences* by [deKleer & Brown, 1], *Qualitative Process Theory* by [Forbus, 2], and *Qualitative Simulation (QSIM)* by [Kuipers, 1], [Kuipers, 2], [Kuipers, 3].

[deKleer & Brown, 1] concentrate on modelling individual physical components, and then derive the behaviour of a system of these components by using their connectivity to constrain the behaviour of the overall system. [Williams, 1] has contributed to this approach. [Forbus, 2] concentrates on modelling physical processes that are activated when the appropriate conditions exist and individuals are present. Both deKleer and Brown and Forbus develop libraries of these physical abstractions: *components* and *confluences* for [deKleer & Brown, 1], and *processes* and their *influences* for [Forbus, 1].

4.4.3 Qualitative reasoning by Forbus

One part of the problem is to represent how quantities change. [deKleer, 1]'s Incremental Qualitative (IQ) calculus can handle this. It represents the change in a quantity by one of four values (*U*, *D*, *C*, or *?*, corresponding to “increasing”, “decreasing”, “constant”, or “indeterminate”) indicating knowledge of the sign of the derivative. Although, this approach is very useful for causal reasoning, it cannot be used

to deduce the limits of physical processes. *Qualitative Process* analysis includes the IQ calculus, but also incorporates notions of rates and amounts [Forbus, 1].

A parameter of a physical system will be represented by a *quantity*. A quantity will have three components: an *amount*, an *IQ value*, and a *rate*. The last two specify how the quantity is changing; the IQ value corresponds to the sign of the change and the rate corresponds to the amount of the change. The quantities in a situation, which represent a particular kind of parameter, are grouped together in a partial order called a *quantity space*.

A *continuous process* acts through time to vary one or more quantities. The specification of a process has four parts: *preconditions*, *quantity conditions*, *relations* and *influences*. Both preconditions and quantity conditions must be true for a process to be acting. The *preconditions* are those factors that are external to the theory, such as someone opening or closing a valve. The *quantity conditions* are those limits that can be deduced within the present theory, such as for example, requiring the temperature of two bodies to be unequal for heat flow to occur. *Relations* hold between quantities affected by the process, and *influences* are the contributions to the way a quantity changes. Relations concern amounts and rates, while influences are assertions about the contribution to the IQ value for a quantity. In order to find out how a quantity is changing requires summing all of the influences on it, since several processes may be acting at once.

Discontinuous changes in processes occur at *limit points*, which serve as boundary conditions. The points are chosen according to the quantity conditions of the processes that can affect that parameter. For example, the temperature quantity space for a fluid could be:

$$T(\text{ice}) \rightarrow T(\text{boiling})$$

where temperatures at $T(\text{ice})$ and below correspond to the solid state, temperatures of $T(\text{boiling})$ and above correspond to the gaseous state, and any temperature in between to being a liquid.

The preconditions and quantity conditions can be used to determine what processes are operating within the situation. To infer the *limits* of a process, first we find the influences on all affected quantities and determine the resulting IQ value. Then, find the neighbouring points within the quantity space. If the neighbour is a limit point, some process may end there and others begin. Thus, the set of possible changes in orderings

involving limit points becomes the ways the current processes might change [Forbus, 1].

4.4.4 Qualitative simulation

The purpose of *qualitative simulation* is to reason from physical descriptions to behavioural descriptions and to explain process observations directly from a process model. The power of qualitative simulation is the ability to yield partial conclusions from incomplete knowledge of the process. Differential equations are the standard language for expressing such models. A representation language, which is a qualitative abstraction of differential equations, makes it possible to handle incompletely-known constraints or parameter values that would prevent a quantitative differential equation model from being formulated. The QSIM algorithm ([Kuipers, 1], [Kuipers, 2], and [Kuipers, 3]) starts with a mechanism model expressed as a set of qualitative constraints and predicts the possible qualitative behaviours of the mechanism. Figure 4.2 below portrays this level of abstraction.

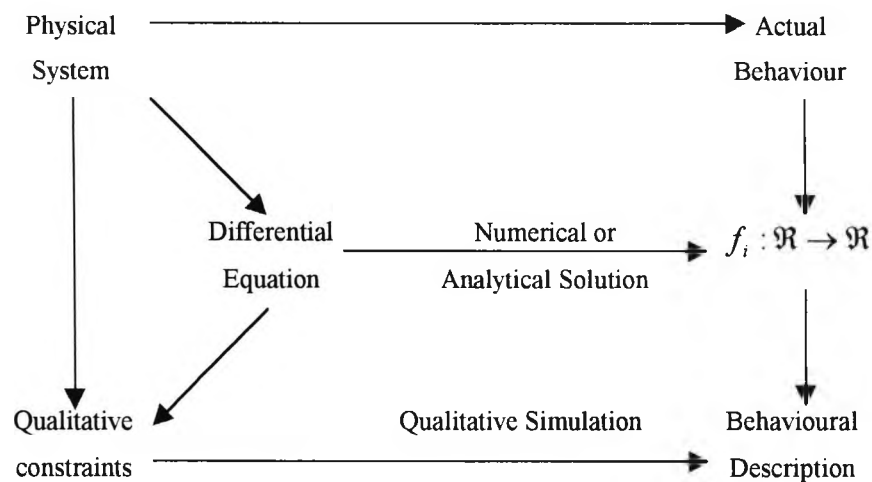


Figure (4.2): Modelling at quantitative levels of abstraction ([Dalle Molle *et al.*, 1]).

4.4.5 Basic concepts

The basic concepts of qualitative reasoning such as *quantity*, *landmark values*, *qualitative value* and *state* are given in the Appendix. Another important concept is the

differential equation, which represents the structure of the system by selecting certain continuous variables that characterise the state of the system, and certain mathematical constraints on the values those variables can take on [Kuipers, 5]. If a differential equation model adequately describes the structure of a physical system, the behaviour predicted by the solution to the differential equation will describe the actual behaviour of the physical system. The strength of the language of differential equations comes from the expressive power to state models that capture the dynamic character of the world, and the inferential power to derive predictions from those models.

A differential equation is an abstraction of the real world. Qualitative models are abstractions of differential equations. *Qualitative differential equations (QDE)s* and their behaviours are abstractions of ODEs (Ordinary Differential Equations) and their solutions (see Figure 4.3).

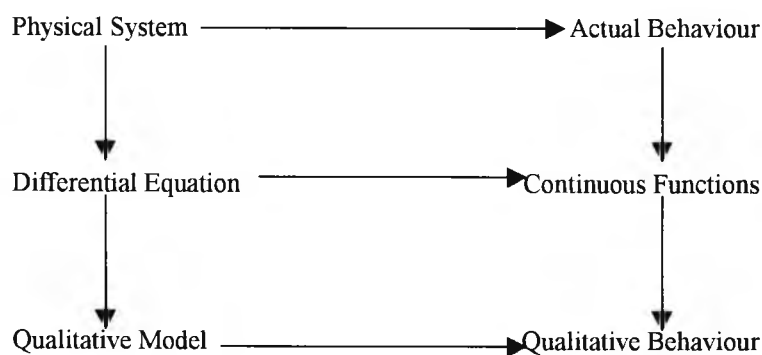


Figure (4.3): Qualitative models are abstractions of differential equations ([Kuipers, 5])

For an extensive and very detailed introduction to the concepts of qualitative simulation, refer to [Kuipers, 5]. Also, see the Appendix. Here, we will only add that *time*, within one possible behaviour, is represented as a totally ordered set of symbolic *distinguished time-points*. The current time is either at or between distinguished time-points. All the time-points are generated as a result of the qualitative simulation process. The predicted behaviours are best displayed graphically as *qualitative plots*, such as shown in Figure 4.4.

The qualitative simulation proceeds by determining all possible changes in qualitative value permitted to each parameter, then checking progressively larger combinations of qualitative transitions and filtering out the inconsistent ones.

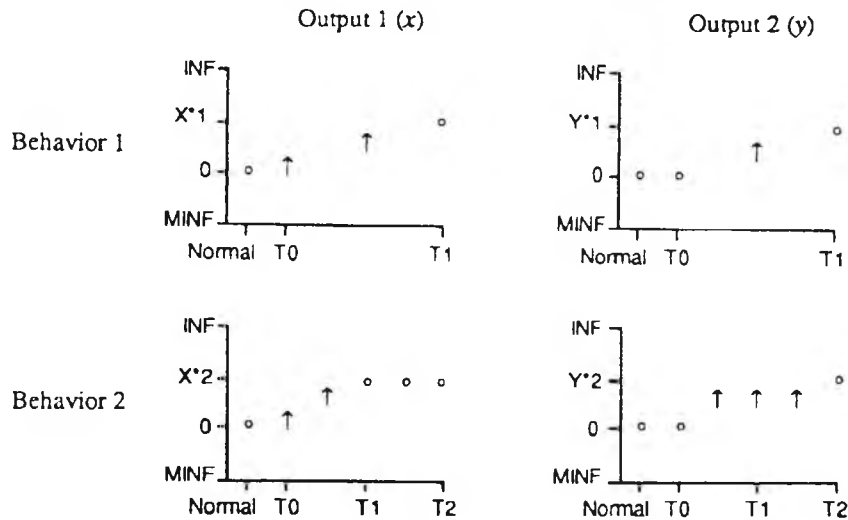


Figure (4.4): Qualitative step responses for a system of two first-order processes in series
([Dalle Molle *et al.*, 1])

4.4.6 Semi-Quantitative Reasoning

Semi-quantitative reasoning is the task of combining incomplete quantitative and qualitative knowledge. There are a number of different representations for incomplete knowledge of quantities, including bounding intervals, probability distribution functions, fuzzy sets, and order-of-magnitude relations. [Kuipers, 5] focuses on the use of bounding intervals $[lb, ub]$ where $lb, ub \in \mathcal{R}^*$, to represent partial knowledge of a real number $x_0 \in \mathcal{R}$:

$$range(x_0) = [lb, ub] \equiv x_0 \in [lb, ub] \equiv lb \leq x_0 \leq ub$$

Q2 (for Qualitative + Quantitative) is the basic semi-quantitative reasoner implemented as an extension to QSIM [Kuipers & Berleant, 1]. Q2 uses constraint propagation ([Kuipers, 5]) and interval arithmetic ([Moore, 1], [Moore, 2]) to tighten interval bounds on value-denoting terms.

The Q2 algorithm is given in [Kuipers, 5]. It does simulation by progressive refinement: define a set of possible solutions, and shrink it while maintaining the guarantee that the solution lies within the set. The cost of semi-quantitative inference with Q2 is not dependent on the size of the parameter space, but is a function of the number of distinct qualitative behaviours predicted by QSIM.

Q2 finds applications both in diagnosis and design. It serves as the *test* component in the water tank examples [Simmons & Davis, 1] and it is the basis for the MIMIC approach to monitoring and diagnosis of continuous dynamic systems ([Dvorak & Kuipers, 1], [Dvorak & Kuipers, 2], and [Dvorak, 1]). Q2 focuses on the task of unifying a given set of quantitative assertions with a given qualitative behaviour, relying on MIMIC to manage the set of possible interpretations.

4.4.7 Other representations

A sophisticated calculus for semi-quantitative reasoning based on order-of-magnitude relations has also been developed ([Raiman, 1], [Raiman, 2], [Mavro. & Steph., 1], [Mavro. & Steph., 2], [Murthy, 1], [Nayak, 1], and [Dague, 1]. The diagnostic system DEDALE [Dague *et al.*, 1] applies knowledge of order-of-magnitude relations. [Yip, 1] applies order-of-magnitude reasoning to simplify complex partial differential equation models.

Fuzzy representations for incomplete knowledge of quantities can also be used in conjunction with qualitative behaviours [D'Ambrosio, 1], [D'Ambrosio, 2], and [Vesconi & Robles, 1]. D'Ambrosio mentions two types of ambiguity in Qualitative Process theory, *internal* and *external* ambiguity. He proposes an extension of QP representations in order to capture more information about the system being modelled and, therefore, solve these problems. His extension utilises: (a) belief functions certainty representations – to permit capture of partial or uncertain observational data, (b) linguistic descriptions of influence sensitivities – to reduce undecidability during influence resolution, and (c) linguistic characterisations of parameter values and ordering relationships.

[Nordvik *et al.*, 1] have discussed the extension of qualitative simulation with fuzzy mathematics, and [Shen & Leitch, 1992], [Shen & Leitch, 1993] have fully implemented this extension.

Finally, qualitative and semi-quantitative knowledge about probability distributions have been investigated by [Wellman, 1] and [Berleant, 1].

4.4.7.1 Linking Petri Nets with Qualitative Reasoning

The simulation of a continuous or a discrete time system can be achieved, basically, in two ways: quantitatively and qualitatively. Quantitative simulation entails an exact description of the relationships between the various quantities involved in the system functioning, in terms, e.g., of algebraic differential or difference equations. Qualitative simulation, on the other hand, exploits relationships that express qualitative connections between the quantities [deKleer, 2], [deKleer & Brown, 1], [Forbus, 2], and [Kuipers, 3]. Of course, such a description does not contain as much information as a quantitative analysis, however, it permits to express incomplete knowledge and hence to handle systems that are not completely known. Also, it provides general solutions for classes of systems rather than numerical solutions of each case, and it treats linear and non-linear systems homogeneously.

One of the disadvantages of qualitative simulation is its fundamental ambiguity: given a qualitative model of a system and a set of qualitative inputs, more than one qualitative behaviour can generally be found. This fact partially depends on the choice of the quantity space used to represent the qualitative value of the variables. Another disadvantage of qualitative analysis derives from the fact that we lack effective simple mathematical tools for carrying out the simulation.

[Fanni & Giua, 2] propose a simple way of avoiding this problem. Since, a qualitative system, with its discrete quantity state space can be seen as a discrete event system (DES) [Ram. & Wonh., 1] (i.e., as dynamic system, in which the state changes only at discrete points in time, in agreement with unexpected occurrence of physical events), its behaviour may be described by using Petri nets models [Murata, 2]. Petri nets have been used in qualitative simulation by [Okuda & Ushio, 1], [Okuda & Ushio, 2]. These authors have noted that each *place* of a net may be associated to a state variable while the firing of each *transition* corresponds to crossing a landmark.

[Fanni & Giua, 2] modify the approach of Okuda and Ushio, and extend the approach of [Fanni & Giua, 1] and [Fanni *et al.*, 1]. First, they assume that each *marking* (not *place*) of a net may be associated to a state of a variable and that a transition may represent more than one landmark crossing. Second, in their approach the behaviour of the physical system is completely captured in the structure of the net.

This allows one to use some of the standard Petri net analysis techniques to study the properties of the model [Murata, 2], [Peterson, 2].

[Fanni & Giua, 2]'s approach has the possibility to model a hybrid system (i.e., system whose behaviour is described by both continuous and discrete event dynamics) as a mere discrete event system in the following way. First, a qualitative description of the continuous time behaviours is given. Second, the qualitative descriptions are captured by discrete event models. Finally, a model that integrates all these different discrete event behaviours is built using Petri nets.

4.4.7.2 Petri Nets as Qualitative models

The qualitative value of each variable is associated with the marking of a subset of places in the Petri net. Thus, we have a correspondence between qualitative states and markings [Fanni & Giua, 2]. The initial state of the system will determine the initial marking M_0 of the net.

The firing of a transition will represent the change of a qualitative variable from one qualitative value to another. The sequence of all possible behaviours is given by all sequences of transitions σ firable from the initial marking.

As we also know, the change of value of a qualitative variable, say x , is often dependent on the value of another one, say v . Thus, in the Petri net model a transition that changes the marking of the places associated to x may depend on the marking of the places associated to v . The influence of v over x may be represented by self-loops, i.e., cycles in the net graph containing only one place and one transition.

[Fanni & Giua, 2] present two different models for state variables in the quantity spaces $\{-, 0, +\}$ and $\{-n, \dots, 0, \dots, +n\}$. For each of these two cases, the authors give general construction algorithms to derive a Petri net model representing a given set of qualitative equations.

There are some advantages in using Petri nets to represent the qualitative behaviour of a system. First, there is simple and intuitive correspondence between the marking of the net and the state of the system. Second, the dynamic behaviour of the system can be studied as sequences of reachable markings of the net [Fanni & Giua, 2]. Third, the mathematical properties of Petri nets may be used to predict important characteristics of the system behaviour without resorting to simulation. Different Petri

net structures may be used to represent the same qualitative behaviour depending on the choice of the variable quantity space.

4.4.7.3 Advances

Q3 ([Berleant, 1], [Berl. & Kuipers, 1], [Berl. & Kuipers, 2], and [Berl. & Kuipers, 3]) extends Q2 with *step-size refinement*, interpolating new states into an existing sequence of states in a simulation trajectory, adaptively reducing the size of the time steps in the simulation.

Step-size refinement can only run within a gap. Constraint propagation is guaranteed to terminate when the label sets containing candidate values have a finite number of elements.

Q3 demonstrates an effective method of obtaining better quantitative bounds on semi-quantitative simulation trajectories, step-size refinement, which often leads to significant improvement in quantitative inferences after interpolating only one state.

4.4.8 Example:

The following example is taken from [Kuipers, 5]. It is a case study and a problem in economics. A paradigm of comparative statics problem involves reasoning about the intersection between *Supply* and *Demand* curves, as shown in Figure 4.5.

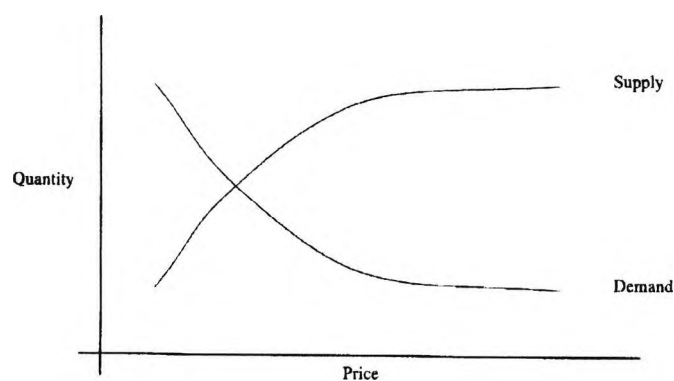


Figure (4.5): The Supply and Demand curves ([Kuipers, 5])

The horizontal axis of the graph represents the price paid per unit of some commodity and the vertical axis represents the quantity produced or consumed. The supply and demand curves represent the loci of equilibrium states of the supply and demand processes, respectively.

The supply curve increases monotonically because, as the price of the commodity increases, producers will tend to increase the supply in order to profit. The demand curve, on the other hand, decreases monotonically because as the price increases consumers will tend to buy fewer amounts of this commodity. The locations of the equilibrium curves are influenced by exogenous factors, such as cost of raw material to the producer, or perceived desirability of commodity to the consumer. A sudden change in exogenous factors can move the system off one or both equilibrium curves. Each process independently changes price and/ or quantity bought or sold, in order to return to equilibrium. Their joint effect will be to return the state of the system to the intersection of the supply and demand curves.

This model can be expressed by two qualitative equations:

$$\text{Supply: } \textit{quantity} = M^+(\textit{price} / \textit{XSF}),$$

$$\text{Demand: } \textit{quantity} = M^-(\textit{price} / \textit{XDF}),$$

where *XSF* represents the aggregate “external supply factors” and *XDF* represents “external demand factors”. The QDE implementation of this model is given in Figure 4.6 and Figure 4.7 shows the response of this system to a sudden increase in exogenous supply factors (*XSF*):

$$[\textit{XSF}]_* = [+] \Rightarrow [\textit{price}]_* = [+] \text{ and } [\textit{quantity}]_* = [-]$$

```
(define-QDE supply+demand
  (quantity-spaces
    (price (0 inf))
    (quantity (0 inf))
    (XSF (0 inf)) ; “external supply factors”
    (XDF (0 inf)) ; “external demand factors”
    (Sterm (0 inf)) ; price/XSF
    (Dterm (0 inf)) ; price/XDF
  )
  (constraints
    ((constant XSF))
    ((constant XDF))
    ((mult Sterm XSF price))
    ((mult Dterm XDF price))
    ((M+ Sterm quantity) (0 0) (inf inf)) ; supply curve: upward
    ((M- Dterm quantity) (0 inf) (inf 0)) ; demand curve: downward
  )
  (independent XSF XDF))
```

```
(defun increase-XSF ()
  (let* ((result (QSEA: from-qde      supply+demand
                    : initial-values ' ((price    ((0 inf)  std))
                                       (quantity ((0 inf)  std))
                                       (XSF      ((0 inf)  std))
                                       (XDF      ((0 inf)  std)))
         : perturb      '( (XSF+))
         ))
    (R (car result))           ; reference state
    (S (cadr result)))        ; perturbed state
  (QSEA-table S: reference R)
  (qsim-display S: reference-states ' ((R ,R))))
```

Figure (4.6): Defining and perturbing the Supply and Demand curve equations ([Kuipers, 5])

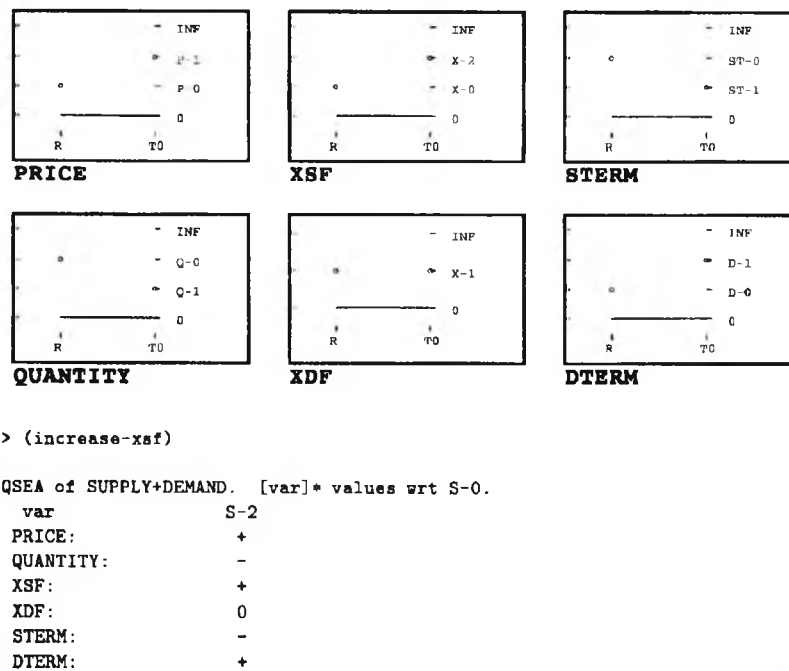


Figure (4.7): The reference state (*R*) and the perturbed solution (*TO*) for the Supply and Demand curves, responding to increased *XSF* ([Kuipers, 5]).

4.4.9 Dealing with complex systems

Many complex system models have too many state variables, and hence too many possible successor states, to be tractable by qualitative simulation. The set of global limit hypotheses grows exponentially with the number of variables. However, we are frequently faced with large systems consisting with variables that do interact, but appear intractable to current qualitative reasoning methods.

Therefore, there is a need to define a valid hierarchical structure that breaks a complex system into a collection of tractable mechanisms. One important method for distinguishing closely related mechanisms within the same large system is the *time-scale abstraction* applied to a complex system made up of interacting equilibrium mechanisms:

Time-Scale Abstraction Principle: If a complex system can be decomposed into equilibrium mechanisms that operate at widely separated time-scales, then a particular mechanism can view a faster one as being instantaneous, and a slower one as being constant.

A discipline for shifting the focus of attention among different time-scales has been developed by [Kuipers, 4]. The two directions of shift in focus from a given mechanism require different methods:

- **Faster to Slower**: Given an initial perturbation to its environment, qualitative simulation predicts the resulting equilibrium state of the fastest mechanism, and shifts attention to the next slower one. The final values of variables that are shared with the slower mechanism can be treated as part of the initial state of the slower mechanism.
- **Slower to Faster**: After a slower mechanism has reached equilibrium, the environment it provides for a faster mechanism may have changed. However, the faster mechanism, by definition, must have tracked the slower mechanism on its way to equilibrium. Thus, the fast mechanism is already in equilibrium, and its state can be derived from the values of shared variables by comparative statics [Kuipers, 5].

Figure 4.8 shows the pattern of control for a three-level time-scale hierarchy. Upward arrows initiate dynamic simulation to a new equilibrium, and downward arrows initiate propagation to a more complete description of an existing equilibrium state. Each bead represents a qualitative state, so simulation produces a string of beads, and propagation of an equilibrium state produces a single bead. Changes in focus of attention take place in the sequence shown. (1) The equilibrium state of the fastest mechanism provides values for initialising a simulation of the next slower mechanism. (2) The final state of the second simulation is first used to propagate a new equilibrium state for the fastest mechanism. (3) Then values from both faster mechanisms are available to initialise the slowest mechanism. And so on.

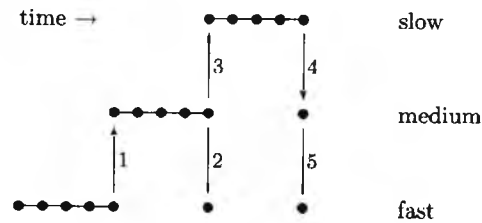


Figure (4.8): Control of focus of attention ([Kuipers, 5])

Formally, the algorithm is as follows. After simulating a mechanism, QSIM identifies the faster mechanisms that share variables with the current mechanism, and propagates the shared values to determine the equilibrium state of the faster mechanism. Once this is done, the slower mechanisms sharing variables are identified. The current values of variables shared with this mechanism are used to define the initial state for which it is simulated. The process repeats recursively [Kuipers, 5].

Some other related work includes the work of [Chin *et al.*, 1] and [Kreiss, 1], who have discussed a variety of analytic and numerical methods applying to ordinary and partial differential equations to determine and exploit knowledge of relative time-scales.

4.4.10 Comments on Qualitative Reasoning

The purpose of qualitative simulation is to reason from physical descriptions to behavioural descriptions and to explain process observations directly from a process model. Its power lies on its ability to yield partial conclusions from incomplete knowledge of the process. A representation language, which is a qualitative abstraction of differential equations, makes it possible to handle incompletely-known constraints or parameter values that would prevent a quantitative differential equation model from being formulated.

[Forbus, 1]'s fairly weak formalisation of quantities (partially ordered amounts, IQ values, and rates) and the basic concepts of Qualitative process have also been presented. The conceptual framework, on which Forbus and Kuipers have based their research work, could possibly as well be applied in deriving a methodology for modelling business processes. This issue could be further elaborated in a future work.

Qualitative modelling and simulation are key inference methods for problem-solvers for the major tasks of model-based reasoning: monitoring, diagnosis, design, planning, and explanation. So far, the methodology has been applied to the representation and reasoning with incomplete knowledge about physical mechanisms: bathtubs, automobiles, refrigerators, electrical circuits, chemical processing plants, the physiology of the body, botany, and so on.

Qualitative simulation shows promise, but limitations exist due to the lack of functional and quantitative information in complex systems. Techniques have been developed and continue to be developed to cope with some of these issues, such as methods for decomposing large problems based on a hierarchical structure of time-scales, for predicting better the behaviour of a system (see Q3 semi-quantitative approach to simulation), and for increasing the precision of available results but still retaining the inherent advantages of qualitative modelling.

However, applications of expert systems are usually in domains where complete, well-defined mathematical models do not exist. The same applies to soft systems. In these cases, it is still possible to reason qualitatively about a system to make useful conclusions, by capturing the underlying principles of a mechanism. In addition, by enabling reasoning from partial knowledge, qualitative reasoning offers a method for relieving the problems encompassed in the modelling of an expert system.

4.4.11 Order-of-magnitude reasoning

Qualitative reasoning [Bobrow, 1] is a well-defined artificial intelligence method for dealing with qualitative models; it uses only the signs of parameters and constrains them through qualitative constraints. Major difficulties encountered in the reasoning effort, particularly in engineering applications, stem from the ambiguity inherent in the qualitative values (-, 0, +) normally used [deKleer & Brown, 1]. When [Forbus, 2] included the inequality relations through the quantity space only partially resolved the ambiguities. Qualitative reasoning focuses the attention on the values of individual parameters, but not on the *relations* between the parameters.

Qualitative reasoning intentionally neglects certain knowledge about parameters because it targets reasoning at a very abstract level. Abstracting algebraic constraints all the way to qualitative constraints among signs also degrades the

knowledge that the constraints contained. Also, it excludes number manipulation, and any information more accurate than signs remains thus unused [Mavro. & Steph.,2].

Therefore, it is desirable to introduce an abstraction level lower than qualitative values, to achieve more expressive power and effective reasoning. **Order-of-magnitude** reasoning offers exactly the necessary intermediate abstraction level.

It is based on the representation of the relative orders of magnitude of the parameters of a system. It captures the semi-quantitative concepts that are implied by statements of the kind "A is much smaller than B".

In 1986 [Raiman, 1] [Dague *et al.*, 1] introduced the FOG (Formal Order of magnitude) system with three basic relations:

- A Ne B: A is negligible in relation to B
- A Vo B: A is close to B (and has the same sign as B)
- A Co B: A has the same sign and order of magnitude as B

The FOG system had 30 rules of reasoning involving its basic relations, classical qualitative values, addition and multiplication. [Raiman, 1] said that, intuitively, this way of reasoning was like using a coarse balance, which weighs quantities with a variable level of precision.

Unfortunately, FOG has several disadvantages and in particular:

1. It does not provide concrete interpretation of its relations
2. Its set of rules appears arbitrary, and it is not clear how it can be extended
3. It does not allow incorporation of partial quantitative information
4. It concentrates signs and magnitudes in single relations
5. It requires the explicit use of negation and disjunction, along with inequality relations, to express even very simple and frequently used relations, such as $>$ and $<$.

4.4.12 Basic Concepts

A *variable* in O[M] refers to a specific physical quantity, with known physical dimensions but unknown numerical value. A *landmark* is similar to a variable, but it has known sign and value. Variables and landmarks are called *quantities*. Within each quantity, there are links, each representing a compatible pair of quantities that can be interrelated. A *link* contains all the order-of-magnitude relations asserted between the

two quantities, and information on where such relations can be obtained from and where they can be used.

Order-of-magnitude relations relate the absolute magnitudes of quantities, without reference to their sign. Thus, there is no interference between signs and magnitudes, so that reasoning with signs can be performed under the usual qualitative reasoning principles [Mavro & Steph., 1]. There are seven *primitive* irreducible binary relations among quantities, see (table 4.1).

| O(M) relation | Verbal explanation |
|--------------------|--------------------------------|
| $r_1 : A \ll B$ | A is much smaller than B |
| $r_2 : A- < B$ | A is moderately smaller than B |
| $r_3 : A \sim < B$ | A is slightly smaller than B |
| $r_4 : A = B$ | A is exactly equal to B |
| $r_5 : A > \sim B$ | A is slightly larger than B |
| $r_6 : A > -B$ | A is moderately larger than B |
| $r_7 : A \gg B$ | A is much larger than B |

Table (4.1): Primitive relations of the O(M) formalism ([Mavro. & Steph., 2])

We accept as a *compound* relation any implicit disjunction of two or more successive primitive relations. For example, the compound relation standing for “A less than B” would be represented as $A \ll \dots \sim < B$, [Mavro. & Steph., 2]. There are in total 21 compound relations. Therefore, we get 28 *legitimate* relations and this set allows full expressiveness without disjunction or negation. All these relations are powerful enough to express quantity-space partial ordering and all of FOG’s relations (see table 4.2). For more information on the semantics of O[M] relations, please refer to Appendix.

| Classical commonsense relations | O(M) |
|--------------------------------------|-----------------------|
| Less than (<) | $\ll \dots \sim <$ |
| Less than or equal to (\leq) | $\ll \dots =$ |
| Greater than (>) | $> \sim \dots \gg$ |
| Greater than or equal to (\geq) | $= \dots \gg$ |
| Equal to (=) | $=$ |
| Approximately equal to (\approx) | $\sim < \dots > \sim$ |

| | |
|---|----------------|
| Less than or approximately equal to ($< \sim$) | $<< .. > \sim$ |
| Greater than or approximately equal to ($> \sim$) | $\sim < .. >>$ |
| Much less than | $<<$ |
| Much greater than | $>>$ |

Table (4.2): O(M) relations representing relations that are commonly used in engineering ([Mavro. & Steph., 2])

4.4.13 Assignments, Constraints, and Rules

Assignments are “solved” algebraic relations that allow some quantities to produce relations among other quantities. For example:

$$A = B + 3.5C \exp(D/ E) - 2F \tag{1}$$

$$(A/ B) = 1 + 3.5(C/ B) \exp(D/ E) - 2(F/ B) \tag{2}$$

The right-hand-side of an assignment called *expression*, cannot be any arbitrary algebraic expression [Mavro. & Steph, 1]. It can involve only links, landmarks, and numerical constants. The left-hand-side, called *destination* can be a link or a variable. In equation (1) the destination is a variable and the assignment produces a range for that variable (i.e. A), while in equation (2), the destination is a link and the assignment produces relations for that link (i.e. relations between A and B).

The system attempts to convert algebraic expressions (i.e. equation (1)) to the acceptable form, by mapping ratios of compatible quantities to links (i.e. equation (2)).

Constraints are “unsolved” algebraic relations among quantities, such as:

$$A + 2F - B = 3.5C \exp(D/ E) \tag{3}$$

There are two ways in which the O(M) can use constraints in the reasoning [Mavro. & Steph., 1]:

- The first way is to simply substitute values for variables and links, and test the constraint. Then to accept or reject assumptions based on the outcome.
- The second way is to form a set of assignments by solving the constraint in all obvious ways. In this fashion, constraint (3) would yield the assignments (1) and (2).

Finally, O(M) allows unstructured knowledge (such as the knowledge of highly-empirical nature) to be expressed in simple *if-then* statements, by forming *rules*. For example:

$$\text{If } (A \ll B \text{ and } D \ll B) \text{ then } (A \sim < D)$$

4.4.14 O(M) Reasoning

The basic strategy of O(M) is depth-first data-driven reasoning. In effect, any potential new fact is first checked for redundancy, and then created and used in more inferences, regardless of whether the use of its “parent” has been completed.

O(M) uses for further reasoning the following :

- From the conjunction of relations within the same link, new relations are inferred
- From the *symmetry* and *transitivity* properties of O(M) relations, new relations are inferred.
- For relations between a variable and a landmark, *numeric transitivity* is applied.
- When a relation (actually its link) can serve as the antecedent of rules, *applicable rules* are invoked. Applying an assignment can yield knowledge about the magnitude as well as the sign of a variable.

The search mode for O(M) is opportunistic forward chaining, but there are two ways to induce search for a particular relation. One way is by stating that the goal is to relate two particular quantities. Then the user can induce additional ways to use constraints and assignments. Whenever one of the two goal quantities occurs, the system uses the other one as well. For example, if the link between the quantities A and G is a target and the constraint (3) is given, O(M) will derive the constraint:

$$(A/G) + 2(F/G) - (B/G) = 3.5(C/G) \exp(D/E) \quad (4)$$

in which the target link A/G occurs. The constraint in turn yields the assignment:

$$(A/G) = (B/G) + 3.5(C/G) \exp(D/E) - 2(F/G) \quad (5)$$

which is capable of producing relations between A and G .

Alternatively, the user may state that alternative relations between two quantities should be examined. Then, the system can create seven assumptions, one for each of the seven primitive relations and check them for consistency with available

knowledge. For example, with A and G as the target, the assumptions $A \ll G$, $A- < G$, $A \sim < G$, $A == G$, $A > \sim G$, $A > -G$, and $A \gg G$ are created.

4.4.15 Related work

Building approximate models is not just useful: they are essential for a clear understanding of the relevant issues in a difficult problem. Model simplification – the derivation of simpler equations from more general ones – is a recurrent problem in many areas of science. The real problem is to reduce these equations, so complex and general, with so many variables linked together to a form responsive to analysis and interpretation [Yip, 1]. Relevant work in equation simplification involves the various styles of order-of-magnitude reasoning.

Raiman's *estimates* [Raiman, 2] is based on the notion of order of magnitude scales. The intuition is that an order of magnitude description is a coarse description of a quantity. The coarse value $V(q)$ of a quantity q is defined as a set of values that contains the value of q . Two quantities p and q are said to be equal in order of magnitude if the sets $V(p)$ and $V(q)$ overlap. Two primitive scales, *small* and *rough*, are provided; they can be used to build up finer order of magnitude scales. Order of magnitude equality, under Raiman's definition, is *not transitive*. As a result, one cannot substitute equals for equals, which complicates the inference machinery.

[Murthy, 1] introduces a four-level quantity space, incorporating some of Raiman's infinitesimal relations, and algorithms. [Bennett, 1] and [Nayak, 1] explore the use of logarithmic scales in approximation. They both define the order of magnitude of a quantity $O(q)$ to be $\log_b |q|$, where the choice of the base b is problem-dependent.

Last but not least, [Yip, 1] has produced an implemented program, called AOM, which is based on two domain-independent ideas: self-consistent approximations, and asymptotic order of magnitude reasoning. The spirit of the analysis is heuristic and exploratory. His approach differs from other works in two major aspects. First, whereas all the previous works deal with either qualitative models or models specified by algebraic or differential equations (ODEs), AOM is one of the first programs to handle systems of non-linear partial differential equations (PDEs). Second,

the program is based on a theory of asymptotic order of magnitude of *functions*, which applies to algebraic equations and ODEs as well as PDEs.

4.4.16 Comments on Order-of-Magnitude Reasoning

O(M) is capable of handling diverse forms of knowledge, including constraints, order-of-magnitude relations, rules, assumptions and even exact numerical knowledge. With a formalisation of order-of-magnitude reasoning many advantages are realised [Mavro. & Steph., 2]:

- Previously informal notions and methods can be described and communicated concretely.
- Order-of-magnitude can be performed by computers
- Available quantitative knowledge can be used along with semi-quantitative knowledge.
- When knowledge-based systems are employed, order-of-magnitude reasoning is indispensable, because it can acquire and use efficiently the semi-quantitative knowledge that experts rely on.

O(M) has been so far applied to different process engineering activities, such as the design of process flowsheets [Douglas, 2] and [Douglas et al., 1], the design of control structures for chemical processes [Douglas, 1], fault diagnosis [Oyel. & Kramer, 1], the explanation of process behaviour [Dalle Molle et al., 1], and the analysis of biochemical systems [Mavro. et al., 1].

The models required for dealing with the analysis of soft systems are necessarily dynamic ones, and quantitative dynamic models are usually hard to obtain and cumbersome to use. Order-of-magnitude reasoning could be applied to the modelling of systems, such as the ones appearing in business systems. O(M) has been used to applications that centre around activities characterised by the use of rough models. It provides us with the ability to discriminate among alternatives or discern dominant factors.

4.5 Part II: Functional Reasoning and Multilevel Flow Modelling

Representing knowledge of how things work has long been and continues to be an important problem in different fields of research. Although, many efforts have focused on determining an appropriate representational model, there has been no real agreement on what essential information these models should include.

Here, the focus is on the description of progress towards a device representation that organises knowledge based on functionality [Keuneke, 1].

4.5.1 The significance of Functional Knowledge

As we have already seen, the connection to different scientific traditions has caused disagreements about the definition of the concept of function. We agree with [Keuneke, 1], who states that the *function* of a device is its intended purpose. The functional specification describes the device's goals at a level of abstraction that is of interest at the device model.

Functions are achieved by *behaviours*. According to [Keuneke, 1], models represent *behaviour* as the causal sequence of transitions of partial states. Finally, *structure* indicates the arrangement and relationship of the components of a device model. *Functional structuring* is useful because in order to understand the functioning of a device, there is often need to decompose the device's function into the component's functions.

4.5.2 Linking Qualitative Reasoning with Functional Reasoning

Researchers within qualitative reasoning have developed methods to derive behaviour from structure ([deKleer & Brown, 1], [Forbus, 1] and [Kuipers, 1]). The relation between function and behaviour has been recognised in this research but is not dealt with explicitly.

In qualitative simulation, device knowledge is acquired through a progression of tasks, beginning with a structural description and then progressing through behavioural representation to functional representation. The first stage determines what

kinds of behaviour can arise, given the structure. [deKleer & Brown, 1] and [deKleer & Brown, 2] try to make implicit assumptions explicit. A functional representation agrees with their view and in addition, rather than focusing on problem solving to determine implicit assumptions and progress from one form to another, it focuses on explicitly determining what the final representation should be.

[Keuneke, 1] has recognised the need of making these assumptions explicit by developing concepts for the representation of the intended behaviour, that is, the functions. This provides a constructive link between structure, behaviour, and function. These links are further strengthened by the hybrid phenomena theory (HPT) of [Woods, 1]. He manages to generate ordinary differential equations from qualitative knowledge.

4.5.3 The Functional Representation

[Keuneke, 1] proposes a representation based on the work of [Semb.& Chand., 1], where *structure* specifies a device's components and the relations between them, *function* specifies the result or goal of a device's or component's activity, and finally, *behaviour* specifies, how given a stimulus, the function is accomplished.

In a functional representation, a device's structure is built using functional components. Thus, every component is associated with functions. To understand a function, we have to know the goal, how the goal can be achieved and also when its achievement is desired.

We can enhance a functional representation with additional distinctions about functions, i.e. by specifying types of functions [Keuneke, 2]. Each of the four function types, mentioned below, indicates different simulation procedures, functional capabilities, and expectations. The function types are:

- **ToMake:** it achieves a specific partial state. It reflects the idea that a component's function is either to achieve a specific state of the device (for example ToMake:(door locked)) or to achieve a value or state of some substance parameter that the device manipulates (for example ToMake:(substance liquid)).
- **ToMaintain:** it achieves and sustains a desired state over a period of time. In general, maintenance implies three important features: (1) continuous monitoring, (2) a range within which a variable is expected to be maintained, and (3) the

potential for needed adjustments to keep the variable value within that range. Maintenance behaviours are not simple, linear, causal sequences that reach an end state; rather they involve continuous, cyclic action to keep the system in some desired state.

- **ToPrevent:** it keeps a system out of an undesirable state. The primary differences between ToMaintain and ToPrevent functions are intent and continuity of use in a given system. ToPrevent functions provide short-term, fail-safe mechanisms, not operations for normal and continuous maintenance.
- **ToControl:** it gives a system the power to regulate state changes via a known relationship. To control something or someone implies a direct multivalued relationship between the device's action and the resulting effects. Control functions possess specific knowledge of how to provide multiple outputs and manipulate components to provide the exact output desired.

This representation seems especially appropriate in diagnosis, since diagnosis involves determining which change in structure resulted in some malfunction. One can find further details on how functional knowledge can be used in [Chand. *et al.*, 1] and [Chand. *et al.*, 2].

4.5.4 Multilevel Flow Modelling

Multilevel Flow Modelling (MFM) differs from other approaches to modelling, especially modelling techniques based on laws of physics. MFM is based on concepts of goals and functions, which traditionally are not used within the natural sciences but used within the human sciences.

Another important aspect of MFM is the representation of a system by a multiple of descriptions on different levels of abstraction. Functional descriptions are context dependent because they relate to system goals. This context sensitivity is specifically addressed in MFM [Lind, 2].

A system is described in terms of goals, functions and the physical components (along the means-end axis). At the same time, each of these descriptions can be given on different levels of decomposition along the whole-part axis (see Figure 4.9).

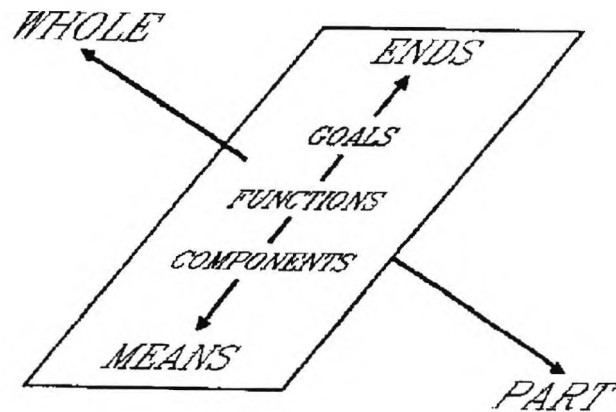


Figure (4.9): Dimensions of representation in MFM ([Lind, 2])

MFM provides a systematic approach in coping with large-scale decision problems, where the models tend to be very complex and unmanageable because the level of detail introduces so many unknown factors. In such cases more qualitative models of relations between goals and resources such as MFM seem to be more adequate.

[Duncker, 1] describes experiments in which human problem solvers have difficulties in solving problems that require change of system representation. The problem is the difficulty of changing the perception of the physical world in response to changing problem solving goals. And problems like that are especially acute in man-made systems. We can find application of MFM in the design of human-machine interfaces in [Agger, 1], [Jensen, 1], [Osman, 1], [Lind, 1] and [Lind, 3].

4.5.5 Concepts of Goals and Functions

The concepts of goal and function are fundamental for the representation of knowledge of man-made systems. These concepts have also been playing an important role within biology, and especially in understanding the living organisms, [Hempel, 1], [Nagel, 1] and [Woodger, 1].

The following definitions on the concepts of goals and functions seem to be commonly agreed:

Goal ascription: States the outcome or (goal) toward which certain activities of an organism / system or of its parts are directed.

Function ascription: States what role an organism / system has in the achievement of one or several goals on the basis of its structure and behaviour. The definition of functions as *roles* is also used within the social sciences.

We should note that these definitions emphasise that goals and functions are not inherent properties of systems such as physical properties of mass or temperature. They are properties ascribed to systems and cannot be identified without knowledge of the environment or content of which the system is a part. A very good example that clarifies the concepts of goals and functions is given below (Figure 4.10).

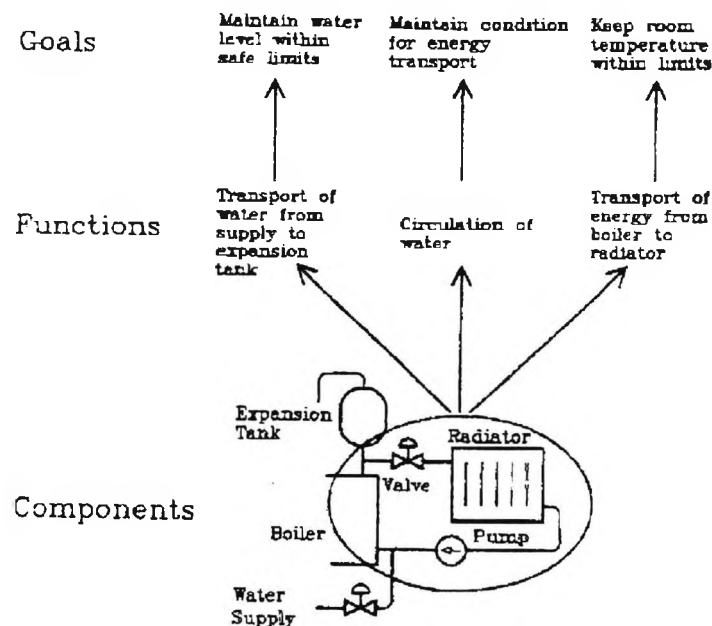


Figure (4.10): Goals and functions of a circulation system ([Lind, 2])

4.5.6 Modelling using the MFM methodology

The aim of MFM is to offer a systematic basis for using means-end and whole-part decompositions in the modelling of complex industrial plant [Lind, 5]. The means-ends distinction is used to represent a plant by the goals of its designer or user, by its functions and its physical components. The descriptions of goals, functions, and components can, at the same time, be given on different levels of part-whole decomposition.

When building MFM models, the initial step is to identify the goals for the system because functional descriptions have no meaning without a previously defined

goal context [Lind, 6]. [Lind, 2] gives examples of descriptions along the two dimensions of the MFM representations by using the water circulation circuit of a central heating system (see the above mentioned Figure 4.10).

It can be seen that the physical parts (components) of this circuit realise different functions and that the functions contribute to the achievement of system goals. The functions are ascribed to the system with a specific goal in view. Thus, if the goal of “maintain water level within safe limits” is selected, then the relevant function of the circulation circuit is to transport the water from the water supply line to the expansion tank. As we can see, the functions here describe how the behaviours of the components are useful for the achievement of various purposes or goals. Therefore, we cannot meaningfully separate the ascription of the functions from the selection goals. The functional ascriptions are dependent on a predefined context of one of several goals [Lind, 5].

The principles of the whole-part decomposition of goals, functions and components are also illustrated in [Lind, 6]. For example, the three goals, depicted in the example, could aggregate into a superordinate goal specifying “the central heating system should operate properly”. Furthermore, as another example, we could decompose the function of transporting energy from the boiler to the radiator into a chain of two transport functions, each describing the functions of the heat transfer between the burning air-gas mixture and the piping and between the piping and the water in the boiler. Finally, another way of decomposition would be to decompose the circulation circuit into the pump, the boiler, the valve, the radiator, and the piping.

In other words, whole-part relations are used to describe the relation between an entity of some category with sub-entities of the same category. They are partial order relations and create hierarchical structures. Means-end relations are used to represent the relation between the goals and the associated functions provided for their achievement. They are divided into two main subcategories of relations, the *achievement* and the *condition* relations (see [Lind, 2]).

Figure 4.11 shows the relations between goals, functions and equipment in the central heating system [Lind, 2].

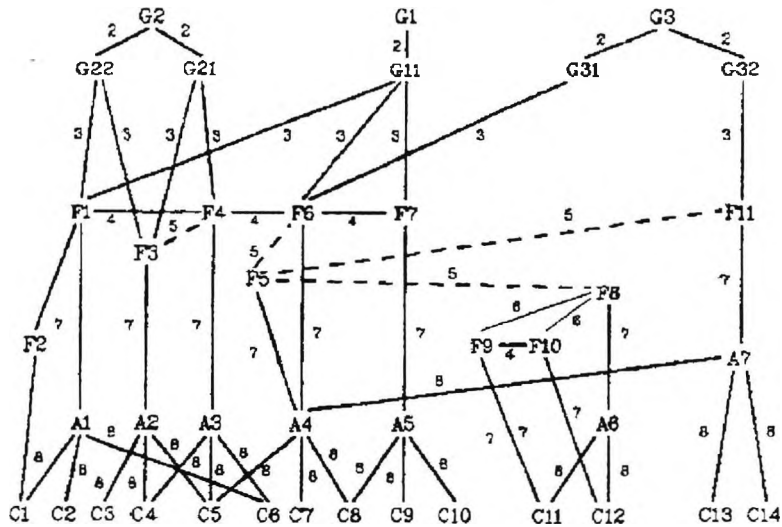


Figure (4.11): Relations between goals, functions and equipment in the central heating system ([Lind, 2])

where:

Goals:

- G11: Maintain room temperature within defined limits
- G21: Minimise heat losses
- G22: Optimise fuel combustion
- G31: Keep water temperature below boiling point
- G32: Keep water inventory below upper limits

Functions:

F1: Energy supply, F2: Storage of fuel, F3: Air-gas path in the boiler, F4: Distribution of heat in the boiler, F5: Circulation of water, F6: Transfer of heat from boiler to radiator, F7: Control of room temperature, F8: Lubrication of the circulation pump, F9: Storage of lubrication oil, F10: Circulation of lubrication oil, and F11: Water supply.

Components:

C1: Fuel oil tank, C2: Oil pump, C3: Air blower, C4: Chimney, C5: Boiler, C6: Oil burner, C7: Circulation pump, C8: Radiator, C9: Temperature regulator, C10: Room, C11: Lubrication oil tank, C12: Lubrication oil pump, C13: Expansion tank, C14: Water valve.

Also in order to make the diagram we have created seven physical aggregates, which each represents a group of components:

- A1: C1 and C2
- A2: C3, C4 and C5
- A3: C4, C5 and C6
- A4: C5, C7 and C8
- A5: C8, C9 and C10
- A6: C11 and C12
- A7: A4, C13 and C14

The relations, which in the diagram are represented as arcs between the different objects, are of different types [Lind, 2]. The functions are the result of the behavioural interactions between the components and relate to a specific goal to be achieved [Lind, 6]. The semantic and pragmatic aspects of MFM modelling are discussed in the work of [Lind, 4], [Larsen, 1] and [Jørgensen, 1].

As [Lind, 6] mentions in his work, by studying this example, one can make some general observations of the nature of MFM models. First, it is seen that means-end and whole-part decompositions lead to a multiple of interdependent representations of the same system linked to many-to-many relations of realisation and achievement. Furthermore, each plant goal or subgoal will in MFM be a starting point for application of the two types of decomposition. This is because functional ascription is made within a context of goals.

4.5.7 Flow Structures

The basic concepts of Multilevel Flow Modelling are given in the Appendix. There one finds the different types of flow functions, which individually are never alone when building flow models. Each flow function requires a context to be interconnected with something else in order to make sense. A flow function cannot exist in its own right [Lind, 2]. *Flow structures* define contexts within which the individual flow functions get their meaning. A *flow structure* is accordingly a graph consisting of nodes which are basic flow functions. Flow structures must satisfy a set of syntactical rules. Please refer to the Appendix for a detailed list of them. Note that even though the rules are followed, it is not a guarantee that the model will be correct in a semantic sense.

Using the example with the circulation circuit of the central heating system, [Lind, 2] presents some flow structures. For further analysis of the specific example, please refer to [Lind, 2], (see Figure 4.12).

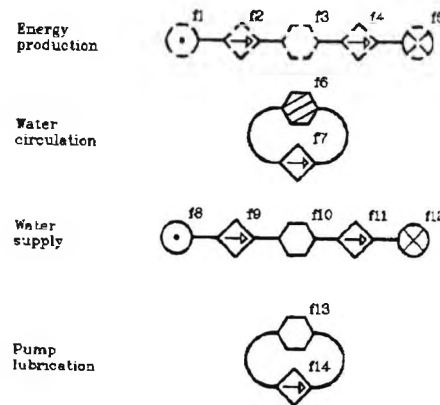


Figure (4.12): Flow structures for the central heating system ([Lind, 2])

[Lind, 5] carries on by introducing the concept of *elementary changes* being influenced by [VonWright, 1]. According to his approach, there are four types of change and not-change that are possible regarding a given state of affairs. He also refers to the concept of *elementary acts*, which are understood to be an act in which the result is an elementary change. The conditions for each type of act are discussed in detail by [VonWright, 1] and are summarised in table 4.3.

| Condition of action | Explanation | Action | Explanation | Result of action | Explanation |
|---------------------|--|--------------------|-------------------|------------------|--------------------|
| $PT\sim p$ | p exists but vanishes unless maintained | $d(pTp)$ | p is maintained | pTp | p remains |
| $\sim pT\sim p$ | p does not exist and does not happen unless produced | $d(\sim pTp)$ | p is produced | $\sim pTp$ | p happens |
| PTp | p exists and remains unless destroyed | $d(pT\sim p)$ | p is destroyed | $pT\sim p$ | p vanishes |
| $\sim pTp$ | p does not exist but happens unless suppressed | $d(\sim pT\sim p)$ | p is suppressed | $\sim pT\sim p$ | p remains absent |

Table (4.3): Elementary Acts ([VonWright, 1])

Finally, the resulting MFM model of the central heating system is depicted below (see Figure 4.13).

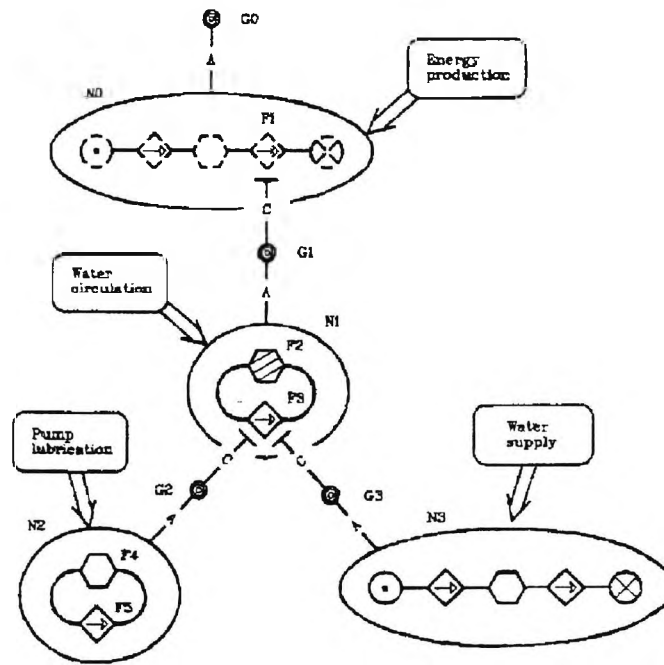


Figure (4.13): MFM model of the central heating system ([Lind, 2])

4.5.8 Tools

Lind has developed an object-oriented modelling and reasoning tool, called Abstractions, in order to implement MFM models and to build diagnostic and planning applications based on MFM [Lind, 3]. Abstractions is made up of a set of classes representing basic modelling concepts such as units, structures, and aggregates, which support the building of models (like MFM) that can be defined by graphs. The tool has been provided with a graphic interface for direct manipulation of model objects. This interface, called Grace [Osman, 2], enables the user to build models by mouse-based operations on icons and provides advanced facilities for navigating through the different levels of a model.

MFM has been applied in two major Commission of the European Community (CEC) projects within the ESPRIT I and II framework programs. In the ESB project (Expert System Builder, ESPRIT I), a model of a Danish power plant was built, and a prototype MFM-based diagnosis system was developed using the ESB tools. In KBSSHIP (Knowledge Based Systems Onboard Ships, ESPRIT II), the focus was on the development of generic MFM models, i.e. models that can be used across several

applications. A C++ implementation of a subset of Abstractions has also been developed [Jørgensen, 1].

[Larsson, 1] has developed in his PhD project an MFM model of a milk sterilisation plant. The model has been developed using an MFM toolbox for the real-time expert system shell G2.

Finally, [Sassen *et al.*, 1] have developed a diagnostic system for ship machinery, using the real-time expert system PERFECT (Programming EnviRonment For Expertsystems Constrained in reasoning Time). To build a monitor- and diagnosis-system of a complex process plant with PERFECT, one must represent a multilevel flow model of that plant in the PERFECT modelling technique.

4.5.9 Comments on Multilevel Flow Modelling

Model building is known to be a complex task that requires a lot of experience. The main problem is how to select the proper level of detail so that the model will be useful for the selected decision task. In the case of MFM models, it is possible to formulate a strategy for building a model. The strategy is based on the basic observation that it is impossible to ascribe functions to a system without having defined a goal for the system.

An MFM model is a complex of interconnected means-end structures. The model is multilevel because it comprises several levels of abstraction and it is about flow because every functional abstraction is described in terms of flow structures. The MFM model gives one some insight in the system as it represents the interdependency between the resources in the system provided for goal achievement. The model puts focus on the organisational structure of the system.

The development of an MFM model is a highly iterative process. The strength of the MFM representation is that it provides a functional plant decomposition that is crucial to solving complex planning and diagnosis problems. An important area of application is in the planning of human-machine interfaces and in the manipulation of large-scale decision problems, such as the ones encountered in chemical engineering and biology. MFM methodology is adequate for coping with design problems that include strategic or tactical decisions about the management of system resources.

4.6 Part III: Goal-driven Methodologies

People are beginning to realise that new technology should not only be used to automate existing business processes, but should also be used as a basis for restructuring these processes to meet new business realities [Dav. & Short, 1], [Hammer, 1] and [Venkat., 1]. One step to a more systematic approach of designing business processes is to develop models that provide appropriate representations of the knowledge that is needed for understanding and reasoning about business processes.

Business modelling and goal modelling can be regarded as top-down methods, which successively decompose and satisfy single modelling states. Here, we will introduce an approach that uses goals as means to drive and manage the business modelling process.

4.6.1 Modelling the intentional structure of a business process

So far, workflow models, which show the flow of work products from one work unit to another, have been commonly used to describe business processes. Their main advantages are that they are intuitive and easy to understand, but workflow models lack to provide the *reasons* that underlie work activities and products. They do not capture the “whys” – the intentions, motivations, and rationales that underlie the “whats” [Hammer, 1]. In order to have a deeper understanding of how a business process operates, the *intentional* dimension of organisational work needs to be captured in our model.

Business processes can be presented by way of a network of interdependencies among actors: actors depend on each other for *goals* to be achieved, *tasks* to be performed, and *resources* to be furnished.

In their work [Yu & Mylop., 1], introduce a modelling framework that consists of two main components. The *Actor Dependency* model describes an organisation as a network of interdependencies among actors and the *Issue Argumentation* model captures the arguments about the relative merits of alternative designs with respect to various issues of concern. *Goals*, *rules*, and *methods* provide representations for generic means- end relationships in the framework. More analytically, we have:

An *Actor Dependency* (AD) model is a graph, where each node represents an “actor”, and each link between two actors indicates that one actor depends on the other for some object in order that the former may achieve some goal. The depending actor is called the *dependor*, and the actor who is depended upon the *dependee*. The object around which the dependency relationship centres is called the *dependum*.

[Yu & Mylop., 1] recognise four types of dependencies. In a *goal-dependency*, the dependor depends on the dependee to bring about a certain state in the world. In a *task-dependency*, the dependor depends on the dependee to carry out an activity. This kind of dependency specifies how the task is going to be performed. In a *resource-dependency*, the dependor depends on the dependee for the availability of an entity. A resource is usually the finished product of some deliberation-action process. Finally, a *soft-goal-dependency* is similar to goal dependency, apart from the fact that the condition to be achieved is not sharply defined, but requires clarification between dependor and dependee.

The model also allows different degrees of strength in the dependencies. There are three levels of dependencies: Open, Committed and Critical [Yu, 1] and [Yu & Mylop., 2].

After an Actor Dependency model has explicitly represented the goals, we realise that there are more than one way of to do things. If the goals are explicit, then a rule can easily be seen as *one* way of achieving a goal. In this modelling framework, a *rule* is expressed in terms of attributes in an activity description. We add a *goal* attribute to activity and in this way we allow the activity to be selected as a candidate for anyone who wants to have this goal met. Also, instead of decomposing only into subactivities, we allow *subgoals* as well as *subtasks* (following [Croft & Lefk., 1]. A subgoal will lead to a search for an activity that matches, whereas a subtask names a particular activity without involving search, Figure 4.14.

When organising a business process the objective is to find a workable division of labour among organisational members, i.e. a network of intentional dependencies. The kinds of rules that are most relevant for organisation design and business process re-engineering are therefore those that lead to dependency relationships among actors [Yu & Mylop., 1]. For example in Figure 4.14, the third option for achieving the goal “ordered (item)” is by way of a goal dependency.

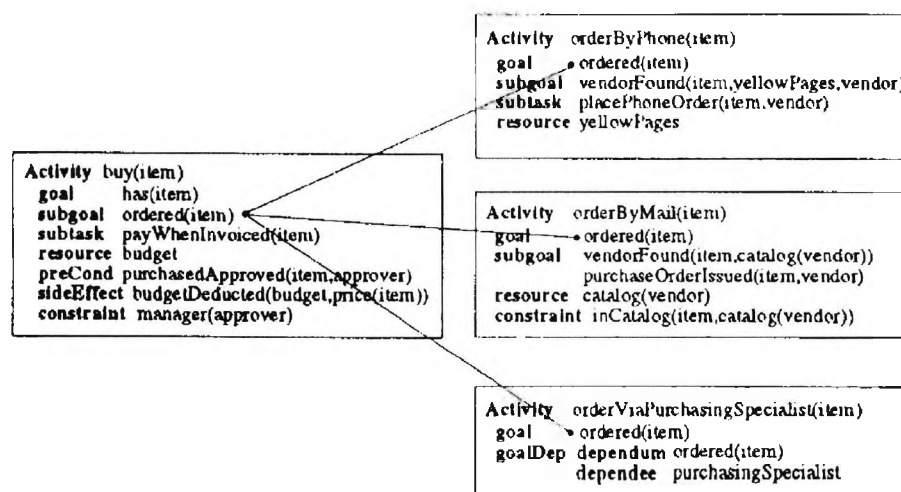


Figure (4.14): Rules expressed as activity descriptions ([Yu & Mylop., 1])

4.6.2 Redesigning the process

The Actor Dependency Model is concerned with the structural elements that enable a goal to be achieved. Nevertheless, there is no representation of how well the goal might be achieved. We adopt an *Issue Argumentation* (IA) model to support reasoning about process redesign. This is a network of assertions (issues) linked by reasons (arguments). The IA model is a reasoning structure, which shows the relationships among a set of issues. [Lee, 1] interprets issues as *goals*. Issues are pursued until acceptable solutions are found.

Issues are treated as *contributing factors* toward other issues. Contributions can be either *positive* or *negative*. [Yu & Mylop., 1] also use the term *satisfied* to indicate that a contribution is strong enough to address an issue. To indicate the different types of relationships between issues (nodes), the IA model provides the following link types, according to the type of information they carry:

- the direction of the contribution – from which node to which node. We use arrowheads that point from the solution towards the goal.
- the sense of the contribution – whether the contribution is a positive (+) or a negative (-) one.

- the extent or degree of the contribution. Following [Chung, 1], we distinguish between enough contribution versus not enough (partial) contribution. These are called *sup* (Δ) and *sub* (∇) connoting above and below.
- how multiple links contribute towards a node – either as a single line arc across the links, indicating the combined effect *AND*, or as a double line arc, indicating the separate effect *OR*.

A *+sub* link represents a positive contribution that is partial in extent, i.e. not enough to say that the goal has been met. A *+sup* link indicates a positive contribution that is enough to address the goal. The *AND* link indicates that several contributions together make enough of a contribution to address a goal. The *OR* link indicates that each of the individual contributions among several is enough to address the goal. Last but not least, the *und* link type is used to indicate that one issue affects another in some way, but the sense of the contribution is not known. For example, Figure 4.15 shows the goods acquisition process.

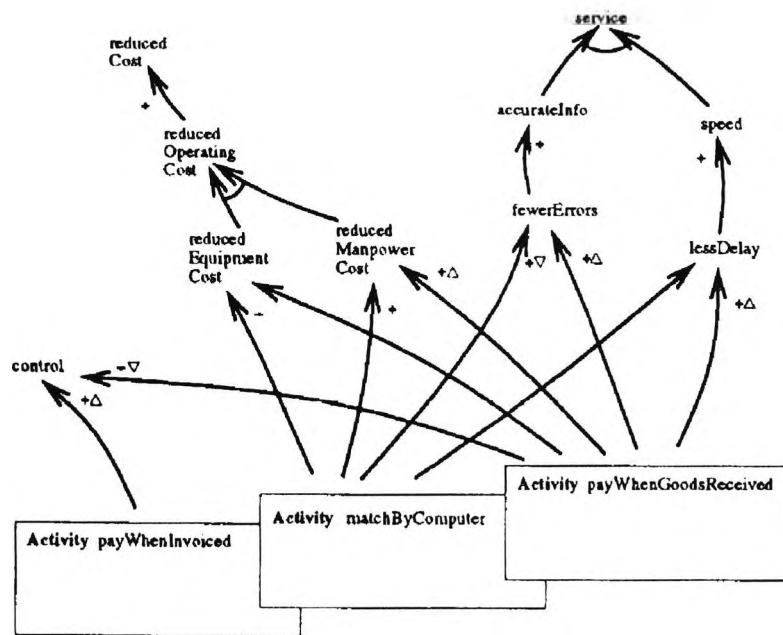


Figure (4.15): Issue Argumentation model supporting redesign of a goods acquisition process.

([Yu & Mylop., 1])

The network of arguments is always dynamically constructed throughout the design process. Nodes will have one of four possible values – *satisfied*, *denied*, *conflict* or *unknown* – that indicates the decision status for that node. These values are propagated over the network through a procedure indicated by [Chung, 1].

4.6.3 A different approach on Goal Driven Business Modelling

[Jac. & Holt., 1] believe that the role of goals has to change from a starting point of a top-down satisfaction to central criteria driving all decisions within the design process. Goals have to be used to estimate current models, to evaluate single alternatives, and thus help to guide the development process according to the visions on the to be built information system.

Therefore, they introduce an approach which uses goals as a means to drive and manage the business modelling process. The *Process Meta Model* (PMM), developed in ESPRIT project 6353, NATURE (Novel Approaches Theories Underlying Requirements Engineering), combines the decision-oriented approach, following the observations of [Suchman, 1] and the experiences of [Grosz & Roll., 1].

The central concepts of the PMM are *intention*, *product*, and *context* (see Figure 4.16).

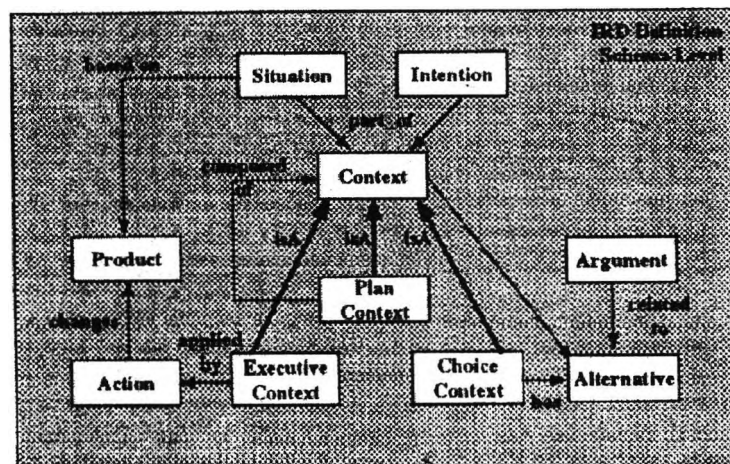


Figure (4.16): The Process Meta Model by ([Pohl *et al.*, 1])

- *Intention* describes the goals to be reached in a certain situation. In other words, it offers the rationale of product transformations.
- *Product* is any kind of document. Speaking of business models the documents describing the different perspectives on the business can be regarded as products.
- *Context* combines situations and intentions. There are three specialisations of context:

1. *Choice context* models possible choices the developer can make during the process concerning the former processes' direction. There exist *arguments* which correspond to the possible *alternatives*.
2. *Executive contexts* directly result in transformations of certain products.
3. *Plan contexts* define a certain order in a set of any contexts. In this part, sequences and process abstractions can be defined.

According to [Jac. & Holt., 1], the basic idea of PMM is to combine the actual product, the envisaged goals, and different kinds of alternatives to proceed. This can be regarded as a general framework, which combines performing, planning, and decision making. Figure 4.17 shows obviously that only the left part of the whole PMM is used, i.e. business modelling can be regarded as goal satisfaction or product transformation.

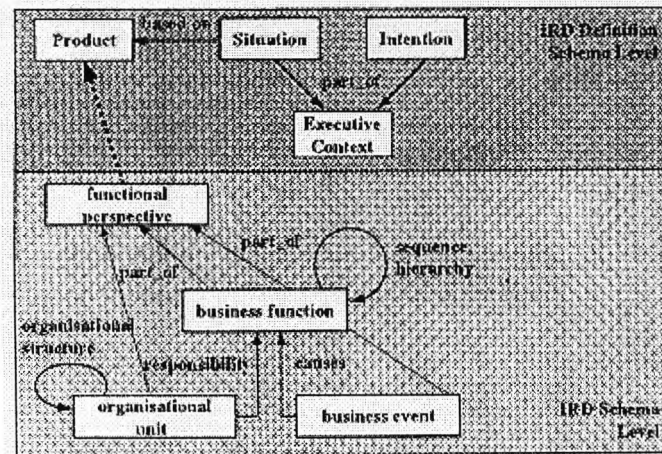


Figure (4.17): Integration of the functional view by into the PMM ([Jac. & Holt., 1])

Goals are used twofold and thus are instantiated twice in the PMM. First, goals are stepwise transformed from abstract goals into concrete design decisions. But goals are not only the starting point of development processes. They also describe the objectives to be reached in certain situations, i.e. they describe the rationale of a transformation. Thus, they are instantiated as *intentions* within the PMM, i.e. they can be regarded as part of the *choice context*. More concretely, goals can be used as criteria to evaluate the benefits and shortcomings of contexts in a given situation.

4.6.4 Driving Business Modelling with Goals

Modelling goals as intentions has two benefits [Jac. & Holt., 1]:

- First, the modelling process is not only guided by structural aspects of the business model but by the visions and objectives of the system to be built.
- Second, decisional aspects of the development process can be expressed in the common model. Therefore, we can answer systematically questions like: *Does the current state fulfil all requirements or do we need to continue the iteration in the development process?*

Business modelling focuses on structural aspects, that is relationships between different perspectives are defined, graphical notations are offered, etc. Business modelling is a transformation process, but [Jac. & Holt., 1] quote that control and evaluation of situations and products can also be succeeded. In the PMM, evaluation is a part of the *Choice context*. Knowledge about the goals, and estimation of products is the foundation of decision-making within the modelling process, i.e. deciding how to proceed in the modelling process.

Business modelling in this case is supported by Qualitative Decision Support, which is well-suited when dealing with lots of complex and ill-structured information that can hardly be formalised (e.g. the goals). Qualitative decision support means to arrange the foundation of the decision in a way that interconnections, relations, and dependencies become clearer.

4.6.5 Comments on Goal-driven methodologies

A methodology for designing and re-designing business processes by combining functional goals and rules (the AD model) with non-functional goals, methods, and correlation rules (the IA model) into a single framework has been presented. In this approach, intentional concepts such as goal, ability, and commitment are used to characterise the *expectations* that actors have of each other's behaviour. By using the concept of intentional dependency, one is able to take a structural view of a business process, without assuming that actors behave deterministically.

Actors can act freely within the confines of dependencies. The dependency types in the AD model indicate the different types of freedom and constraints that exist in the

relationships among actors. Actors may violate expectations and commitments. Rules provide a systematic approach for arriving at the partial solution – the AD structure.

For finer differentiation among alternatives based on additional quality dimensions, [Yu & Mylop., 1] overlay on top of an AD model another level of reasoning – the IA model. Here a qualitative, argumentation framework is used because of the typically “non-functional” or “soft” nature of these design goals. The distinctive characteristics of this level of reasoning are the use of methods and correlation rules that produce contributions towards goals. The method needs to be practically applied, therefore the development of algorithms for use with the framework and testing the adequacy of the set of modelling concepts, might be a possible area of research.

As to the approach suggested by [Jac. & Holt., 1], goals are not only used as the source of the whole development process, but as the overall intention of each action during the development process. Thus, the result (product) of each action can be evaluated against goals. Goals are used as criteria to drive the decisions within the development process. The goals are used as decisional criteria, whereas the single perspectives of the business model are the subject of decisions.

4.7 Part IV: Bayesian Networks

Bayesian networks have attracted a great deal of attention, during the early 1990s. Contrary to most other expert system techniques, a good deal of theoretical insight as well as practical experience is required in order to exploit the opportunities provided by Bayesian networks [Jensen, 2].

The first *expert systems* were constructed in the late 1960s. Their scope is decision-making characterised by repeatedly deciding on almost similar cases. A so-called *rule-based system* consists of a *knowledge base* and an *inference system*. The knowledge base is a set of production rules and the inference system combines rules and observations to come up with conclusions on the state of the world and on actions.

A way to incorporate *uncertainty* in rule-based systems is to extend the production rules to the format:

if condition with certainty x **then** fact with certainty $f(x)$, where f is a function.

Normative expert systems are an alternative to rule-based expert systems. Both deal with repeated decision making on almost similar cases, but the design principles for normative systems differ from rule-based systems in three ways:

- instead of modelling the expert, model the domain;
- instead of using a noncoherent uncertainty calculus tailored for rules, use classical probability calculus and decision theory;
- instead of replacing the expert, support him/her.

4.7.1 Causal networks

A *causal network* consists of a set of *variables* and a set of *directed links* between variables (mathematically, the structure is called a *directed graph*). When talking about the relations in a directed graph we use the wording of family relations: if there is a link from *A* to *B* we say that *B* is a *child* of *A*, and *A* is a *parent* of *B*. In a causal network, a variable represents a set of possible states of affairs. A variable is in exactly one of its states; which one may be unknown to us.

Reasoning about uncertainty also has a quantitative part, namely calculation and combination of certainty numbers. Whenever calculus is used, it must obey the below stated rules:

1. Evidence may be transmitted through a *serial connection* unless the state of the variable in the connection is known. When the state of a variable is known we say that it is *instantiated* (Figure 4.18).

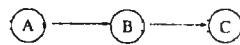


Figure (4.18): Serial connection. When *B* is instantiated it blocks communication between *A* and *C*
 ([Jensen, 2])

2. Evidence may be transmitted through a *diverging connection* unless it is instantiated (Figure 4.19).

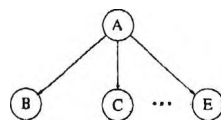


Figure (4.19): Diverging connection. If *A* is instantiated, it blocks communication between its children
 ([Jensen, 2])

3. If nothing is known about A except what may be inferred from knowledge of its parents B, \dots, E , then the parents are independent. If any other kind of evidence influences the certainty of A , then the parents become dependent. Hence, evidence may only be transmitted through a *converging connection* if either the variable in the connection or one of its descendants has received evidence (Figure 4.20).

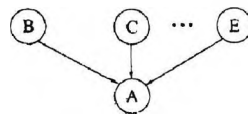


Figure (4.20): Converging connection. If A changes certainty, it opens communication between its parents ([Jensen, 2])

Evidence on a variable is a statement of the certainties of its states. If the statement gives the exact state of the variable, it is called *hard* evidence, otherwise it is called *soft*.

Definition: Two variables A and B in a causal network are *d-separated* if for all paths between A and B there is an intermediate variable V such that either:

- the connection is serial or diverging and the state of V is known
- or
- the connection is converging and neither V nor any of V 's descendants have received evidence.

The concepts of causal network and d-connection are due to [Pearl, 2] and [Verma, 1].

4.7.2 Bayesian networks

So far nothing has been said about the quantitative part of certainty assessment. The basic concept in the Bayesian treatment of certainties in causal networks is *conditional probability*. A conditional probability statement is of the following kind:

Given the event B , the probability of the event A is x .

The notation for the statement above is $P(A|B)=x$. Also, for these kind of probabilities the well known Bayes' rule stands:

$$P(B \setminus A) = \frac{P(A \setminus B)P(B)}{P(A)}$$

For more details on the properties of conditional probabilities refer to [Jensen, 2].

Causal relations also have a quantitative side, namely their *strength*. This is expressed by attaching numbers to the links. A *Bayesian network* consists of the following:

1. A set of *variables* and a set of *directed edges* between variables.
2. Each variable has a finite of mutually exclusive states.
3. The variables together with the directed edges form a *directed acyclic graph* (DAG). (A directed graph is acyclic if there is no directed path $A_1 \rightarrow \dots \rightarrow A_n$ such that $A_1 = A_n$).
4. To each variable A with parents $B_1 \dots B_n$ there is attached a conditional probability table $P(A \setminus B_1, \dots, B_n)$.

One of the advantages of Bayesian networks is that they admit *d-separation* (for a proof of that see [Lauritzen, 1]: if A and B are d-separated in a Bayesian network with evidence e entered, then $P(A \setminus B, e) = P(A \setminus e)$. This means that one can use d-separation to read-off conditional independencies.

Bayesian networks have a long history in statistics, and in the half of the 1980s they were introduced to the field of expert systems through work by [Pearl, 1] and [Spieg. & Knill-Jones, 1].

4.7.3 Building models

Bayesian networks create a very efficient language for building models of domains with inherent uncertainty. However, most of the time performing the necessary calculations is a tedious job, but fortunately, software tools, which can do the calculation job is available.

The first thing to have in mind when organising a Bayesian model for a decision support system is that its purpose is to give estimates of certainties for events which are not observable (or only observable at an unacceptable cost). These events are called *hypothesis events*. The hypothesis events have to be organised into a set of

variables, each of them incorporates an exhaustive set of mutually exclusive events. That is, for each variable precisely one of its events is true.

The next thing to have in mind is that in order to come up with a certainty estimate, one should provide some *information channels*. So, the task is to identify the types of achievable information, which may reveal something about the state of some hypothesis variable. This is also done by establishing certain variables, *information variables*, such that a piece of information corresponds to a statement about the state of an information variable. Typically, this information will be a statement that a particular information variable is in a particular state.

After that, it is time to consider the causal structure between the variables. At this stage, there is need to worry about how information is transmitted through the network, and particularly, which events have a direct causal impact on other events. Variables in a model, which are neither hypothesis variables nor information variables, are called *mediating variables*. Usually, these variables will ease the acquisition of conditional probabilities and thereby also increase the precision of the model [Jensen, 2].

4.7.4 Modelling tricks

If one is building a model over a domain, where experts actually do take decisions based on estimates, one should be able to make his/hers Bayesian network estimate at least as well as the experts. The acquisition of numbers is of course not without problems, and there are some methods which can help someone in this job. Please refer to the Appendix for a detailed list of the methods.

4.7.5 Example:

The following example is taken by [Jensen, 2]. To help the bank decide when a customer applies for a mortgage on a house, the customer is asked to fill in a form giving information on various economic and personal matters. The answers are used to estimate the probability that the bank will get their money back. The information is the following: type of job, yearly income, other financial commitments, number and type of

cars in the family, size and age of the house, price of the house, number of the previous addresses during the last five years, number of children in the family, number of divorces, and number of children not living in the family.

In principle, each slot in the form represents a variable with a causal impact on the variable *money back?*. However, the information can be partitioned into variables describing the economic potentials of the applicant, variables describing the stability of the applicant, and variables describing the security of the mortgage. So the many parents can be divorced by three variables.

4.7.6 Junction trees

[Jensen *et al.*, 1] introduces an algorithm for probability updating in Bayesian networks. The algorithm does not work directly on the Bayesian network, but on a so-called *junction tree* which is a tree of clusters of variables. The clusters are also called *cliques* because they are cliques in a triangulated graph, which is a special graph constructed over the network. Each clique holds a table over the configurations of its variables, and HUGIN propagation consists of a series of operations on these tables. The HUGIN algorithm yields the exact updated probabilities.

4.7.6.1 Cluster trees

A *cluster tree* over U is a tree of clusters of variables from U . The nodes are subsets of U , and the union of all nodes is U . (A tree is an undirected graph without cycles). The links are labelled with *separators* that consist of the intersection of the adjacent nodes. Each node and separator holds a real numbered table over the configurations of its variable set (see Figure 4.21).



Figure (4.21): A Bayesian network and a corresponding cluster tree. Separators are in square boxes
([Jensen, 2])

Now, let BN be a Bayesian network over U . A cluster tree corresponding to BN is constructed in the following way:

- form a family of nodes such that for each variable A with parent set $pa(A)$ there is at least one node V such that $pa(A) \cup \{A\} \subseteq V$;
- organise the nodes as a tree with separators (so far there is no restriction on how you organise the tree);
- give all nodes and separators a table of ones;
- for each variable A choose exactly one node V containing $pa(A) \cup \{A\}$ and multiply $P(A \mid pa(A))$ on V 's table.

4.7.6.2 Junction trees

Here, we shall give a method for constructing junction trees for DAGs (directed acyclic graph). A DAG is *singly connected* if the graph you get by dropping the directions of the links is a tree (see Figure 4.22). For singly connected DAGs it is easy to construct junction trees.

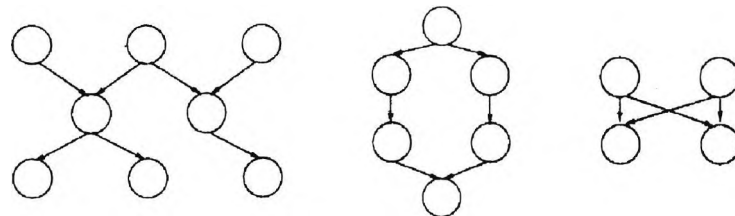


Figure (4.22): Examples of singly connected and multiply connected DAGs ([Jensen, 2])

For each variable A with $pa(A) \neq \emptyset$ you form the cluster $pa(A) \cup \{A\}$. Between any two clusters with a non-empty intersection one adds a link with the intersection as a separator. The resulting graph is called a *junction graph*. All separators consist of a single variable and if the junction graph has cycles, then all separators on the cycle contain the same variable. Therefore, any of the links can be removed to break the cycle, and by removing links until you have a tree, one gets a junction tree (see Figure 4.23).

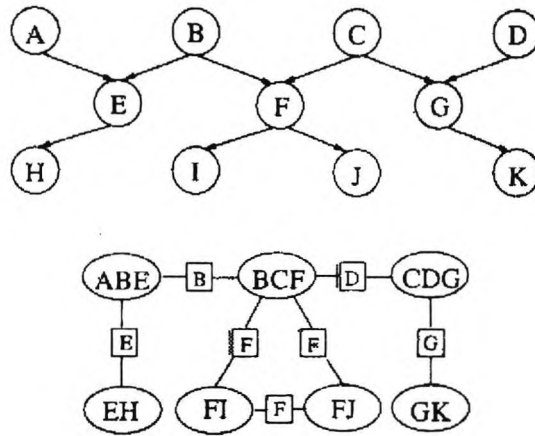


Figure (4.23): A singly connected DAG and its junction graph. By removing any of the links with separator F one gets a junction tree ([Jensen, 2])

For all variables A there must be a cluster V containing $pa(A) \cup \{A\}$. We can illustrate this on a graph by having a link between any pair of variables which must appear in the same cluster. This means that we take the DAG, add a link between any pair of variables with a common child, and drop the direction of the original links. The resulting graph is called the *moral graph*. From this graph, one can read the clusters to consider, namely the cliques in the graph (see Figure 4.24).

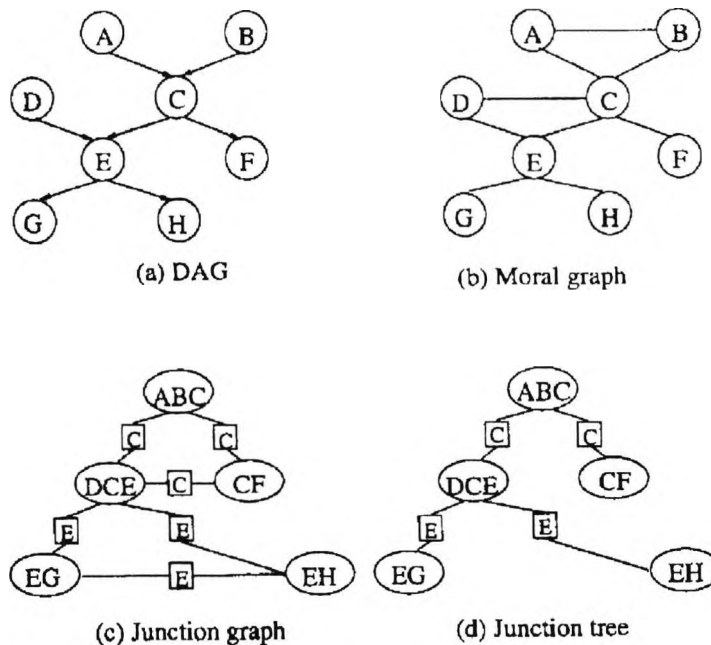


Figure (4.24): Construction of a junction tree for a singly connected DAG ([Jensen, 2])

In case the junction graph contains cycles that cannot be broken, the problem is solved by adding the so-called *fill-ins* to the moral graph. The general rule for filling-in

the moral graph is that any cycle with more than three variables shall have a chord. In this case the graph is called *triangulated*. In Figures 4.25 and 4.26 there is another example of the process from DAG to junction tree.

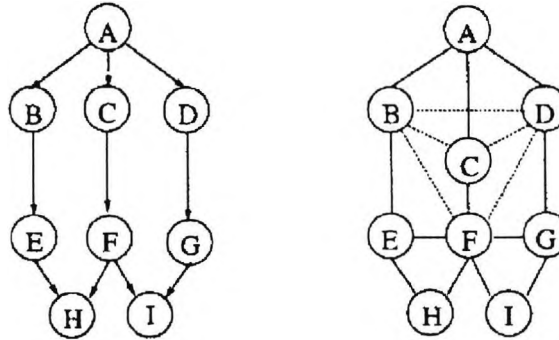


Figure (4.25): A DAG, the moral and triangulated graphs. The fill-ins are indicated by dotted lines ([Jensen, 2])

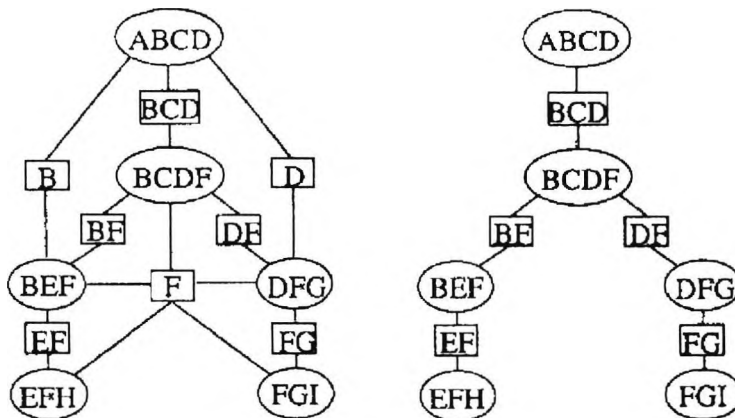


Figure (4.26): The junction graph for the triangulated graph in figure 4.25 and a junction tree ([Jensen, 2])

In HUGIN propagation [Jensen *et al.*, 1], the cliques in the junction graph shall have joint probability tables attached to them. The size of the table is the product of the number of states of the variables. So, the size increases exponentially with the size of the clique. Therefore, a good triangulation is a triangulation yielding small cliques, or to be more precise, yielding small probability tables.

The concepts of triangulated graphs and junction trees have been discovered and rediscovered with various names. In [Bert. & Brios., 1] they are used for dynamic programming, and [Beeri *et al.*, 1] use them for database management. [Golumbic, 1] speaks about triangulated graphs and [Tar. & Yan., 1] give various triangulation methods and very efficient methods for testing whether a graph is triangulated.

4.7.7 Actions

A Bayesian network serves as a model for a part of the world, and the relations in the model reflect causal impact between events. The reason for building these computer models is to use them when taking decisions. That is, the probabilities provided by the network are used to support some kind of decision-making. There are two kinds of decisions, namely *test-decisions* and *action-decisions*.

A test-decision is a decision to look for more evidence to be entered into the model, and an action-decision is a decision to change the state of the world. In order to be more precise, it should be stated that decisions have two *aspects*, namely a test aspect and an action aspect, which are handled differently in connection with Bayesian networks.

Actions should also be divided into two types, namely *intervening actions*, which force a change of state for some variables in the model, and *non-intervening actions* of which the impact is not part of the model. Impact of the intervening actions can only follow the direction of the causal links.

Decision problems are treated in a certain framework, called *utility theory*. The various decisions should be evaluated on the basis of the usefulness of their consequences. "Usefulness" is measured on a numerical scale called *utility scale*, and if several kinds of utilities are involved in the same decision problem, then the scales have a common unit. This assumption is treated in the extensive literature on utility theory, for example [Lindley, 1] and [Winter. & Edwards, 1].

Let $A = (a_1, \dots, a_n)$ be a set of mutually exclusive actions, and let H and G be determining variables. What is required in order to specify the problem of deciding between the actions in A is a *utility table* $U(A, H, G)$ yielding the utility for each configuration of action and determining variables.

[Jensen, 2] gives examples and describes the way the intervening and non-intervening actions are treated through utility theory. Decision theory has a long history. The structure that [von Neum. & Morg., 1] have used was *decision trees* but later on *influence diagrams* were proposed as an alternative to decision trees in [Howard & Matheson, 1]. Methods for solving multiple decision problems through influence diagrams are constructed by [Shachter, 1], [Shenoy, 1], and [Jensen *et al.*, 2] use HUGIN propagation for solving influence diagrams.

4.7.8 Comments on Bayesian networks

Bayesian networks is a formal modelling technique for constructing decision support systems or expert systems. We have seen so far the basic principles of Bayesian network theory and we have addressed the primary concepts encompassed in this theory.

Bayesian networks create a very efficient language for building models of domains with inherent uncertainty. The area of application of Bayesian networks is very wide; agriculture, computer vision, computing, information processing, and medicine are only some of the areas, where one finds BN-based tools developed for different activities. To these, we can add the *Hailfinder* developed to forecast severe weather in the plane of north-eastern Colorado [Abramson *et al.*, 1], and *FRAIL*, which is an automatic Bayesian network construction system [Gold. & Charn., 1].

The link of Bayesian networks with influence diagrams bears the aspiration that *BN* could may as well be used in the creation of more powerful soft models. They could provide us with models for business processes, which are enriched with more information and ability to decide on whether a process should take place.

4.8 Conclusion

Chapter 4 has provided the reader with a thorough account of a number of modelling techniques. All these succeed in describing the dynamic aspects of systems' behaviour despite the diversity in their technology. One could even argue that these modelling methods present significant complementarities. Although their majority have been created and applied to serve the modelling purposes of physical and chemical systems, their capabilities suggest that there is room for much-promising results if these methods are used in the manipulation of business processes.

As it has been presented, Qualitative Reasoning builds the conceptually unified theoretical framework for developing and simulating qualitative models of systems, where knowledge of those systems is incomplete. The paradigm of business processes involves systems like these, where well-defined mathematical models do not exist. An idea would be to include the four parts - preconditions, quantity conditions, relations and influences – in the specification of a process (just like in Qualitative Reasoning)

and use them to enrich the model derived by System Dynamics. Once more, both preconditions and quantity conditions must be true for a process to be acting. The preconditions will be those external factors such as the sudden rise in the demand for projects. The quantity conditions will be those limits that can be deduced within the present theory, such as for example, requiring the discrepancy between the required workforce and the equivalent workforce to be positive for hiring to take place. Relations will hold between quantities affected by the process, and influences will be the contributions to the way a quantity changes.

Simulation of business process models could also benefit from Qualitative Reasoning. As we know, simulation of a continuous or a discrete time system can be achieved, basically in two ways, quantitatively and qualitatively. Quantitative simulation entails an exact description of the relationships between the various quantities involved in the system functioning, in terms, e.g., of algebraic differential or difference equations. Qualitative simulation, on the other hand, exploits relationships that express qualitative connections between the quantities. Although such a description does not contain as much information as in a quantitative analysis, it permits however, the expression of incomplete knowledge and provides general solutions for classes of systems rather than numerical solutions of each case, treating homogeneously linear and non-linear systems. Unfortunately, it fails to provide effective mathematical tools for carrying out the simulation.

[Fanni & Giua, 2] have proposed a simple way of avoiding this problem. Since a qualitative system, with its discrete-quantity state space can be seen as a discrete event system (i.e., as dynamic system, in which the state changes only at discrete points in time, in agreement with unexpected occurrence of physical events), its behaviour may be described by using Petri net models. Firstly, a qualitative description of the continuous time behaviour is given. Secondly, the qualitative descriptions are captured by discrete-event models. Finally, a model that integrates all these different discrete-event behaviours is built using Petri nets. This proposal suggests one more link between Petri Nets – a method that will be thoroughly presented in Chapter 8 – and Qualitative Reasoning, aiding the simulation of soft systems.

Order-of-magnitude reasoning enhances Qualitative reasoning by focusing on the relations between the parameters, achieving in this way, more expressive power and effective reasoning. On the other hand, Multilevel Flow Modelling (MFM) takes modelling a step further by introducing functions in the systems' structure. The method

is based on concepts of goals and functions, which traditionally are not used within the natural sciences but used within the human sciences. It provides a systematic approach in coping with large-scale decision problems, where the models tend to be very complex and unmanageable because the level of detail introduces so many unknown factors.

The model, which derives, is multilevel because it comprises several levels of abstraction and it is about flow because every functional abstraction is described in terms of flow structures. The MFM model succeeds in providing some insight in the system as it represents the interdependency between the resources in the system provided for goal achievement. Business process modelling could incorporate some of the concepts of MFM in order to produce a functional process decomposition that is crucial to solving complex diagnosis problems. The truth of this argument lies on the link between MFM and Goal-driven methodologies.

The latter capture the intentional dimension of organisational work, for a deeper understanding of how a business process operates. They also include a reasoning structure, which shows the relationships of activities within a process among a set of goals. This has two benefits: First, the modelling process is not only guided by structural aspects of the business model but by the visions and objectives of the system to be built. Second, decisional aspects of the development process can be expressed in the common model. Perhaps, a combination of Role-activity diagrams and goal-driven methods would provide a powerful tool for the re-design of a process; the first would capture the roles, their components activities and their interactions, together with external events, while the latter would dwell on the intentional dimension of this organisational structure.

Last but not least, Bayesian networks create a very efficient language for building models of domains with inherent uncertainty. The link of Bayesian networks with influence diagrams bears the aspiration that BN could may as well be used in the creation of more powerful soft models. They could provide us with models for business processes, which are enriched with more information and ability to decide on whether a process should take place.

Chapter 5

CONCEPTUAL MODELLING AND THE SOFT SYSTEMS APPROACH

5. CONCEPTUAL MODELLING AND THE SOFT SYSTEMS APPROACH

5.1 Introduction

The soft systems approach is a viable approach to business process modelling; it proves very useful in identifying loosely specified processes as the ones we find in the analysis of 'soft systems'. By 'soft systems', we mean those, which involve human activity or human judgement, such as the activities of business management [Kingston, 3]. The soft systems approach offers the ability of drawing together disparate approaches to a process; however, it lacks guidance during its use and requires quite a lot of time to produce a detailed model.

Conceptual modelling is used in order to, amongst others, assist the communication between analysts and end-users during the phases of facts acquisition and specification-verification [Louc. & Zic., 1]. It takes place in the early phases of system development. It is concerned with the investigation of the problems and requirements of the issue of research, leading finally to the definition of the requirements specification of the desired system. Conceptual models mostly consist of modelling concepts and guidelines related to a language for specifying both the structure and the behaviour of a system.

In this chapter, the main concepts of these two methodologies are presented, and their linkage is discussed. The development of the conceptual model of the consulting firm's problem is elaborated and the power of conceptual model unravels soon after.

5.2 The Soft Systems Approach

5.2.1 The development of Soft Systems Approach

The soft systems approach was developed in the early seventies and was mainly based on the work of Peter Checkland at Lancaster University [Checkland, 1]. The essence of his approach is to identify one or more problem areas in a system and then to model only those business processes that are relevant to that problem area. The researcher calls this identification of a "relevant" system. He models the system by

representing processes as nodes in a diagram, and indicating the flow from one process to another using arrows. In case of complex processes, these are expanded into sub-processes in separate diagrams.

An example of such modelling is shown below in diagrams copied by [Checkland, 1]. The system is concerned with a consultant who is having problems providing his client with the type of report that the client desires. In this case, the "relevant" system is chosen to be the consultant/client relationship simply as a transfer of advice from one party to the other; all the other aspects of this relationship are ignored. The resulting system is shown below:

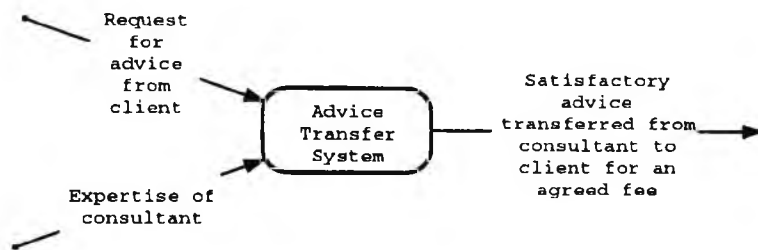


Figure (5.1): The advice transfer system at the highest level of abstraction ([Kingston, 3])

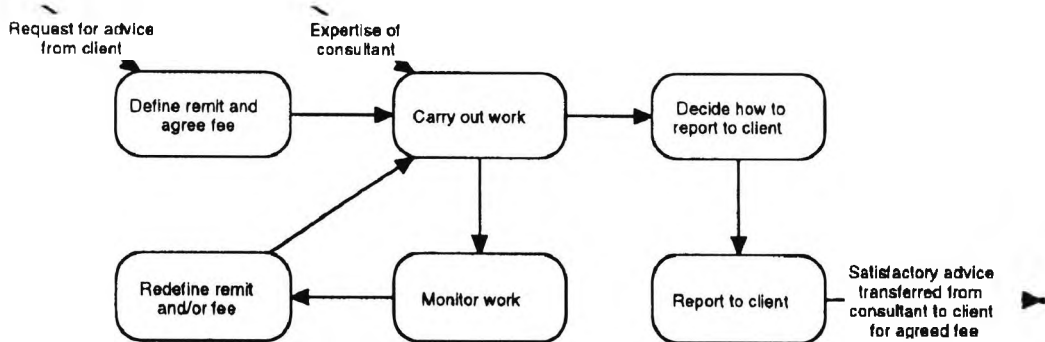


Figure (5.2): The advice transfer system expanded into individual processes ([Kingston, 3])

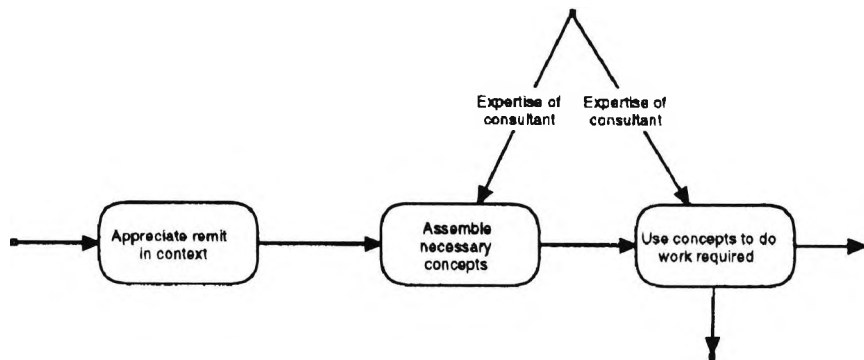


Figure (5.3): The process "Carry out work" is broken down to sub-processes ([Kingston, 3])

In his approach, Checkland identifies a number of key attributes relating to each process. These attributes are:

- Actor – who (or what) normally performs the action
- Customer – whom the action is intended to benefit
- Owner – who has the authority over decisions made
- Environment – the environment in which the soft system operates
- Worldview – the underlying assumptions which participants in the soft system hold

The soft systems approach requires representation of:

- individual processes which are carried out as part of an overall process;
- attributes attached to individual processes;
- data flows between processes;
- the ability to have more than one data flow to or from a process;
- the ability to break down a single process into a sequence of subprocesses

By identifying a relevant system, decomposing complex processes into subprocesses and finally identifying key attributes of each process, we produce a rich representation of a particular business process. Once a “relevant” system has been identified, Checkland’s approach continues with modelling the “ideal” version of the relevant system. This is achieved by presenting the model to the participants in the process, and then iteratively refining the model based on elicited comments, i.e., regular meetings and interviews. This iterative process carries on until all participants agree that the model represents the ideal processes of this soft system.

5.2.2 A bird’s eye view of Checkland Methodology

Checkland’s methodology is a process of inquiry with a number of distinct stages, which passage through is usually iterative rather than linear. The basic idea is that every problem exists in a particular context, and the context may be perceived differently by different people. The stages of the methodology are:

Task 1: To assemble a representation or picture of the situation which is rich in both quantitative or factual, and qualitative and subjective information.

Task 2: The analyst seeks general patterns or aspects, which express or encapsulate characteristics of the situation.

Task 3: Search for systemic ways of viewing the situation, i.e., find some potentially fruitful way of viewing the problem situation by thinking up some notional systems, which might bear on significant aspects of it.

Task 4: Derive the conceptual model that contains all the essential activities, which the notional system would logically have to perform.

Task 5: Compare the conceptual model with what is perceived to exist in the actual problem situation.

The technique for building conceptual models is based on very simple principles. A model of a human activity system will contain a set of activities connected together. The basic language used for model construction is therefore all the verbs in the analyst's speaking language. The model will contain the minimum number of verbs necessary for the system to be the one named and concisely described in the *root definition*. The aim is to build an activity model of *what* must go in the system. Particular *hows* may be included in subsequent more detailed models obtained by the expansion of the first-level model [Checkland, 1]. Conceptual model is, at this phase, simply the product of combining graphics and natural language, in order to organise and depict all relevant information.

5.2.3 The use of HARDY in the modelling of Soft Systems

HARDY is a diagramming tool, which supports the creation of nodes and arcs, whose shapes, sizes and attributes are defined by the user. The resulting diagrams can be manipulated in different ways; in addition, HARDY supports an integrated programming language, which allows the definition of menu options that automatically carry out one or more diagram manipulations. HARDY can also be customised to produce graphical support tools for a variety of methods and approaches. For the soft systems approach, HARDY provides the following formats:

- Processes – they are presented as nodes on a hypertext card that bears the name of the process.
- Attributes – each process node has its own attributes. HARDY supports the creation of named attributes for both nodes and arcs.

- Data flows – between processes are represented as arcs with arrowheads between process nodes. We can put labels on the arcs to indicate what data is being transferred.
- Expansion of a process into subprocesses – HARDY allows the automatic creation of “expansion” cards, so that the ‘parent’ process is linked to another hypertext card and all the subprocesses the new card might include.

An example of modelling a soft system by using HARDY is given by [Kingston, 3]. Here the commercial sales process in a small company was to be modelled. Through interviews of experienced individuals, it was decided that the biggest areas of uncertainty in this process were in the following up of an initial proposal for business, and in the estimation of the likelihood of closing a potential sale. Therefore, it was decided that the “relevant” system was a “Bid Management System”, in which the sales process was represented as a system for preparation, refinement, and resourcing of proposals for business (“bids”). The final top-level model of the Bid Management system, represented graphically with HARDY, can be seen in Figure 5.4.

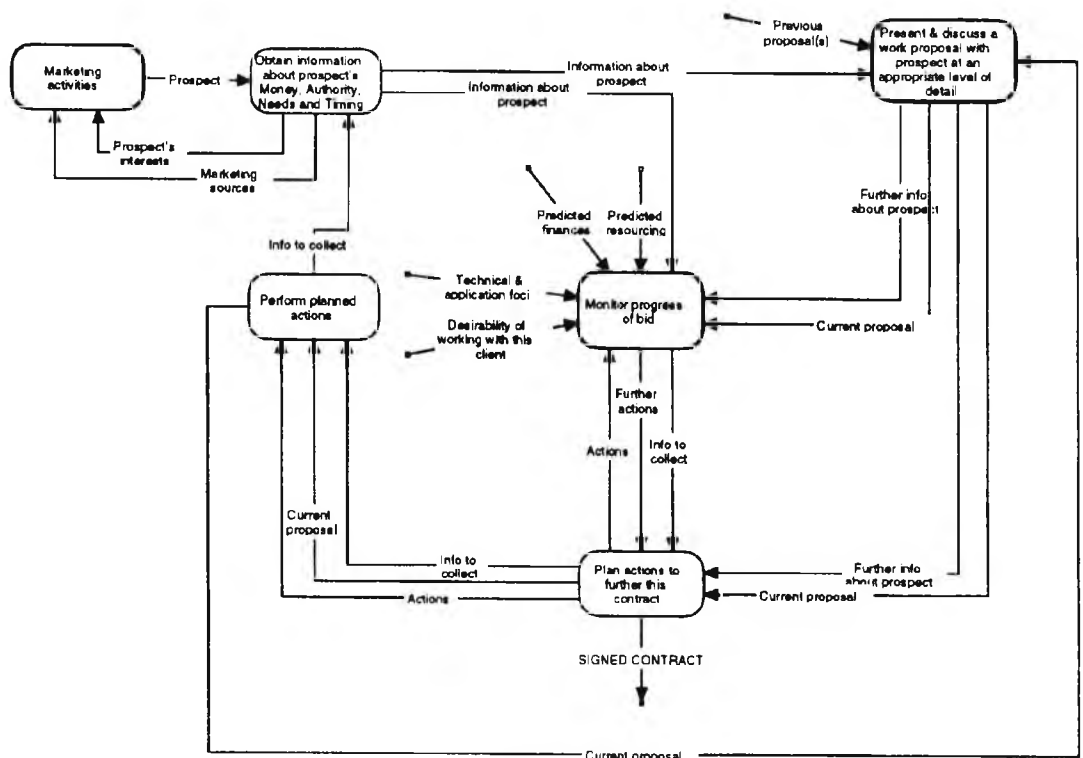


Figure (5.4): The top level of the Bid Management system ([Kingston, 3])

5.2.4 Guiding Soft Systems modelling through CommonKADS

The CommonKADS methods [Wielinga *et al*, 1] have been developed in an effort to provide some additional guidance to the modelling of soft systems. They offer a series of models, which represent the analysis, and design of a knowledge-based system (or expert system). CommonKADS provides a library of “generic” inference structures – models that represent the reasoning steps performed as part of the overall task – which are used as a starting point for the development of inference structures for particular projects. This library has been used in a number of projects [Kingston, 1] and [Kingston, 2]. Some typical task types, in which this library has been useful, are diagnosis, configuration and design. More details on the implementation of CommonKADS are provided in [Kingston, 3].

5.2.5 General comments on the Soft Systems Approach

The soft systems approach offers hope of introducing a structure to a loosely defined problem, by identifying and focusing on a “relevant” system [Kingston, 3]. Through the process of modelling a relevant system, the individual approaches to produce the ideal system are synthesised and that means that the recommendations at the end of this analysis stand a reasonably good chance to be accepted by all parties, making Soft Systems a very good initial step in the modelling process of a business activity. This methodology tackles both the intellectual problems of interpretation, analysis and synthesis involved in conceiving ideas for change. Thus, it uses powerful ideas, like the concept of system, to generate insight into a problem situation, but only in conjunction with a process of inquiry and debate which incorporates people in discussion of the implications of abstract ideas [Naug., 1]. The soft systems approach also helps to identify unstated assumptions, and disagreements on underlying principles, when defining actors, customers, owners, environment and worldview. As it will be shown in the following, this approach aids the process of knowledge elicitation, and this is the first stage in the conceptual modelling process.

One of the main disadvantages of the soft systems approach is that although it permits multiple inputs and multiple outputs to processes, it does not explicitly record which outputs depend on which input(s) [Kingston, 3]. The only way to surpass this

limitation is by breaking down any process with multiple inputs into subprocesses until each process at the lowest level has only one input. Another weakness of the approach is the lack of guidance on selection of a relevant system, and on choosing what to model. Checkland has provided a little guidance on selection of a relevant system [Checkland & Scholes, 1] but this needs to be expanded.

The ever-increasing complexity and size of business processes has called for high level concepts and formal techniques so as to model these processes at different levels of abstraction. Particularly nowadays, the continuously changing and restructuring business can only be modelled by a dynamic system supporting life cycles of its components [Leo & Karc., 1]. Therefore, the need for a more powerful tool to cope with all these issues has arisen, and that is how the birth of conceptual modelling has taken place.

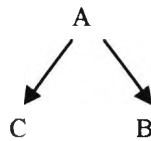
5.3 Conceptual Modelling

5.3.1 The mind and the origins of conceptual modelling

The subject of how the human mind works as a mechanism and what kind of tools (machinery) it employs in order for the brain to operate, has always been an intrinsic subject of interest for many scientists coming from different disciplines. Is there, after all, a specific “formal” procedure in the operation of the mind that we could possibly conceive? How could we describe the void between the information that the senses sent to the brain, and the instructions of other decisions that issue from it, i.e., how is the input to the brain changed to an output from it? In his paper, [Bronowski, 1] argues that the symbols with which the brain works are physical, chemical, and electrical and that the signs, which it employs in its reasoning, must constitute a formal language that follows precise rules just any other language of symbols in which people write out logical and mathematical arguments, does. He further quotes that nature cannot be represented in the form of a logical machine operating on a basic set of axioms by making formal deductions from them in the exact language. We believe that the establishment of a new axiom is simply a free play of the mind, an invention outside the logical processes, and that the language, that we use in describing nature, imposes the form and the limitations of our model.

Strictly speaking, if we are to regard conceptual modelling in terms of a philosophical approach and not just as a modern methodology developed for the construction of databases, we can find its origin in the very first attempts of people to conceive the world. What is of immense interest, is the subconscious effort, found in the “*Dialectic and Dichotomous Division Method*” of Plato ([Jowett, 1], [Taylor, 1]) to introduce a “formal” guidance for reasoning. Plato’s method appears in a number of his works, but it is in the *Sophist* where it is explicitly elaborated. This method is applied in any case where the composition of a satisfactory definition is attempted. Plato chooses a parameter X to define and then works through a number of steps:

1. Start with a bigger and more familiar category A
2. Invent a division of category A into two main sub-categories B and C. Category B has a feature of X while category C has not. Category B is the right sub-category and C is the left one.



3. Keep on the procedure by leaving each time the left subdivision unexamined. Elaborate only the right sub-category by dividing it in smaller and smaller sub-categories to some extent where one right sub-category agrees with the initial parameter X. The definition of X results from the successive characteristics of right subdivisions.

The above-mentioned procedure is instigated by the identification of a specific goal or objective that will guide the research. Although the way that the human mind perceives the issue of study and further defines it, still remains a mystery, in Plato’s method, the definition of the initial parameter is driven by the selection of “relevant” to the goal concepts when dividing a category into sub-categories. What is considered to be relevant and what not is a matter of the inner mechanism of the brain and further elaboration of the question is beyond the scope of this thesis. Nevertheless, we believe that conceptual modelling, as perceived in contemporary times, also commences its modelling process by introducing the overall objective of the problem and then identifying the relevant entities and their relationships, perhaps through a series of questions and discussions, i.e., interviews, among the people involved in the problem.

5.3.2 Basic concepts

Conceptual modelling, taking into consideration the previous paragraph, is not exclusively about the development of a computerised information system. It usually involves a deeper, much more philosophical analysis of an organisation. Even if the ultimate task is to generate an information system, this has to be integrated in the broader environment, hence, it is essential that we have a uniform way of relating it with the other components of the organisation, regardless of the fact that these might never be computerised. The business modelling approach presented here follows the framework adopted by the object oriented methodology and the experiences of conceptual modelling and systems analysis.

In the following lines, some of the main concepts of conceptual modelling will be laid out, starting from the main concept of this methodology, which is the *conceptual schema*. Hence, this unique central description of the information that may be in the soft system will be called a *conceptual schema* and it plays a key role, as it can be later used as a reference for the entire development process of systems. Conceptual schema is produced through a number of tasks. The guidelines for performing these tasks, for choosing the most appropriate task depending on the particular situation, and for helping the modeller to make decisions, in other words, for specifying both the structure and the behaviour of a system, constitute the *conceptual modelling process*.

Definition (5.1): The conceptual modelling process consists of the following stages:

1. Eliciting the knowledge – it involves the acquisition of a comprehensive description of a system, including the understanding of the users' needs, the business rules, and, finally, the selection and classification of relevant to the problem knowledge.
2. Formalising the knowledge – it involves the structuring and refining of the acquired knowledge. Here, the modeller is asked to find the concepts that are best capable of expressing in a formal way the desired system structure and behaviour.
3. Representing the knowledge – it involves the depiction of the various entities, attributes and relations in an easy to communicate form, consisting of arrows and rectangular boxes.

and leads to the construction of a functional schema that serves a pre-defined set of goals.

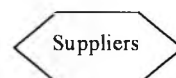
The basic issues in the above-mentioned process are the following:

1. Specification of goals and objectives – setting the goals we aim to achieve when building a model of the problem, drives the whole process of conceptual modelling. This is a very important issue of the process. Goals and objectives are constantly used as a reference to validate the ‘correctness’ of the derived model.
2. Specification of the “relevant” framework of the problem – here the notion of the “relevant” system, as it appears in the Soft Systems approach, is once again used. The essence is to identify one or more problem areas in a system and then to model only those business processes that are relevant to that problem area.
3. Definition of entities and their relations - using the two types of knowledge: explicit and tacit [Non. & Takeuchi, 1], the modeller is able to ‘drill’ the main entities of the problem and identify their relations. Explicit knowledge is formal and systematic, it can be expressed in scientific formulae, codified procedures or a variety of other forms. It consists of two components: a language and information. The language is used to express and code knowledge. Information is coded externalised knowledge. It is potential knowledge, which is realised when information is combined with context and experience of humans to form new tacit knowledge. Tacit knowledge, on the other hand, is embedded in individual experience such as perspective and inferential knowledge. Tacit knowledge includes insights, hunches, intuitions, and skills that are highly personal and hard to formalise, making them difficult to communicate or share with others. Tacit knowledge is deeply ingrained into the context, i.e. the owner’s view and imagination of the world and into his/her experience, which is previously acquired knowledge.
4. Representation of entities and relationships in order to produce the conceptual schema for the problem – the way this schema is constructed, is explicitly described in the following pages.
5. Validation of the derived schema as far as goals and objectives are concerned - the modeller checks whether the conceptual schema is consistent and whether it expresses in full the requirements informally stated by the users.
6. Maintenance of the schema – since user requirements evolve continuously, the existing conceptual schema has to change in order to adapt to new needs or to satisfy new objectives.

The activity of conceptual modelling is a prerequisite for the development of subsequent formal models. Initially, it was used in order to depict an abstraction of the activity of developing an information and decision-making system that recognises the importance of first developing models that are oriented towards the understanding of a particular process. Its purpose did not differ a lot from the Soft System's approach that is to develop an accurate model or graphical representation of the client's information needs and business processes. Nevertheless, the continuously changing and restructuring business can only be modelled by a dynamic system supporting life cycles of its components.

In the past, programming languages, artificial intelligence and databases areas have contributed independently to the development of conceptual modelling. Database design typically emphasised static properties, whereas, programming languages' approach was radically different, giving emphasis on the dynamics [Rolland & Cauvet, 1]. The object-oriented (OO) approach places equal emphasis on structure and behaviour modelling and it succeeds in the integration of the two. The object concept allows the modeller to concentrate on structuring reality in terms of a collection of objects and their interactions. In the OO approach, the elements of the system are *objects*. The emergence of object-oriented concepts has resulted in a number of specific approaches, which diverge on how to perform object-oriented modelling. Throughout this thesis, we have adopted the data approach, as mentioned in [Rolland & Cauvet, 1], that bases the identification and description of objects on an entity/relationship model. Here, the object concept is very close to the one of entity.

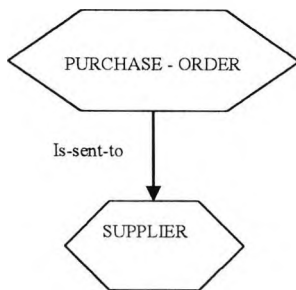
An *Entity* can be anything, concrete or abstract, an object of significance to the business problem. An entity may be a tangible or real object like a person or a building; it may be an activity like an appointment or an operation; it may be conceptual as in a cost centre or an organisational unit. Any lexical object that serves to identify, quantify, classify, or otherwise express a property of an entity is called an *attribute*. Attributes cannot exist in their own; they are only of interest when they are associated with an entity. *Entity class* is called the set of individual entities to which a specific definition and common properties apply. An entity class is represented by a hexagon.



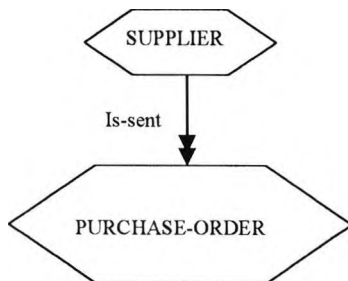
The *attribute class* is the proposition establishing a domain of values for the attribute. An attribute class is represented by an orthogonal.

SUPPLIER
NAME

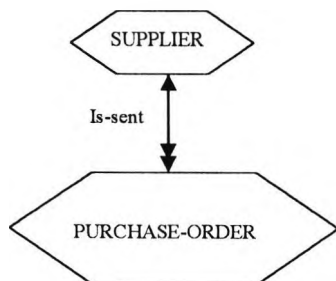
Entities do not exist in isolation but are associated by relationships. A *relationship* is any temporary or permanent association, linkage between two entities or between an entity and an attribute. There are three types of relationships, one to many, many to many, and one to one [Chen, 1]. This denotes the possible restrictions in the relationships, i.e., the number of times an entity or attribute can participate in this involvement. One to one denotes that at any time the two associated entities can have only one valid relationship. One to many denotes that at any time the entity at the one end can have only one valid relationship. Many to many denotes no restrictions. Some examples of the different types of relationship are provided below:



The 'to-one relationship': A purchase-order is sent to only one supplier.



The 'to-many relationship'. A supplier is sent many purchase orders.



The 'one-to-many relationship'.

Both relationships, mentioned previously, can be shown in the same diagram by a single line with a double arrowhead at one end and a single arrow head at the other. In ‘many-to-many relationships’, multiple occurrences of one entity are related to one occurrence of another. Usually, we convert this kind of relationship into two ‘one-to-many’, ‘many-to-one’ relationships, in order to be able to convert it into database tables. Whenever there is a one-to-one relationship between an entity class and an attribute class then each instantiation of the attribute class identifies an instance of the entity class. In this case, the attribute class will be called an *identifier* (we usually underline it in the diagram).

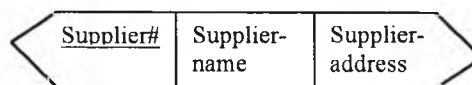


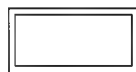
Figure (5.5): Representing the attributes of an entity

In Figure 5.5, we do not denote the different types of relationship between the entity class Supplier and the two attribute classes Supplier-name and Supplier-address. This is a less detailed way to depict the conceptual model of a soft system.

Extensions of an entity class are called *complex entity classes* and are usually created through an abstraction mechanism (aggregation or grouping) of entities (complex or simple), relationships and attributes (complex or simple). They are represented as shown below:



Similarly, a *complex attribute* is an abstraction of attributes that can be complex or simple and is represented as:

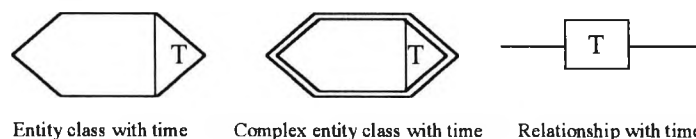


The collection of all attribute classes (simple or complex) that relate with an entity class is called the *entity state space*.

Hence, any system consists of entities, and each of them can be constructed at some initial point of time, have its state modified during its life, and in the end destructed when needed. In this way, the OO follows what is happening in real world where objects are born (constructed), live (modified) and die (destructed)

[Leo. & Karc., 1]. Every time an entity is constructed, the state space of the overall system is extended by the new entity state space and is reduced again when the entity is destructed. In this thesis, we are interested in developing a framework for dealing with systems, which entities can perform all actions (construction, modification, destruction) at particular points of time. In other words, we would like to keep track of their evolution, i.e., to keep their history and consequently be able to reason about it. The only points of time related to an entity are the time points of its construction and destruction and the time points of construction and destruction of extensions. The time evolution will be realised by creating and destructing extensions of one entity class using other entity class (multiple inheritance).

We can associate with each simple or complex entity class a time period, which represents the period of time during which an entity class exists. Relationships can also be “time-stamped”, i.e., they can be associated with a time period that represents the start and the end of the validity of the relationship. Hence, from now on, each time-varying object (whether entity or relationship) is assigned a time period. Whenever entities or relationships appear in the conceptual model without a time period, it is assumed that these have existed from the system start-up time. We represent these time-varying objects as shown below:



In their work, [Theod. *et al.*, 1] provide a similar concept of time, creating a framework that supports reasoning about calendar as well as symbol periods. A complex entity and a complex attribute should be alive only if the entities/attributes they consist of respectively are alive.

Normalisation is a step-by-step process for analysing data into its constituent entities and attributes. This process regroups attributes into a form having very useful properties and such that the relationships between entities are clearly shown. The normalisation guidelines can be regarded not only as a discipline by which the designer can capture a part of the semantics of the real world, but also as a means to recognise undesirable relations and to convert them into a more desirable form. For more details refer to [Open, 1].

5.3.3 Drawing the conceptual model for the consulting firm's problem

The first stage in the analysis of data is to list the obvious entity types. This will not normally reveal all the entities; new ones will emerge as the analysis proceeds. Attributes should be associated with each entity type but it is not always clear which entity an attribute really belongs to. Moreover, it may be difficult to decide whether a particular item of data should be classified as a separate entity or an attribute of some other entity. Normalisation is a technique which helps in just this way.

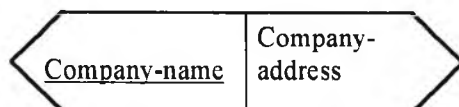
Below, we present the final form of the conceptual model for the specific problem. We have already applied the three phases of normalisation, which are:

- a) Remove all repeating groups from entities
- b) Ensure that all the non-identifying attributes for an entity are functionally dependent on the whole of the identifier for the entity
- c) Ensure that all non-identifying attributes (except candidate identifiers) are mutually independent.

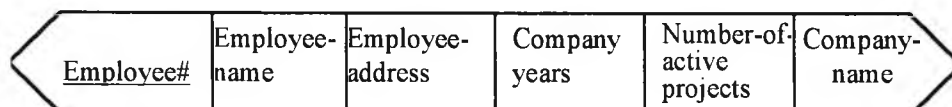
We have also avoided to represent the relationships between entity classes and attribute classes for the sake of keeping our model simple and easy to explain. Hence, we assume that employees consist of consultants, senior consultants and external consultants. Junior consultants are not considered employees until they pass successfully the employment process. Employees participate in the training/supervision activities. Finally, we also suppose that different departments in the consulting firm undertake different kind of projects.

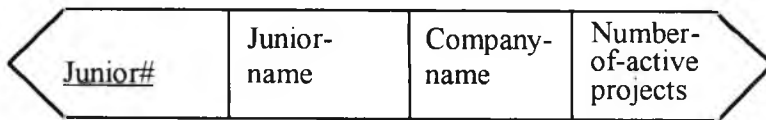
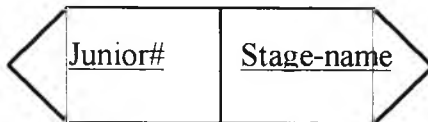
The entities for the conceptual model, as well as their attributes are identified below. Note that entities are in bold and the unique identifiers are underlined.

COMPANY

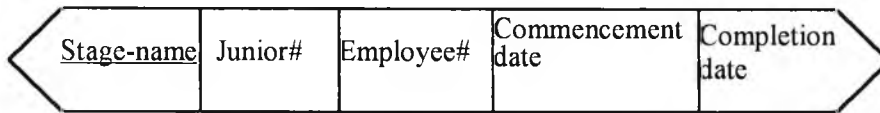
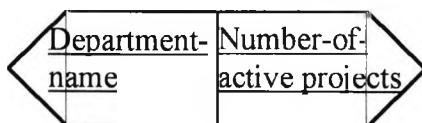
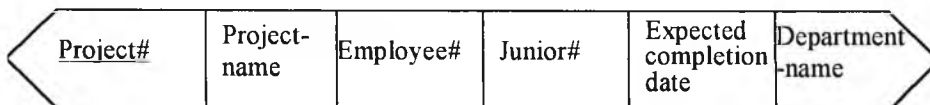
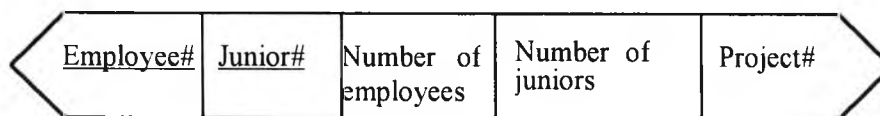


EMPLOYEE



JUNIOR**EMPLOYMENT PROCESS**


where by stage-name we mean training / supervision

STAGE**WORKLOAD****PROJECT****WORKFORCE**

In Figure 5.6, a global conceptual model for the problem has been constructed. This has been derived by combining all the different entities in the several local models and all the relationships between them. All repeated relationships and redundant attributes are then removed. Every relationship is given a name. Therefore, by looking at the figure, we see that the company (in our case, the consulting firm) *recruits* a number of junior consultants, and the recruitment policy depends on the demand for consultancy services. Junior consultants *pass through* an employment process that *involves* two stages. These are the training process and the supervision process. When

the employment process is finally completed, juniors qualify for employees. The firm's workforce consists of employees and juniors, who *undertake* projects; the number of which forms the firm's workload. As it was mentioned previously, different departments in the consulting firm undertake different kind of projects.

Figure 5.6 provides the reader with the functional dependencies of the different attributes (these are indicated by \rightarrow arrows). For example, there will be one name, one address and a specific number of years in the company for a particular employee number. Of course, nothing is implied about the arrow in the reverse direction. In the Workforce entity, the number of employees and the number of juniors are dependent on both Employee# and Junior#, and this can be shown by drawing the large hexagon, which shows that both Employee# and Junior# form the composite identifier. It is

important to appreciate the difference between the arrows.  is the relationship between entities, i.e., it relates one collection of attributes to another.

Note also that some of the attributes have been removed because of redundancy purposes. For example, Company-name as an attribute of the entities Employee and Junior is redundant since the information can be gained from the relationships *has* and *recruits* respectively. Finally, in the process of analysis some new entities have been identified. For example, initially a many-to-many relationship between the entities Employee and Project stood, i.e., an employee is involved in more than one project, and a project is undertaken by many employees. We managed to convert this relationship into two one-to-many relationships by introducing a new entity, called Workload. Workload requires many employees and it consists of many projects. In reverse, each employee deals with the workload and each project adds to the workload.

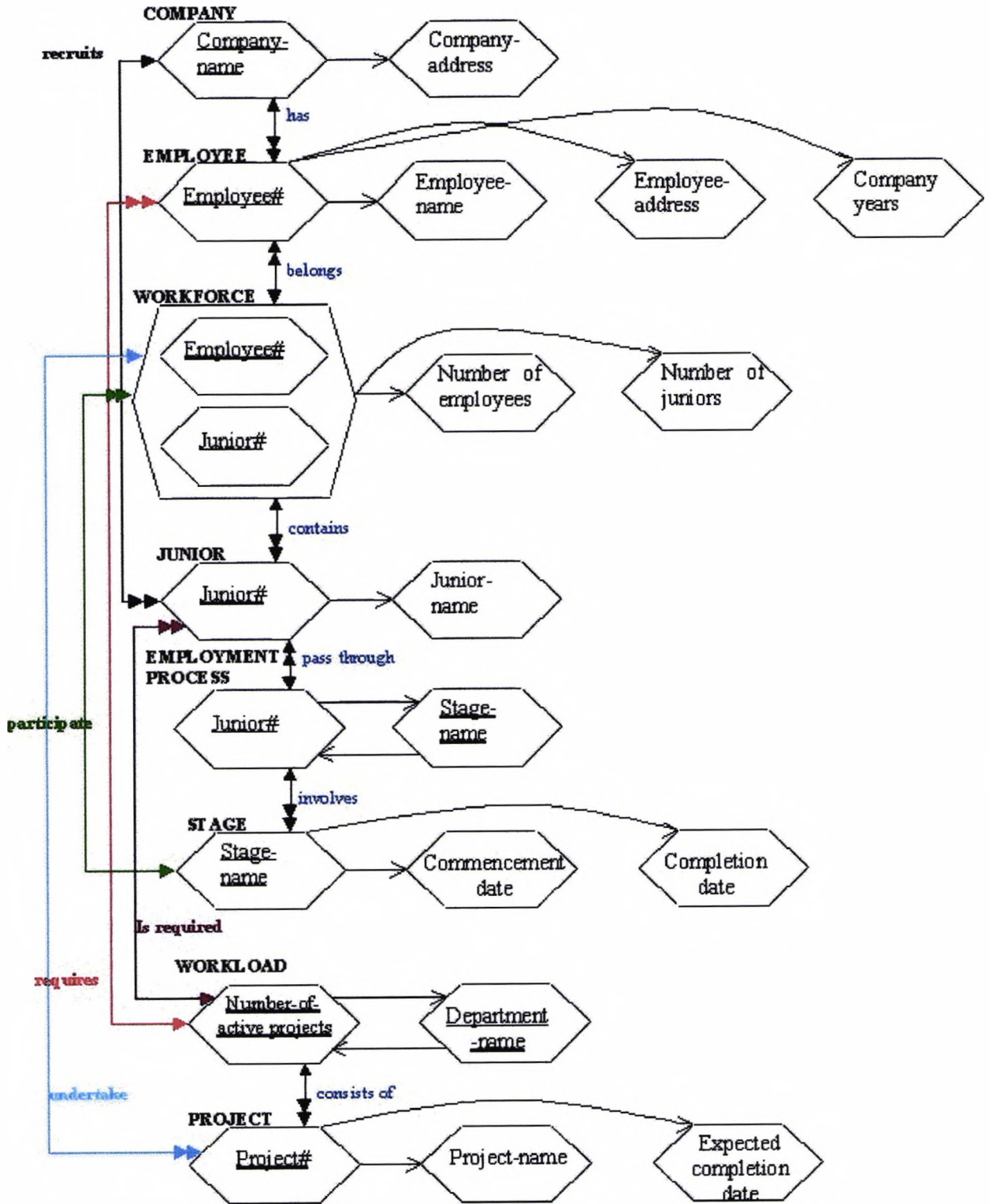


Figure (5.6): The global conceptual model for the consulting firm's problem

In case we wanted to represent the relationships between the various entity classes and the attribute classes, as well as the time-varying objects of the model, we

would end up with what is depicted in Figure 5.7. This figure represents only part of the model in Figure 5.6 because our model grows extremely and becomes rather complicated.

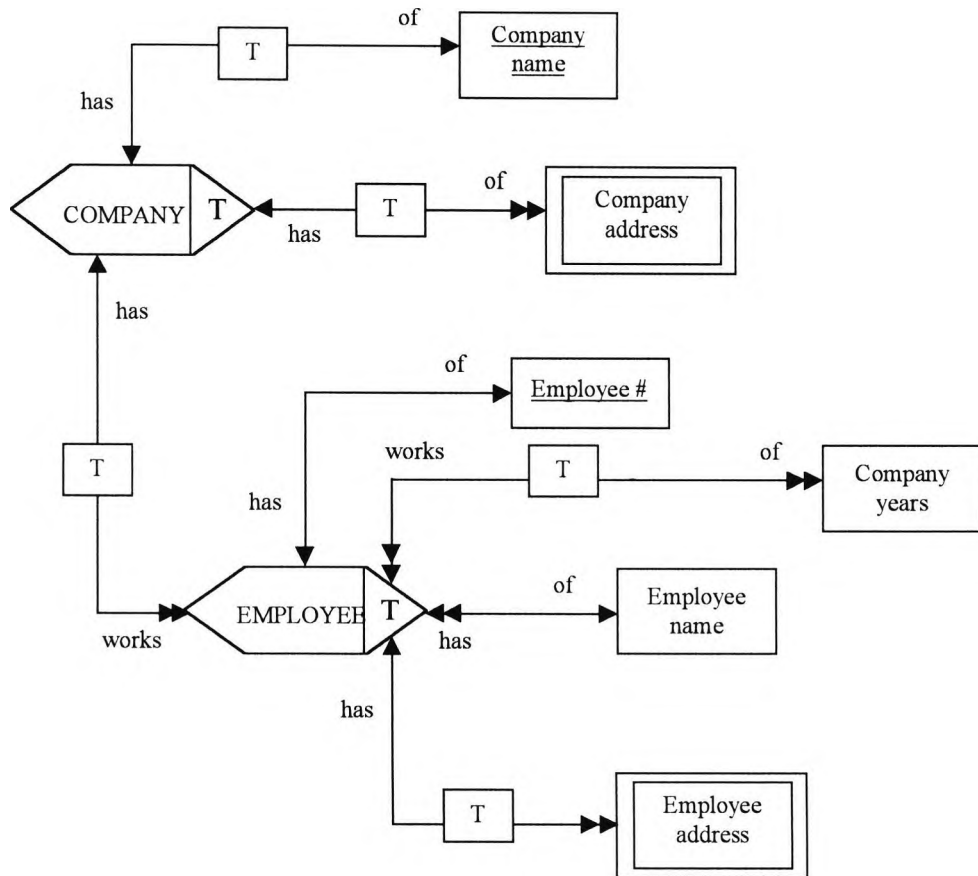


Figure (5.7): Part of the conceptual model of figure 5.6 that includes the time-varying objects

As one can see, attribute classes like Employee address are complex because they are formed through an aggregation of attribute classes such as Street name, Post Code and City name.

5.3.4 General comments on Conceptual Modelling

Conceptual modelling gives one the opportunity to shed the layers of processes covering up the fundamental essence of one's business. It offers an excellent means of communicating information between the designer and the client. Its main aim is to capture knowledge about the universe of discourse i.e., the modelled domain in the language of conceptual modelling, and reprint it in such a way so as to enable a system

developer to reason about this knowledge, communicate this understanding to end-users for validation, and specify the allowable structures and transitions on the information base [Theod. *et al.*, 1].

Two activities set conceptual modelling into motion. We first, try to understand the problem using descriptions expressed in natural language. The process the human mind follows subsequently in order to conceive the problem and to select a set of “components”, which will actually lead to an abstraction of the real system, is by far a very intriguing research issue. It is indeed remarkable how one can move from a verbal description of the problem to the second activity of conceptual modelling, that is the activity of abstraction; in other words to express the requirements of an application by defining the contents and structure of the appropriate components. The perception of what is considered as an entity and of the way entities relate with one another is a complex and confusing process. Conceptual modelling does not provide one with any guidelines for the selection of entities; elicitation of knowledge is based mainly on the goals and objectives of the modelling effort. Once again the modeller is faced with a blank piece of paper.

Conceptual Modelling has been used by the fields of Artificial Intelligence, Programming Languages, and Software Engineering, in a way that focuses mainly in the creation of semantic models that can suppress all the irrelevant details and emphasise only the essential ones. This is carried out by using abstraction mechanisms such as aggregation, grouping, classification, and generalisation. Nevertheless, even though these conceptual models have succeeded in economy of expression, in modelling flexibility and efficiency, their disadvantage is that they have emphasised so far on the static aspect of systems, i.e., on their structure without providing a modelling support for the behaviour of systems.

Behaviour refers to state transitions and dynamic properties. Additional research on semantic models has led to the extension of their properties so as to create behavioural models. These provide a conceptual formalism for expressing how and when changes occur to entities and relate with one another. An occurrence of a real-world external event triggers the execution of a number of processes, reflecting the effects of the event on the system and producing a state transition. By introducing the notion of the entity state space, it is easy to express the state change when events occur. There is a modelling approach in the conceptual modelling methodology, called *operational approach* [Rolland & Cauvet, 1], that has been strongly influenced by

database notions and incorporates the behavioural aspect. According to this approach, an event is deemed as a control mechanism of operation executions on entities. Naturally, this approach involves the specification of all permitted transitions in the conceptual schema. Techniques have been developed to reduce ambiguity, inconsistency and incompleteness in behaviour specification. Most of the behavioural models make use of tools for checking the specification. The tools use techniques deriving from finite state machines [Louc. & Zic., 1]. They consider the system specification as a hypothetical machine that generates a state change in response to an input. These techniques are supported by notations such as state transition diagrams and state transition matrices. Petri Nets are probably the most known technique of this class; they are particularly suitable to describe an intended system behaviour where precise process synchronisation is required.

Also, time is a very important subject in conceptual modelling because it is intimately tied to the description of the dynamic aspects of the world. Research interest in the time modelling area has increased dramatically over the past decade [Theod. & Louc., 1], [Theod. *et al.*, 1] and in this thesis we have dealt with this issue. The introduction of the time dimension in the modelling effort, permits one to keep historical information for the modelled domain, include a strong mechanism for expressing temporal requirements, and, finally, model the evolution of complex objects through time in a natural way.

In the consulting firm's problem, conceptual modelling has helped us to identify the main entities of the problem and the functional dependencies of their attributes. The conceptual model will therefore act as a backbone for the application of some more flexible and powerful methodologies, such as the System Dynamics and the Petri Nets. At the moment, we are not in position to infer anything about the recruitment policy the consulting firm should adopt.

5.3.5 Ontologies and Conceptual Modelling

So far, we have seen that the process of conceptual modelling involves the communication and the interaction of people and organisations that have different needs and diverse background contexts. As a result, each uses a different jargon, or may have differing or overlapping concepts and methods, leading to the existence of widely

varying viewpoints and assumptions regarding what is essentially the same subject matter. Also, difficulties arise in identifying the specifications for a particular system and users are frequently led to re-invent the wheel. Communities of people have started to recognise the importance of having a *shared understanding* that can function as a unifying framework for the different viewpoints, i.e., create a formal encoding of the important entities, attributes, processes and their inter-relationships in the domain of interest [Usch. & Grun., 1].

In some cases, the term ontology denotes the result of the activity of conceptual modelling (i.e., it is a conceptual schema), but our belief is that the construction of an ontology is by far a highly interdisciplinary approach [Guarino, 1], [Guarino, 2], where besides the basic tools of logic and computer science, an aptitude towards the subtle distinctions of philosophy and the challenging issues of natural language is necessary.

Definition (5.2): An *ontology* is the shared understanding of some domain of interest under which a *conceptualisation* is entailed. Conceptualisation is a world view of the particular domain; in other words, it is a set of concepts (e.g. entities, attributes), their definitions and their inter-relationships.

There are of course, different types of ontology. An ontology can be either an implicit conceptualisation (residing in someone's head) or an explicit representation of a conceptualisation, and that is where it overlaps with the representation phase of conceptual modelling. [Usch. & Grun., 1] identify four types of ontology depending on the degree of formality, but we will deal with how one can move from a highly informal ontology, i.e., expressed loosely in natural language, to a formal one, i.e., expressed in an artificial formally defined language. Ontologies assist the specification analysis of a system according to their degree of formality. In an informal approach, ontologies facilitate the process of identifying the requirements and the relationships among the components of the system, while in a formal approach, an ontology provides a declarative specification of the system.

[Usch. & Grun., 1] present two approaches of developing ontologies. The informal approach includes the following steps:

- identify purpose and scope;

- ontology capture, including the production of precise unambiguous definitions for the terms of the ontology and agreement on these definitions;
- ontology coding, including the identification of the representation language for the ontology;
- integrating existing ontologies;
- evaluation
- documentation

This more rigorous approach to the development of ontologies also provides a method for working with the different stages of the conceptual modelling process, and is of significant importance in this thesis. As in the ontology development, one is motivated by scenarios, i.e., problems encountered in enterprises or examples which are not adequately addressed. A motivating scenario provides a set of intuitively possible solutions to the scenario problems and these, in turn, provide an informal set of semantics for the objects and relations that will later be included in the ontology (or the conceptual schema). This coincides with the identification of goals and objectives issue in the conceptual modelling process. Soon after, a set of questions will arise expressing the requirements for the system. These are called informal competency questions, since they are not yet expressed in the formal language of ontology. The relationship between the informal competency questions and the motivating scenario, provides an informal justification for the new or extended ontology in terms of these questions. Note that people may have prior informal ontologies (explicit knowledge) when being in this stage. In that case, for every object, attribute, relation and axiom in the prior ontology, there should first be an informal competency question that requires the objects or constraints defined with the object [Ush. & Grun., 1].

A competency question is structured as seen in Figure 5.8 and it makes use of a set of assumptions, constraints and a set of sentences that are given in the statement of the question. A sentence is then formed to express the query, and particularly, investigate the rationale for the question and/or specify the decomposition of the question.

In this way, one manages to specify the requirements for an ontology. Through the informal competency questions, people can extract the set of terms that will form the informal terminology for the ontology. The first step in specifying the terminology of the ontology is to identify the objects in the domain of discourse. These will be

represented by constants and variables in the language. Attributes of objects and relations among objects are defined using predicates. A formal ontology will then be a formal description of objects, properties of objects, and relations among objects. [Usch. & Grun., 1] provide a very analytical description of the development of a formal ontology, but a further elaboration into the matter is beyond the scope of the thesis.

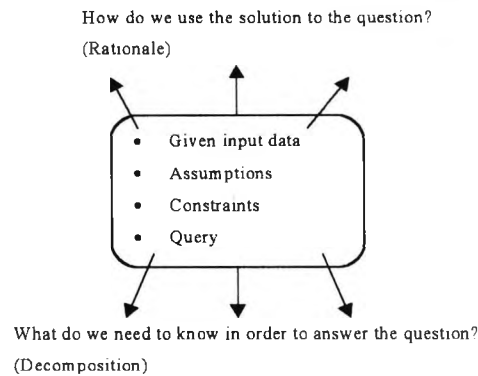


Figure (5.8): Structuring the competency question

The aim in this part of the chapter is to show the inter-relationship of ontologies with the process of conceptual modelling. We believe that at the development time, an ontology can play an important role in the requirement analysis and conceptual modelling phase, since it promotes communication between people and can speed up the modelling process through the re-usability of ontologies. In addition, the process of constructing ontologies assists the different stages of the conceptual modelling process, and points out the driving force behind both processes, which is, undoubtedly, the specification of goals and objectives (i.e., possible solutions to a motivating scenario).

5.4 Conclusion

Through this chapter, and especially through its use in the commercial sales modelling project, we have seen that the Soft Systems approach is a useful technique for modelling business processes, and particularly in identifying loosely specified processes as the ones we find in the analysis of soft systems. Emphasis is given on appreciating and improving problem situations rather than on “solving” problems. The approach provides significant help when ill-defined problems in a process occur. The models generated are clear, informative, and can be used as a focus for interviews in the later

stages of the project [Kingston, 3]. Unfortunately, throughout the construction of the model guidance is very limited, unless a process is identified as requiring knowledge-based reasoning, and in this case the CommonKADS library proves to be of help. The Soft Systems approach proves out to be the starting point for the development of the conceptual schema, because the first stage of the conceptual modelling process is based on it.

Conceptual modelling has an advantage over the natural language, or diagrammatic notations in that it is based on a formal notation, which allows one to “capture the semantics of the application”. It also has an advantage over the formal (or mathematical) notations because it supports structuring and inferential facilities that are useful for human communication. The process of conceptual modelling is further empowered by the use of ontologies, which support the communication between people with disparate background contexts, and assists re-usability, as well as identification of specifications for a particular problem. One could argue that conceptual modelling comprises the starting point and the basis of all those methodologies that offer a graphical description of the system. In the following chapters, one will discover how analysis can become more powerful by using concepts from System Dynamics and Petri Nets.

Chapter 6

SYSTEM DYNAMICS

6. SYSTEM DYNAMICS

6.1 Introduction

During the 1940's, formal analysis, often involving mathematical and statistical techniques, had been applied to problems as diverse as fighting of a war, and, the running of industries and business firms. System Dynamics was introduced at the MIT in the late 1950's by Professor Jay W. Forrester, whose work [For., 2] led to the development of a number of models of industrial problems. Realising that the mathematical techniques in Control Theory do not apply to managed systems because they tend to be much more complicated than engineering problems, Forrester used System Dynamics to describe a plethora of Soft Systems. The final stage in Forrester's intellectual work has been to define the structure of specialised computer simulation languages to enable the calculations in System Dynamics to be performed quickly and easily.

Since Forrester's pioneering steps, System Dynamics has been applied to an enormous range of real problems by defence analysis civil servants and consultants in several countries. Topics include combat modelling, manpower planning, equipment design, equipment support management and many others. Therefore, System Dynamics can be considered as the branch of management science, which deals with the dynamics and controllability of managed systems.

This chapter is structured as follows: First, a discussion on the basic concepts of System Dynamics and the influence of positive and negative feedback in the behaviour of the model is provided. Then, moving on to the structural approach of System Dynamics, the discussion on the influence diagrams, the simulation technique, the role of optimisation in System Dynamics and the issues raised when dealing with time is extended. Having presented the general methodology of System Dynamics, we come up with some criteria of quality that a System Dynamics model should satisfy and dwell on the assumptions made when using this methodology. Soon after a brief summary of the various tools that are based on the SD approach, the SD model for the consulting firm's problem is drawn. By using this model to test different policy scenarios, the power and the flexibility of the method unravel. The chapter concludes with the development of a discrete-time model for the consulting firm's problem, its

implementation through use of dynamic programming and a general discussion on the two approaches presented here.

6.2 Basic concepts of System Dynamics

Forrester has defined System Dynamics as: "...the investigation of the information-feedback characteristics of [managed] systems and the use of models for the design of improved organisational form and guiding policy." [For., 2]. But Wolstenholme has elaborated a bit more on the description of the methodology and quotes that it is a rigorous method for qualitative description, exploration and analysis of complex systems in terms of their processes, information, organisational boundaries and strategies; which facilitates quantitative simulation modelling and analysis for the design of system structure and behaviour [Wolst., 1]. And this is exactly what System Dynamics is about. It uses quantitative means to investigate the dynamic behaviour of socio-technical systems and their response to *policy*. Policy is the rationale that determines how a stream of decisions will be modulated in response to changing inputs of information [For., 1].

System Dynamics builds on the mental database and, where relevant, on written and numerical data. System Dynamics, through the use of simulation models, gives precise answers where the case-study approach is in the greatest difficulty - in the interpretation of a given set of assumptions into their dynamic implications. It has also forced into the open many modelling and analysis issues (philosophical and technical), that are common in fields using quantitative models to study systems having aggregate human behaviour as a component [Starr, 1]. These issues include establishing a model purpose, causality, selecting a means of validation, choosing a mathematical format, variable selection and aggregation, response tracing and policy formulation and analysis.

The methodology has a fundamental interest in affecting, if solving real problems, and a willingness to create, define or abandon causal theories of behaviour as necessary for model construction. The concepts encompassed in System Dynamics include [Starr, 1]:

- (a) A focus upon descriptive behaviour over continuous time, such as growth, decay, reversals, oscillation.

- (b) Identification of a system boundary such that the behaviour of interest will be generated internally from feedback loops containing time delays and non-linear elements.
- (c) A specified format for identifying and depicting feedback system elements, goals, discrepancies, levels, rates and flows.
- (d) A model-building attitude that attempts to include and quantify all factors that are believed to influence the behaviour of interest, whether or not such factors have a foundation in accepted theory or empirical data.
- (e) Identification of behaviour characteristics from the visual inspection of time response patterns generated by computer simulation.

A vital principle of System Dynamics is to incorporate all information believed to significantly influence behaviour into the model, leaving out unnecessary detail. System Dynamics is also problem-centred, or better goal-centred. The phase of problem definition is critically important.

6.3 Positive and negative feedback

In all managed systems, i.e., those upon which people can act [Coyle, 2], the presence of feedback loops is mandatory. There are two types of feedback loop: the *goal-seeking* or *negative* loops and *growth-producing* or *positive* loops. The differences between the two types of loop are quite crucial in understanding dynamic behaviour.

Central heating systems, aircraft autopilots and any other managed system, which seek to achieve a target, must include at least one *negative* or *goal-seeking* feedback loop. The essential idea of negative feedback is that, when there is a difference between the desired and actual states of the system, actions are generated, under the influence of the system's policies, so as to eliminate the difference. Ideally, if a discrepancy arose, it should be smoothly eliminated within a reasonable period of time. Nevertheless, there are cases in which the oscillations may get progressively larger, because the policies are not well designed in relation to the "physics" of the consequences in the system.

Positive feedback, on the other hand, acts as a growth-generating mechanism. For example, the state of the system, such as one's bank balance, grows continually larger as interest payment acts as the rate of change. Although, positive feedback is

quite common in managed systems, in an engineering system, positive feedback is undesirable and is designed out.

6.4 Methodology and Approaches

6.4.1 A structured approach to System Dynamics

Usually, when applying the System Dynamics approach, analysis is broken down to five stages [Coyle, 2]. The first stage is to recognise the problem and to find out which people care about it and why. Usually, the problem is re-examined since very rarely right answers are produced at this stage. In the second stage, the description of the system by means of an influence diagram takes place. The latter is sometimes referred to as a “causal loop diagram” indicating the interplay of the influences, which is the cause of the system’s dynamic behaviour. These first two stages comprise a more formal manipulation of the problem than the one offered in the Soft Systems approach. Positive and negative signs are assigned to the relationships between entities so as to indicate the type of interaction.

Stage three introduces “qualitative analysis”. The term simply means looking thoroughly at the influence diagram in the hope of understanding the problem better. This is, undoubtedly, a most important stage where one often does not need to continue to further analysis (this is the meaning of the dotted line between stages 3 and 4, as it is seen in Figure 6.1). In this stage, the analyst draws on so-called bright ideas and pet theories. The former arise from experience with other problems, where as the latter are the views of experienced people in the system so as to what is wrong with it.

If qualitative analysis does not produce enough insight to the solvability of the problem, work proceeds to stage four, which is the construction of a simulation model. In effect, the influence diagram and the simulation model are simply two versions of the same model; one written in arrows and words, the other in equations and computer code. This property is of fundamental importance in System Dynamics as it leads to some powerful practical consequences, which are, first, speed, and, secondly, ease of revision and expansion of a model. Since equations can be written quickly, a model can easily be revised as better understanding develops.

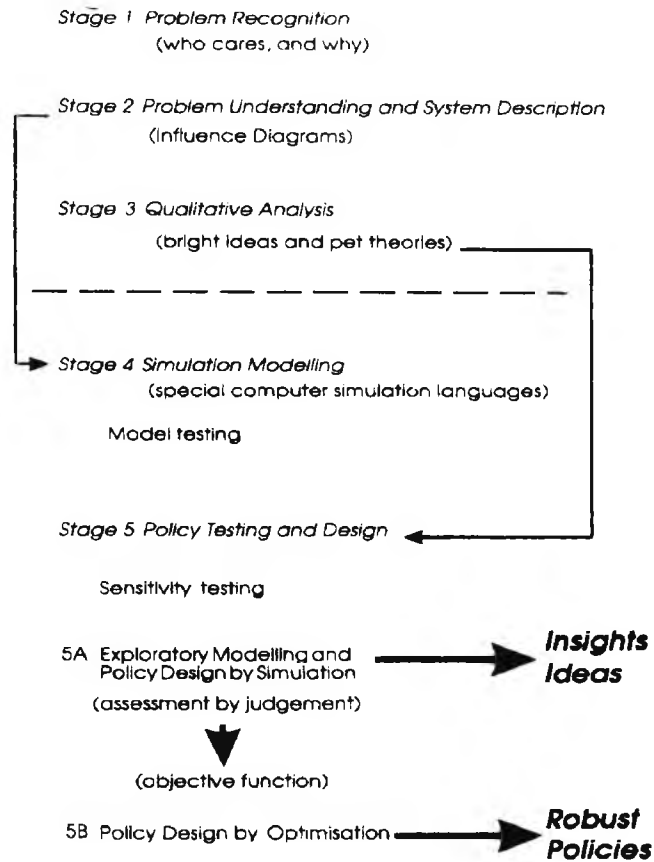


Figure (6.1): The five stages approach ([Coyle, 2]).

Similarly, it is easy to extend a model and that is why modellers usually exploit this property of System Dynamics by building small models to start with and allowing them to expand in a controlled fashion, rather than seeking to write a large model from the outset. In addition, the fact that the model exists in the two equivalent forms allows one to use whichever form is most conducive to effective thought and communication. This stage also includes the testing and the debugging of the model. It is necessary that the model is “valid”, meaning well suited to a purpose and soundly constructed.

Stage five is where results based on quantitative analysis start to emerge. In stage 5A, the emphasis is heavily on exploring the behaviour of the system rather than predicting precise details. The system is tested by simulating potential changes. Finally, it is possible to develop an extra set of equations (objective functions), which capture the essence of what the system is trying to achieve, or what one would like it to achieve, and use them to exploit the power of optimisation software in designing robust policies.

According to [Ander.& Richar., 1], System Dynamics rests squarely in the tradition of the scientific method. Grossly described, the paradigm for scientific

research comprises six or seven phases: *observation*, *hypothesis formation*, the *design* of an experiment to test the hypothesis, the *carrying out* of the experiment, *comparison* of experimental results with those predicted by the hypothesis, *refinement* of the hypothesis and a *return* to the beginning of the sequence.

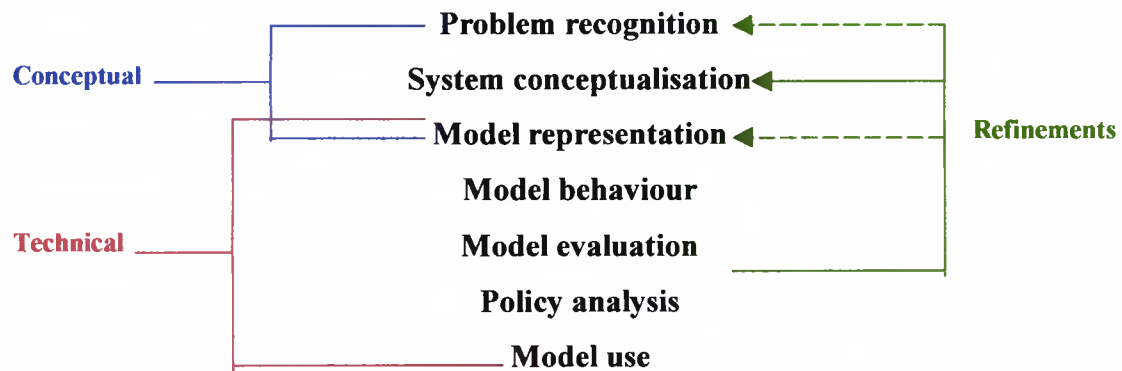


Figure (6.2): The inquiry procedure in System Dynamics

6.4.2 Influence Diagrams

The initial System Dynamics' representation of the problem is graphical. [Ander.& Richar., 1] call the patterns of these graphs as the '*reference modes*' of the problem, which help to more focused subsequent analysis. Influence diagrams also identify variables that must ultimately be in the completed model, moving the modelling effort at a workable level of aggregation. From this simple directive to 'draw a graph' comes an exceptionally powerful strategy, aiding the process of problem definition.

To verify this, the completed diagram for the central heating system will be studied, taken from [Coyle, 2]: The owner of the house has set a desired temperature on the thermostat, but the actual temperature may not be at that value. If it is not, the boiler is automatically turned on and heat starts flowing into the water pipes and radiators. After a delay due to the length of the pipes, heat starts to reach the room, adding to the quantity of heat already there. Depending on the size of the room, the heat quantity is sensed as temperature and when the temperature is close to the desired value, the supply of heat stops. We also suppose that this is an air-conditioning system, so if the room gets too warm, the system starts to subtract heat until the room cools down.

The simple trick in designing a good diagram is to "think physics" [Coyle, 2]. As seen, something flows and recognising what flows in the system is the key to a good

model (see Figure 6.3). Of course, there are rules and diagrammatic conventions that allow us to develop an influence diagram.

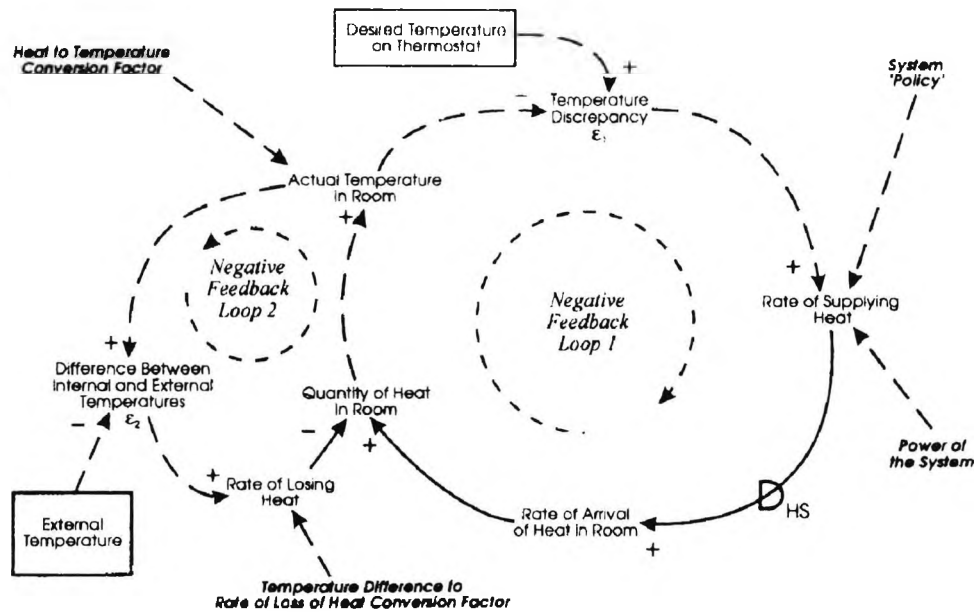


Figure (6.3): The influence diagram of the central heating system ([Coyle, 2]).

Influence diagrams are a vital tool in System Dynamics. There are three different methods for building these diagrams:

- The list extension method
- The entity/state/transition method, and
- The common modules method

All the above are thoroughly described in [Coyle, 2] and summarised in the Appendix.

6.4.3 Discussion on Simulation

Simulation models may be constructed for a variety of purposes [Stover, 1].

Among these purposes are:

- Testing a theory of system behaviour.
- Projecting the general behaviour of a system.
- Forecasting specific system parameters.
- Designing policies to affect system behaviour.

All the uses of simulation models have one characteristic in common - they deal with the future behaviour of the system. Information about past relationships is used to define the relationship in the model, which is then used to simulate the future. System Dynamics' models generally contain a relatively high degree of information about the future. Non-linear relationships, included in System Dynamics' models through table functions, often partly depend on historical data and partly on the modeller's perception of the future shape of the relationship. Future events are discrete, specific occurrences that have not necessarily occurred in the past but might occur in the future.

By using simulation, a set of equations is created in order to represent the system. This allows the equations to run forward in simulated time so as to attempt to mirror the behaviour of the real system as it runs forward in real time. The essential idea in time-step simulation is that the model takes a number of steps along the time axis. Each step is quite short, so that there might be many steps in total. The step length is always denoted by DT . There are also two types of variable. The first represents the *states* of the system, which in System Dynamics language are called *levels*. The second represents the physical flows in the system, which arise as a result of actions and are called *rates*. There is also a third type of variable used: the *auxiliary*. Rates are the movements across time steps, levels are the thresholds between time steps and auxiliaries are intermediate stages by which the levels determine the rates and they, therefore, also exist at the thresholds between time steps and are "now" variables (see Figure 6.4).

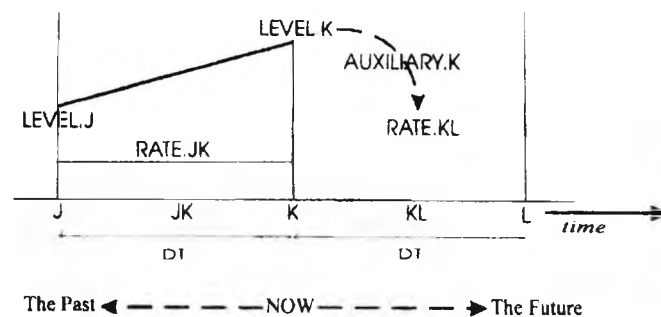


Figure (6.4): The diagram of relationship between levels, rates and auxiliaries ([Coyle, 2]).

The fundamental equations of System Dynamics are:

$$L \text{ Level.K} = \text{Level.J} + DT * \text{Rate.JK}$$

and

$$R \text{ Rate.KL} = f(\text{Level.K, Auxiliary.K, Parameters})$$

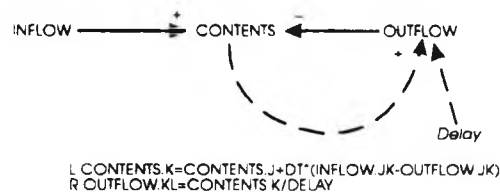
The first fundamental equation copes with the consequences of what happened during the JK time-step. The second is the information/action phase, which will take place at K.

6.4.3.1 Representing delays in models

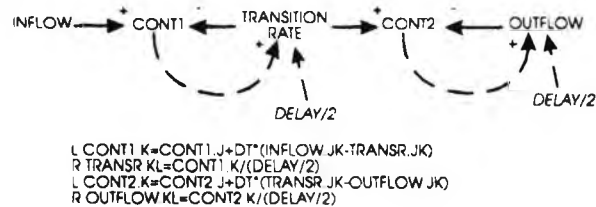
Delays are an extremely significant feature of managed systems. A method of representing them in quantitative models is developed; in influence diagrams we usually represent a delay by:

INFLOW-D → OUTFLOW

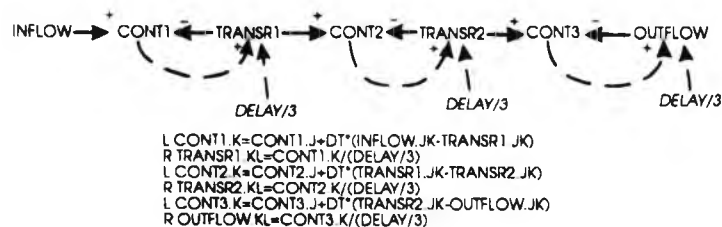
We add CONTENTS as a level variable and we have:



A) A First Order Delay



B) Two First Order Delays 'Cascaded' to Give a Second Order Delay



C) Three First Order Delays Give a Third Order Delay

Figure (6.5): The different types of delays ([Coyle, 2]).

The first type of delay is called *first-order* because only one level is used to store the contents. This delay can be quite a good model for certain processes [Coyle, 2]. For other problems, the first-order delay is too simple. For example, the training of scientists must take at least three years to complete a first degree, after which some will become productive quickly, while others will take longer. Such delays are modelled by “cascading” as many first-order delays as required to produce a higher order of delay. In System Dynamics, an infinite order delay is called a “*pipeline delay*”.

6.4.4 Optimisation in System Dynamics

It would be highly desirable to have some automated way of performing parameter variations up to a certain number and reporting to the analyst the best result found in the set of experiments. Therefore, one needs some sort of guided search of the parameters to be considered and the numerical value each might have, so as to seek out the result, which is most rewarding in terms of enhancing the system’s performance. Dynamic optimisation comes very close to providing this fine searching of the design possibilities of the system. An important aspect of dynamic optimisation is the development of measures of system performance, i.e., objective functions.

To develop an objective function one formulates equations to penalise failures to meet the target factors. The objective function should thus be understood of what it is: a mathematical artefact to guide the optimisation search, not a judge of what is good and what is not [Coyle, 2]. The ultimate choice of whether or not to accept and implement results derived from optimisation should be made by reviewing the output carefully. Optimisation might keep track of how improvements to behaviour can be found, but one has to be extra careful when choosing objective functions. Simply getting smooth dynamic behaviour is not always ideal if the smoothness has nothing to do with what the company is really trying to achieve.

It is very important to write a simple model to test the workings of a proposed objective function with very simple inputs. This is usually a very good strategy because the objective function is normally only formulated after the model has been developed, revised and experimented with.

6.4.4.1 Example:

This example describes a study of the fine paper division of a large integrated forest products company, made by [Risch, Troy. & Ster., 1]. The goals of the study were to help the company solve a thorny business problem and to demonstrate a strategic application of System Dynamics methodology.

“...GVI is a diversified forest products company with sales of several billion dollars per year. The company doubled its total assets in the last four years and more than doubled its sales. The company has several dozen plants and manufactures a broad variety of pulp and paper products. Over the years GVI’s strategy has been based on several themes. First, vertical integration maximises profit by allowing fuller utilisation of resources. Second, vertical integration must be accomplished from the market up. GVI purchases facilities close to the end-use market and then builds enough demand to justify a purchase one step upstream. Third, GVI has made part of this strategy to purchase existing capacity and overhaul it wherever possible. Entering the market through the commodity paper markets, the firm slowly built strong end-user demand that eventually allowed it to build several mills. However, in a move to diversify the business out of the commodity paper market, GVI recently moved away from its strategy when it purchased Owens Paper, a manufacturer of speciality fine papers. The purchase did, though, represent a continuation of the policy of buying existing facilities for upgrade. Almost immediately, GVI began a significant program of expansion and modernisation. GVI’s large capital outlay raises the issue of whether GVI will be able to generate an acceptable return on its investment, and how quickly. The issue is further compounded by the financial condition of the mill, which, at the start of the study was sustaining losses of several million dollars per month. Furthermore, since acquiring Owens, GVI has lost some share in the fine-paper market...”

The process was initially focused on understanding the fine-paper industry using firm level data, causal loop diagrams and policy structure diagrams. The authors drew on multiple data sources including observation, interview, statistical, and archival data. Internal interviews were conducted with personnel in operations, sales and marketing. External interviews addressed all major market segments and all stages of the down stream value chain.

The main question was what had been the motivation for Owens’ decision to enter the speciality paper market. There were two assumptions supporting the decision

to move into specialities. First, Owens believed it would be able to increase its average prices, without a corresponding increase in costs. The second assumption was that Owens would be able to penetrate speciality markets quickly and with a sales volume sufficient to run its machines close to full utilisation.

Figure 6.6 captures the causal relationships that are contained in the assumptions described previously. Balancing loop B1 shows management's "higher average prices" assumption: by moving into specialities (i.e. increasing the "breadth of the product line"), the average price received per unit of production and—given a sufficient sales volume—total revenues would increase, boosting profitability. By establishing a connection between profitability and breadth of product line, Owens created a balancing loop to restore profitability. Loop B2 acts in the same way: Owens' believed that as breadth rose, they would be able to sell more paper, boosting revenue and profitability. The third negative loop, B3, results from the recognition that total profitability increases with pulp consumption. As greater product line breadth raises total paper sales, pulp consumption also grows, boosting pulp mill profits and helping to restore profitability.

Reinforcing loop R1 captures the effect of greater product line breadth on unit variable costs. As breadth grows, unit variable costs grow (as a result of shorter run length, etc.), raising total costs and reducing profitability. If this loop were dominant over B1-B3, low profitability would cause still more expansion of line breadth and still higher costs. Balancing loop B4 reflects Owens' capacity limitations. Once the mill reaches capacity, slack to develop and produce new paper types disappears and expansion into new speciality segments comes to a halt.

Although, as [Risch, Troy. & Ster., 1] state, a causal loop diagram was used to represent the assumptions of Owens' management, the company's management was not explicitly aware of this feedback structure. The model suggests that Owens would increase the breadth of its product line by moving into more and more exotic types of speciality paper, until either profitability were restored or the mill reached capacity. They expected the balancing loops B1-B3 to dominate loop R1, so that profitability would soon rise as the company moved to specialities and expanded product line breadth. But while the results of the new strategy were promising at first, losses soon resumed.

Figure 6.7 shows what the authors believe is a better representation of the underlying system Owens faced as it tried to expand into the speciality paper market.

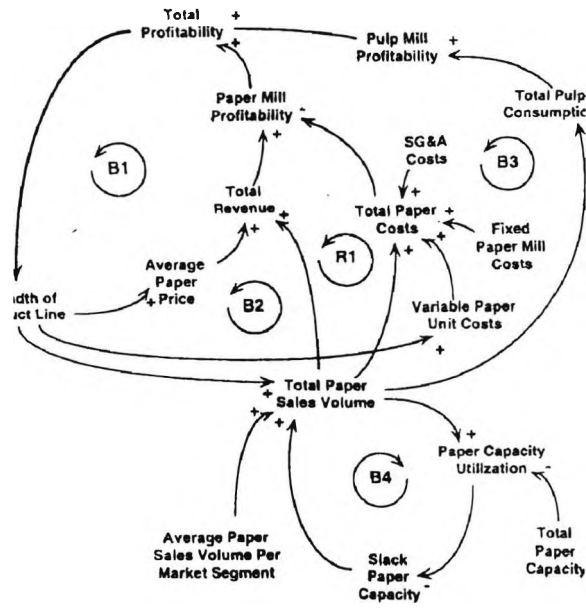


Figure (6.6): Influence diagram of the management's rationale for expansion into the speciality business (Risch, Troy. & Ster., 1)

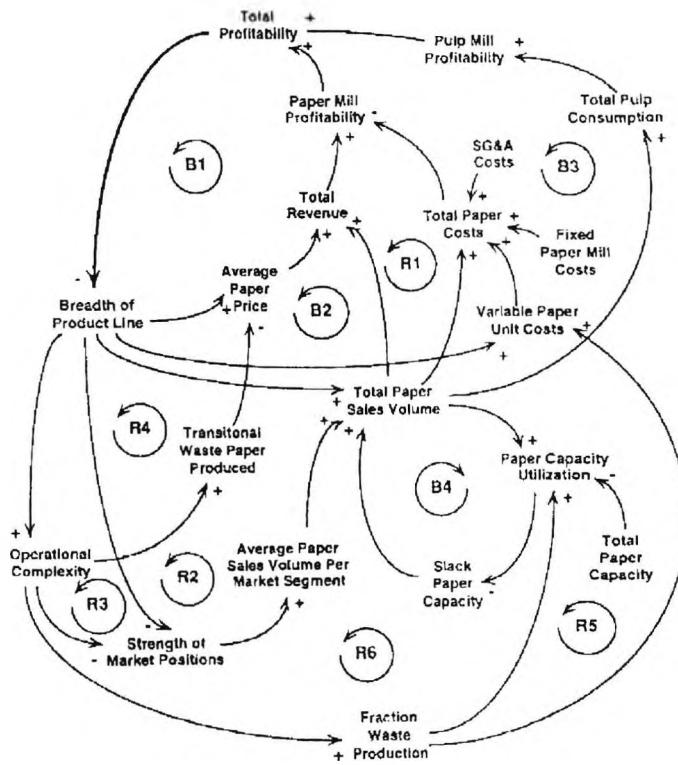


Figure (6.7): Revised loop diagram with the unanticipated side effects on mill operations of greater product line breadth (Risch, Troy. & Ster., 1)

The revised structure differs from the management's mental model in two ways. First, some of the causal connections in the actual structure are weaker, and others are stronger, than Owens' management believed. Second, it includes elements that were not contemplated in management's mental model.

6.4.5 Time in System Dynamics

System Dynamics is intensely affected by the ambiguity that prevails between scientists, when it comes to the conception of time. Certainly, some of the important controversies about System Dynamics are directly related to its ambiguous time concept. One of these issues, and perhaps the most important, is the matter of irreversibility. System Dynamics models at least conceptually can be run backwards to produce exactly the same trajectory in reverse. In simple terms, System Dynamics models change, but they do not evolve [Perel., 1].

System Dynamics models can be constructed so, that behavioural goals are a function of the system's past history. This tactic does not fully capture the stochastic quality of evolutionary processes, but still may offer a more authentic simulation of social system. Perhaps the main reason for the tendency to dismiss very long-term behaviour modes from models is because the resulting policies would make no difference in the short run.

6.5 Tests for building confidence in System Dynamics models

Confidence in System Dynamics models can be increased by a wide variety of tests that include tests of model structure, model behaviour, and a model's policy implications. There is no single test that serves to 'validate' a System Dynamics (SD) model. According to [For. & Senge, 1], confidence is the proper criterion to justify the validity of the system, because there can be no proof of the absolute correctness with which a model represents reality. There is no method for proving a model to be correct. A SD model is just tested against a diversity of empirical evidence, disproofs are sought and confidence is developed as the model withstands tests. Some of these tests are:

1. Structure-verification test: Verifying structure means comparing structure of a model directly with structure of the real system that the model represents. This test may include review of model assumptions, which might involve comparing model assumptions to descriptions of decision-making and organisational relationships found in relevant literature.
2. Parameter-verification test: Parameter verification means comparing model parameters to knowledge of the real system to determine if parameters correspond conceptually and numerically to real life. Structure verification and parameter verification are interrelated. Both tests spring from the same basic objective that SD models should strive to describe real decision-making processes.
3. Extreme-conditions test: If knowledge about extreme conditions is incorporated, the result is almost always an improved model in the normal operating region. This test is effective for two reasons. First, it is a powerful test for discovering flaws in model structure. Considering extreme conditions can also reveal omitted variables. The second reason for utilising the extreme-conditions test is to enhance usefulness of a model for analysing policies that may force a system to operate outside historical regions of behaviour.
4. Boundary-adequacy test: This test considers structural relationships necessary to satisfy a model's purpose. It involves developing a convincing hypothesis relating proposed model structure to a particular issue addressed by a model.

Some other tests are the statistical tests usually applied to social and economic models. There is also another family of tests; the family of model behaviour tests, which evaluate the adequacy of model structure through analysis of behaviour generated by the structure.

1. Behaviour-reproduction tests: They examine how well model-generated behaviour matches observed behaviour of the real system. Behaviour-reproduction tests include: symptom generation, frequency generation, relative phasing, multiple mode and behaviour characteristic.
2. Behaviour-prediction tests: They are analogous to behaviour-reproduction tests. Whereas, behaviour-reproduction tests focus on reproducing historical behaviour, behaviour-prediction tests focus on future behaviour. Behaviour-prediction tests include: pattern prediction and event prediction.
3. Behaviour-anomaly test: In constructing and analysing a SD model, one expects it to behave like the real system under study; but frequently the model-builder discovers

anomalous features of model behaviour, which sharply conflict with the behaviour of the real system. Once the behavioural anomaly is traced to the elements of model structure responsible for the behaviour, one often finds obvious flaws in model assumptions.

4. Family-member test: This test permits a repeat of the other tests of the model in the context of different special cases that fall within the general theory covered by the model.
5. Surprise-behaviour test: When unexpected behaviour appears, the model-builder must first understand causes of the unexpected behaviour within the model, then compare the behaviour and its causes to those of the real system. By running this test, the model-builder gains confidence in the model's usefulness.
6. Extreme-policy test: This test involves altering a policy statement (rate equation) in an extreme way and running the model to determine dynamic consequences. In this way, we discover the resilience of a model to major policy changes.
7. Boundary-adequacy (behaviour) test: This test considers whether or not a model includes the structure necessary to address the issues for which it is designed.
8. Behaviour-sensitivity test: This test focuses on sensitivity of model behaviour to changes in parameter values. The test is typically conducted by experimenting with different parameter values and analysing their impact on behaviour.

Finally, [For. & Senge, 1] present a number of policy implication tests. This attempt to verify that response of a real system to a policy change would correspond to the response predicted by a model. The tests also examine how robust policy implications are, when changes are made in boundaries or parameters.

1. System-improvement test: This test considers whether or not policies found beneficial after working with a model, when implemented, also improve real-system behaviour. Although it is a very important test, it presents many difficulties. First, it will not be tried until the model, from which the new policies come, enjoys enough confidence for the implementation experiment to be made. Second, the very long time for reaction in most social systems means that results of the system-improvement test accumulate slowly. This test becomes the decisive test in time.
2. Changed-behaviour-prediction test: This test asks if a model correctly predicts how behaviour of the system will change if a governing policy is changed. Usually, the test can be made by changing policies in a model and verifying plausibility of resulting behavioural changes.

3. Boundary-adequacy (policy) test: This test examines how modifying the model boundary would alter policy recommendations arrived at by using the model.
4. Policy-sensitivity test: Policy sensitivity testing, not only reveals the degree of robustness of model behaviour, but it also indicates the degree to which policy recommendations might be influenced by uncertainty in parameter values.

The accessibility of the whole testing process is crucial to possibilities for success in System Dynamics modelling. If fully exploited, the large variety of tests available to a multiplicity of evaluators should enable the development of useful models in which there is widely shared confidence.

6.6 Parameter estimation in System Dynamics modelling

System Dynamics modelling offers an attractive tool for policy evaluation. [Graham, 1] lays out the main techniques of an important step in System Dynamics modelling: *the estimation of parameters*. He tries to explicate the practical considerations involved in estimating parameters for System Dynamics models, he categorises estimation techniques and characterises the strengths and weaknesses of each category.

According to [Graham, 1], in SD models, data falls into two broad classes: data *below* the level of aggregation of model variables, and data *at* the level. The latter is straightforward (if “housing units” is a variable, a datum at the level of aggregation would be the actual number of housing units in a particular city at a given time). Data *below* the level of aggregation is information on the individual events and items, which might correspond to the model variables only after being aggregated together.

All factual knowledge about system-records, books, eyewitnesses, and personal experience, falls into the category of unaggregate data, which is by far the most abundant source of knowledge about real systems. The greatest single pitfall in using unaggregate data occurs when the modeller creates a model structure and parameters that are aggregated to the point where the processes characterised by the parameter values cannot be reliably observed [Graham, 1]. As a result, the parameters have little real-life meaning, and to estimate them, the estimator must draw conclusions based on the mental model of the behaviour of the system, rather than simply reporting observations.

There is a wide range of specific estimation techniques, from direct observation to ad hoc computations. Of particular interest is the problem of estimating extreme values and slopes, specifying the normal point, and drawing a smooth curve through the extreme and normal points. The main pitfall in estimation with unaggregate data is formulating an equation and its parameters in an aggregate, simplified manner, so that participants in the system cannot reliably observe a value of the parameter as a characteristic of the real system.

Estimation using a model equation: Estimation using a model equation starts with aggregate statistics that correspond to model variables. The pitfall, then, in estimating parameters with data at the level of aggregation of model variables is that the computations require two assumptions: accuracy of an equation, and appropriateness of the data. Such assumptions always make things more complicated.

Estimation using multiple equations: Estimation using multiple equations consists of manipulating several equations to compute a parameter value. The general pitfall of multiple-equation estimations is the same as for single-equation estimations: the techniques assume the accuracy of the equation(s) and data. The implicit assumption that seems to trouble multiple-equation estimation is that the discrepancy between real behaviour and model behaviour can in fact be attributed to the values of the parameters being estimated. The problem is that the unrealistic behaviour of the model may have been due not to the inappropriate parameter value, but to an unrealistic formulation or to some other parameter value being awry. Another pitfall of these methods arises from their use of data at the level of aggregation of model variables.

Therefore, to sum things up, estimation using data below the level of aggregation of model variables relies on observations of individual items or events that are represented in the aggregate by model variables. The principal pitfall of this technique is structuring the model on a level of aggregation too high to allow observers within the system to reliably translate their experiences into parameter values. The other two categories share pitfalls. First, use of data, at the level of aggregation of model variables, diminishes the ability to validate. Second, these techniques are vulnerable to systematic errors when assumptions are violated. Multiple-equation methods theoretically deliver greater accuracy but invoke more assumptions and hence, are more sensitive to minor violations of assumptions than single-equation methods.

6.7 Assumptions

The essential aim of System Dynamics is to achieve in socio-economic systems the standards of controllability and dynamic behaviour, which are commonplace in engineering systems. But in order to achieve this aim, a number of assumptions should be taken into account so as to guarantee confidence in results. Since any model is a simplification of reality, one needs to ensure that the significant parts of the real world are in the model; conceptual modelling can play a significant part in this phase of model development. The key step is to make some assumptions that are sensible for the problem, subsequently making sure that those assumptions are accurately maintained. For example, in most cases, it is assumed, as a rule of thumb, that the length for a model should be about two or three times the longest delay [Coyle, 2].

In formulating models, system dynamicists readily admit relationships that cannot be directly observed in available data [Starr, 1]. That is, they consider the effect of one variable on another when all other variables affecting the second variable are constant. Since all other variables are rarely, if ever, constant in reality, the hypothesised causal relationship may not match the observed relationship between the two variables. In this sense, the relationship may be unobservable. Furthermore, the technique of considering extreme conditions directly increases points of contact between models and reality. Moreover, the practice directly contradicts the common statistical modelling motto that one should never push model relationships beyond the database for which they were derived.

Last but not least, the omission of exogenous time-series variables guarantees that model behaviour arises from feedback loops - that all interactions necessary for understanding the causes of behaviour are captured within the model structure. Omitting these variables enables simulation testing to determine if model mechanisms provide a causal explanation of observed behaviour. If the model is able to generate observed behaviour, the absence of exogenous variables guards against the possibility that the model matches reality simply because of its exogenous inputs rather than its internal structure.

6.8 Tools

Some of the software tools for system thinking and dynamic modelling are:

1. COSMIC and COSMOS are an integrated environment for the development, analysis and optimisation of system dynamics models. The software suite is based on a Workbench concept and can be run on any PC. It can run under DOS or WINDOWS. Model building is supported by a graphical development environment in which the user builds influence diagrams on the screen, creating and defining new variables as required. Level equations are written automatically; rates and auxiliaries are created on the screen by the modeller using a mouse. COSMOS supports four methods for optimisation.
2. DYNAMO is a mainframe or IBM-PC based simulation language. It is widely used in logistics, project management, and other corporate applications. The tool allows the user to create large models (via a text editor) using arrays, macros and functions (either built-in, such as TABLE, or definable in C). Flowcharts or other diagrams can be imported for display.
3. DYSMAP2 accepts models from a source file. Equations may be written in any order and variable names may be up to 32 characters. The software supports dimensional analysis and includes the usual logical, time-related and delay functions. Up to 10 runs can be performed in any one session and new runs can be launched interactively from the ICE (Interactive Command Environment).
4. EISToolKit integrates a graphical user-configurable interface with simulation models, databases, and spreadsheets. It is available for both Macintosh and IBM machines. A number of petrochemical and pharmaceuticals firms are among those who use the tool.
5. The iThink software comes into two versions: Core and Authoring. The Core version is designed to enable people to construct system dynamics models by piecing together stocks, flows, convectors, and connectors on an electronic tableau. The package supports multi-variable sensitivity analysis and run-by-sector capability. The Authoring version of the software enables people to create run-time accessible, stand-alone Learning Environments, or Management Flight Simulators. The STELLA and iThink are Macintosh-based. Graphical user interface facilitates

- interactive model development. They are widely used for operational and strategic modelling.
6. The Powersim software is a WINDOWS-based graphical system dynamics modelling package. It supports both flow diagrams and causal loop diagrams. The underlying equations can be entered when the model's structure is defined. Equations are created with the aid of a library of built-in functions and a visual editor for drawing graph functions. Simulation results are presented as animation, numbers or graphs. It supports the creation of games and "management flight simulators", as well as multidimensional arrays.
 7. S⁴ is a powerful simulation interface best used for management flight simulators. S⁴ contains many unique features and can read models developed in DYNAMO or STELLA.
 8. Last but not least, Vensim is a powerful simulation environment with graphical interface and innovative capabilities for statistical estimation and calibration of models with historical data. It supports causal loop diagrams and stock and flow diagrams. It provides *Document*, *Loops*, *Outline* and *Tree* tools to generate different representations of model structure and includes a dimensional analysis facility. The software supports subscripts, special variable types and user-defined macros. An interface can be used to create special purpose functions. It is used in a variety of corporate and defence-industry applications, as well as large-scale macroeconomic modelling at MIT.

6.9 Drawing the SD model for the consulting firm's problem

In this part of the chapter, we focus on the study of the Human Resources Management System of a consultancy firm. This system captures the hiring, assimilation, transfer and attrition of the firm's workforce. Such actions are not carried out in a vacuum but are driven by the company's estimated turnover, which is an indication of the demand for services. The research vehicle for the study of the dynamics of this system is a comprehensive *system dynamics* model. A detailed discussion of the model's structure as well as of its behaviour is provided.

In particular, the model can be used to test the interchangeability of personnel (junior consultants, consultants, senior consultants, and external consultants) when the

demand for consulting services varies. The model is also capable of providing a decision-support to answer what-if questions and this mechanism is also illustrated. Ithink[®] modelling program is used here to demonstrate the causal relationships of the variables within the system, as well as to simulate different policy scenarios.

There are four basic icons in the Ithink[®] language; the two primary icons are the *stock* and the *flow*. The former, symbolised by a rectangle, is used to represent accumulations. Accumulations occur in all functional areas within the organisation. In Human Resources, for example, one finds build-ups of people, policies and procedures, as well as frustration and commitment. As for flows, symbolised by pipes, Human Resource flows include hiring, quits and training. Both stocks and flows are necessary for generating change over time, or dynamics. Converters are another icon in the Ithink language. These are represented by circles and convert inputs into outputs. Unlike stocks, converters do not accumulate. The final building block is the connector. Connectors link stocks to converters, stocks to flow regulators, flow regulators to flow regulators, stocks to flow regulators.

After having parsed the narrative, i.e., the description of the problem as stated in Chapter 2, we detect the main entities, which are Juniors, Consultants, Senior Consultants and Externals, as well as the physical flows between these four entities. All these are included in the diagram (see Figure 6.8). At this instance, there is only one policy area by which the system can be managed, the recruitment of juniors. As the model stands, recruitment is governed only by the annual turnover of the firm and the size of the workforce, including juniors and supervisors. When juniors are recruited, qualified people (consultants and senior consultants) are immediately assigned to train and supervise them, which has an effect on the availability of qualified consultants not acting as supervisors.

The Equivalent Consultant Workforce is shown as being influenced by six variables and one parameter (Figure 6.9). The productivity parameter is necessary because the firm want to calculate workforce in “consultant equivalent” units. However, in Figure 6.8 there is no indication of how these seven influences determine the variable. It is, however, evident in the Equations Level of the Ithink software that the relationship below holds:

```
equivalent_consultant_workforce=External_Consultants+con_available_for_projects+senior_con_available_for_projects+productivity_of_jun_&_sup*(Junior_Con+con_supervising_junior+senior_con_supervising_junior)
```

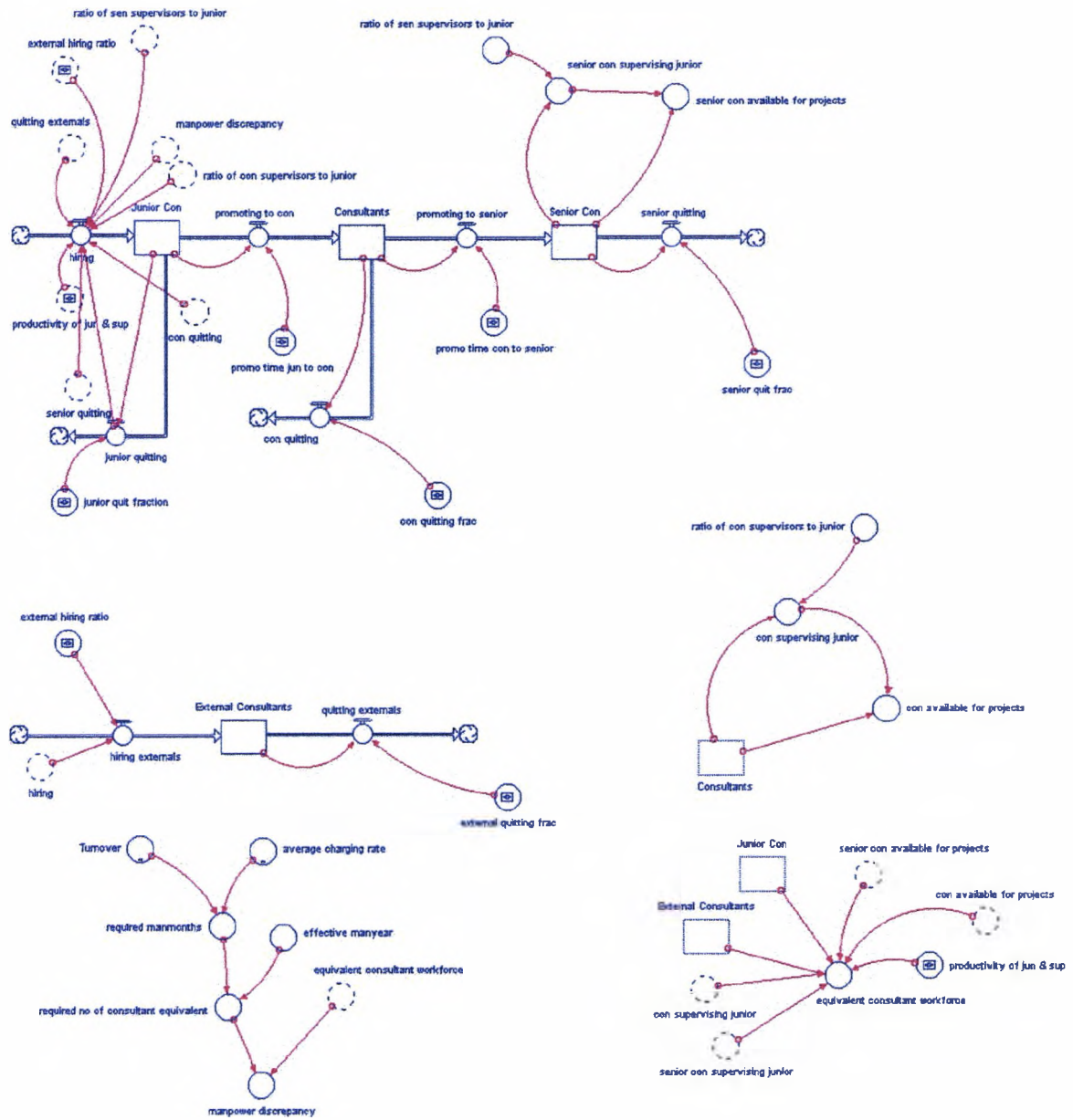



Figure (6.8): The overall system dynamics model for the consulting firm's problem

A detailed account with all the mathematical equations, as well as, all the initial values used for the parameters in our model, is provided in the Appendix. In the manipulation of this model we tried to make use of data of generic nature, as well as, experience, gained through various case studies appearing in literature (an example would be [Abdel-Hamid, 1]'s work). Annual figures and predictions have been provided by the Greek company of consultants PLANET S.A. The process of gaining information is of course an iterative process. Interviews and discussions with Planet's personnel have and continue to help in better sketching the nature of the problem.

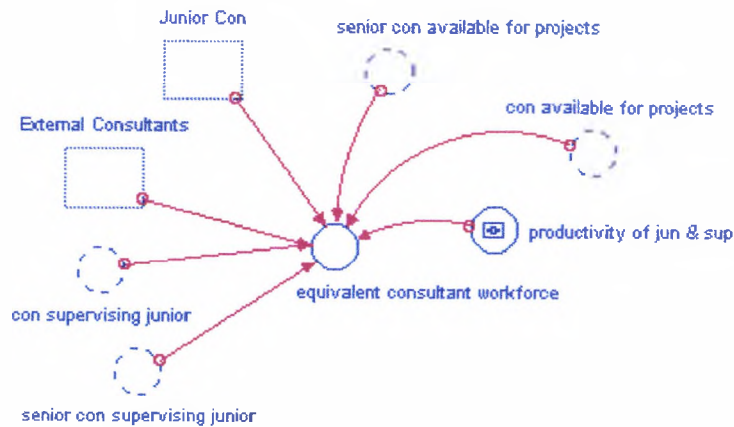


Figure (6.9): The SD model for the equivalent consultant workforce

After interviewing some senior consultants from PLANET, it was decided that the demand for consultancy services would be incorporated in the modelling of the problem by means of turnover, instead of counting the projects the company undertakes annually. PLANET aim to be able to raise the average charging rate every year, and that is why, in the function definition of the average charging rate, there is a rise of 2% per annum. Turnover is then easily translated to manmonths through the relation:

$$\text{required_manmonths} = \text{Turnover} * 1000 / \text{average_charging_rate}$$

and the latter indicate the Required number for Consultant Equivalent. The discrepancy between that and the actual number of employees the company has (Equivalent Consultant Workforce) constitutes a component of the desired recruitment rate.

Hiring takes place in two levels of consultants: the Junior and the External Consultants. There is a company standard practice that the ratio of externals over the internal consultants that the firm has, remains constant and does not exceed a certain value. This ratio is called here the External Hiring Ratio and governs the hiring of the personnel at the two different levels. The hiring of Juniors is determined by the manpower discrepancy, as well as, by the total annual quitting of the personnel at the various levels (Figure 6.10), always calculated in consultant equivalent.

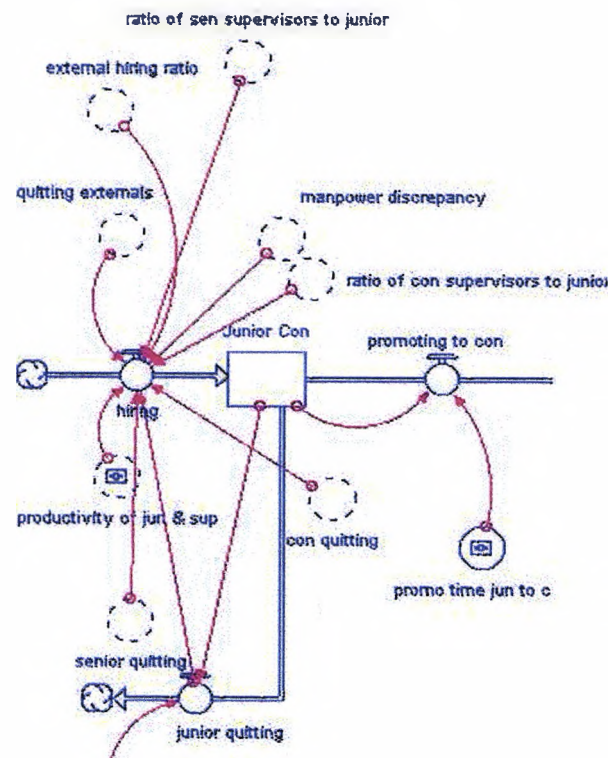


Figure (6.10): The SD model for the hiring of Junior Consultants

The equation describing the hiring of Junior Consultants is provided below:

hiring=

$$\begin{aligned} & \text{manpower_discrepancy}/(\text{productivity_of_jun \& sup}+\text{external_hiring_ratio})+\text{productivity_of_jun \& sup}*\text{junior_quitting} \\ & +\text{con_quitting}*((1-\text{ratio_of_con_supervisors_to_junior}) \\ & +\text{ratio_of_con_supervisors_to_junior}*\text{productivity_of_jun \& sup}) \\ & +\text{senior_quitting}*((1-\text{ratio_of_sen_supervisors_to_junior})+ \\ & \text{ratio_of_sen_supervisors_to_junior}*\text{productivity_of_jun \& sup})+\text{quitting_externals} \end{aligned}$$

where as one can see, quitting at the various levels is translated to consultant equivalent (due to the multiplication with the productivity parameter), and manpower discrepancy is multiplied by the $1/(\text{productivity} + \text{external_hiring_ratio})$ factor to reflect the truly effective consultants that are required. The hiring of the External Consultants is based on the hiring of the Juniors and is actually a fraction of it, i.e., $\text{hiring_externals} = \text{external_hiring_ratio}*\text{hiring}$. After being recruited, Juniors undergo a period of training and supervision. The determination of the amount of effort to commit to the training of new employees is typically based on organisational custom. According to [Abdel-Hamid, 1], there are no proposed formulas in the literature, or in any of the organisations he had studied. There are, however, rules-of-thumb and these ranged from committing 15 percent of an experienced employee's time (per new employee) to a 25 percent

commitment. In our model, we have simplified the situation by having ratios of qualified consultants over juniors, who act as supervisors (Figure 6.11). The same formula applies to Senior Consultants who also act as supervisors.

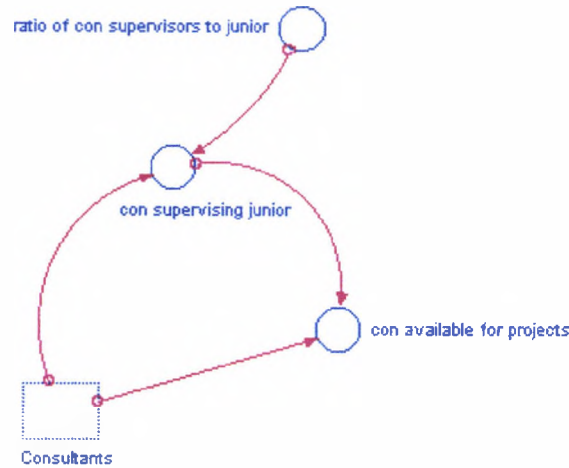


Figure (6.11): Distribution of Consultants to supervisors and to those available for projects

6.9.1 Simulating the model

In the Interface Level of the Ithink software, the user can use simulation to test and evaluate different policy scenarios that the firm might implement in controlling recruitment. The interactive model simulator, hence, provides a richer understanding of how the system is more likely to perform in response to one's decisions, without the risk and cost of running real consulting practices into the ground. The Slider Input Device allows model users to adjust constant values, and to override equation logic (and graphical function relationships) with numerical inputs (Figure 6.12).



Figure (6.12): A slider for the promo time jun to con

Taking the promo-time from junior consultants to consultants to be 5 years and promo-time from consultants to senior to be 7 years, junior quit fraction 0.40, consultants quit fraction 0.20 and senior quit fraction to be 0.20, as productivity of junior & supervisors equal to 80%, and external consultants hiring ratio as 30%, we find that the number of

junior consultants over the 7 years that the simulation is running, rises significantly (blue line) and so does the number of externals (green line) (see Figure 6.13).

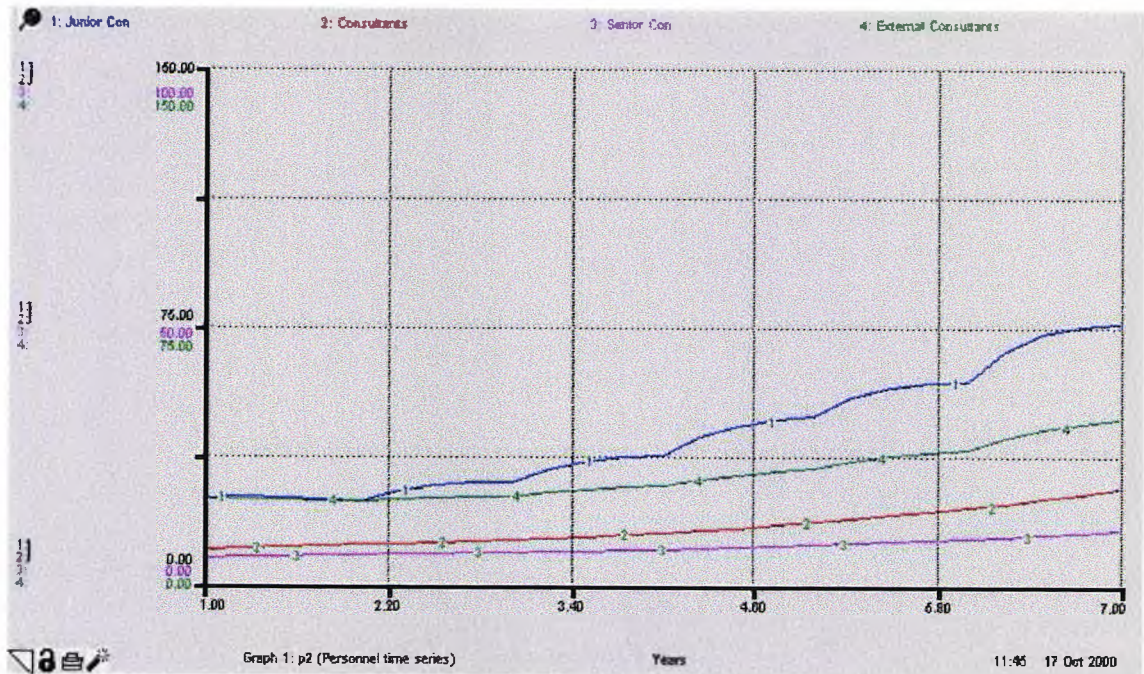


Figure (6.13): A first attempt to run the model

To get a better view of the behaviour of our model, I think software provides us with a bar graph (see Figure 6.14), as well as with a detailed table of the different values of the variables of interest (Table 6.1).

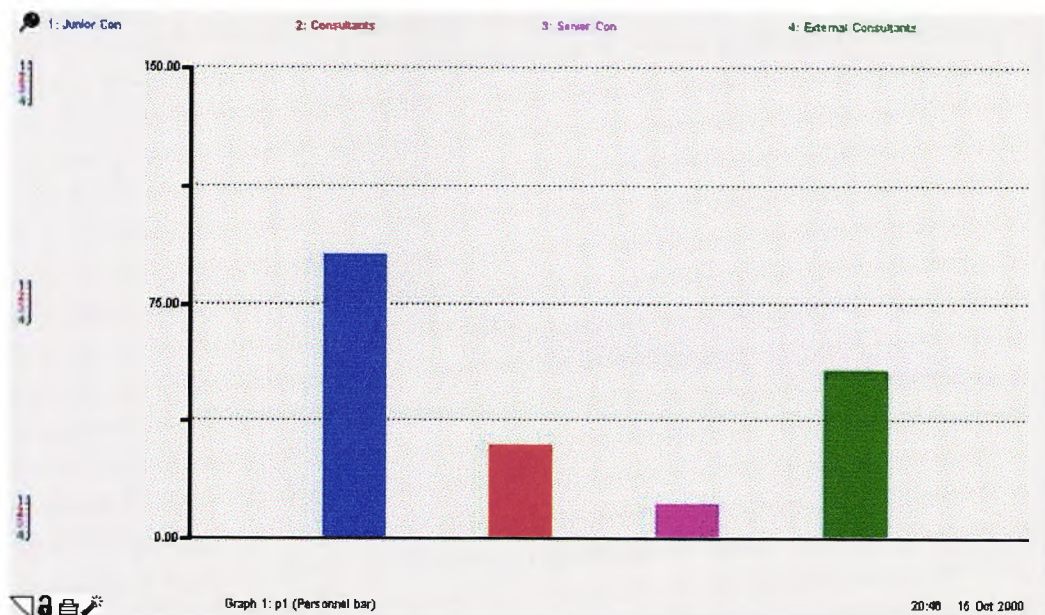


Figure (6.14): The personnel bar graph

Table 6.1 provides the reader with the summed ending balances of the variables of interest for seven consecutive years. Senior Consultants rise from 5 initially, to 10 after seven years, and this number is considered as an “expected” value by the company. Manpower discrepancy remains positive throughout the simulation, a fact that indicates that the increase in demand (increase in turnover) requires more and more people to be available for projects.

In this model, the proportion of people moving to the next higher level is a management control influenced by circumstances, rather than parameters. That, of course, suggests the idea that, while the consulting company may well not be prepared to make people redundant, there is no reason why the proportion staying after training/supervision or work at a particular level, should depend only on the preferences of those individuals. Perhaps management should have a policy say, in which case the proportion would cease to be a parameter and become a variable, influenced by the manpower discrepancy. For example, the company could consider the stability of the workforce. Thus, before adding new project members, management could try to contemplate the project employment time for the new members [Abdel-Hamid, 1].

| Years | Initial | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|--------------------------|---------|--------|--------|----------|----------|----------|----------|---|
| Junior Con | 25.0 | 25.0 | 24.8 | 44.0 | 57.0 | 67.8 | 90.3 | |
| Consultants | 10.0 | 11.6 | 12.6 | 15.4 | 18.9 | 23.6 | 28.2 | |
| Senior Con | 5.0 | 5.4 | 6.0 | 6.8 | 7.6 | 8.7 | 10.4 | |
| External Consultants | 24.0 | 24.4 | 26.9 | 30.5 | 36.2 | 42.5 | 62.9 | |
| con available for projec | 8.5 | 9.8 | 10.9 | 13.1 | 16.1 | 20.2 | 24.8 | |
| senior con available for | 4.8 | 5.2 | 5.7 | 6.3 | 7.1 | 8.3 | 9.8 | |
| hiring | | 18.8 | 24.4 | 30.0 | 36.4 | 46.0 | 63.1 | |
| Junior quitting | | 10.0 | 10.4 | 13.0 | 17.0 | 22.8 | 27.1 | |
| promoting to con | | 5.0 | 5.2 | 7.0 | 8.8 | 11.4 | 13.8 | |
| Turnover | 1.50 | 1.06 | 1.48 | 3.16 | 3.80 | 5.18 | 6.90 | |
| average workforce rate | 2.30 | 2.46 | 2.80 | 2.72 | 2.90 | 2.84 | 3.15 | |
| required manmonths | 852.17 | 776.51 | 923.09 | 1,139.71 | 1,357.14 | 1,734.80 | 2,190.48 | |
| required no of consulta | 69.3 | 70.5 | 83.8 | 103.6 | 123.4 | 157.7 | 199.1 | |
| equivalent consultant u | 59.3 | 61.8 | 73.1 | 87.2 | 107.6 | 128.4 | 163.7 | |
| manpower discrepancy | 0.0 | 8.7 | 10.8 | 18.5 | 15.8 | 29.3 | 35.4 | |

Table (6.1): Table of values for Figure 6.13

Since, after all, the whole study of the problem aims at identifying how well the company deals with the increasing turnover, i.e., the increasing demand for projects,

one needs to run different case studies so as evaluate the best possible scenario. Ithink software offers a very helpful tool called sensitivity analysis that allows the user to input a different value in the parameter of interest and then run the simulation as many times as the different values of the parameter. Through this analysis, we discovered that the best case scenario for our problem (Figure 6.15) would be the one that would guarantee a small manpower discrepancy, i.e., the company deals with the demand in the best possible way and this is where external quit fraction is 0.2, productivity of juniors & supervisors is 0.8, and the external hiring ratio is 0.4. If the HR manager changed the period of training from 5 to 3 years, manpower discrepancy would fall even more but the number of senior consultants would rise to 13, a number that supersedes the predictions and the previous experience of the company.

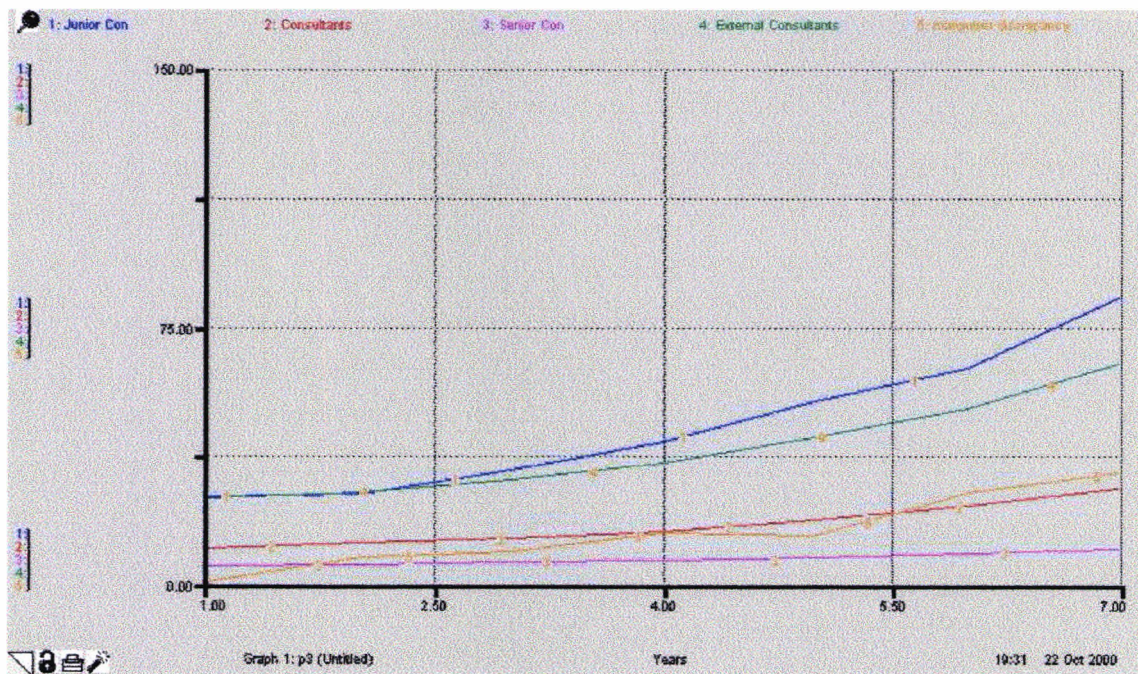


Figure (6.15): The best-case scenario

A closer to reality scenario (Figure 6.16) would mean that the junior quit fraction is decreased to 0.3, productivity is 0.7, external quit fraction is 0.1 and the external hiring ratio is 0.3. In this case, manpower discrepancy (orange line) rises a bit; the company seems to be in a bit of a difficulty to keep up with the demand. The number of senior consultants is well around 10.

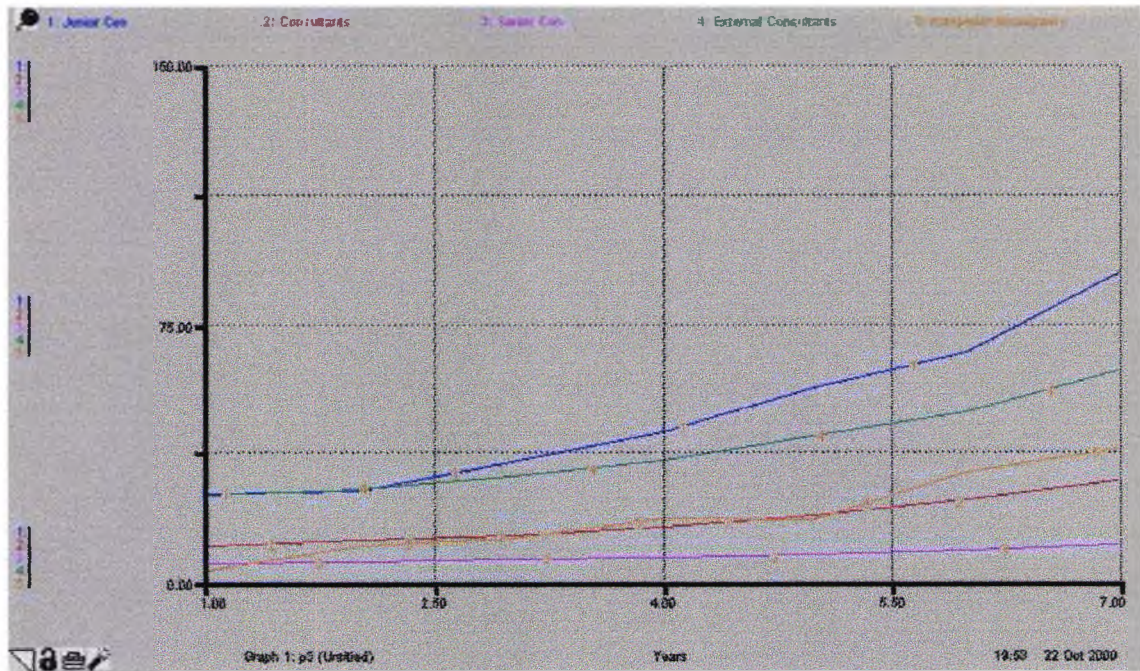


Figure (6.16): A closer to reality scenario

The role of manpower discrepancy is critical in the decision of the company to hire new personnel. Management could face three possible situations:

- a. First, the manpower discrepancy between the Required no of consultant equivalent and the Equivalent consultant workforce actually on hand, could be zero. That would result in a model that would give information of the necessary hiring rate that is required to simply compensate for the annual overall attrition of personnel. To verify the case, manpower discrepancy is set to zero, hence the following results are produced: In this case, hiring is very small (Figure 6.17 and table 6.2), typically, around 10, just to cover the overall quitting of consultants, and the model gives output, (i.e., equivalent consultant workforce) with small fluctuations, from year to year.
- b. A second, more common situation, is where the Required no of consultant equivalent is found to be larger than the current Equivalent consultant workforce. In this case, new staff will be added to the firm so as to deal with the demands of the projects. This, of course, takes time. While some recruits will generally be available within a short period of time from within the organisation, others, especially when the firm are seeking special skills, will not be available for a much longer term.

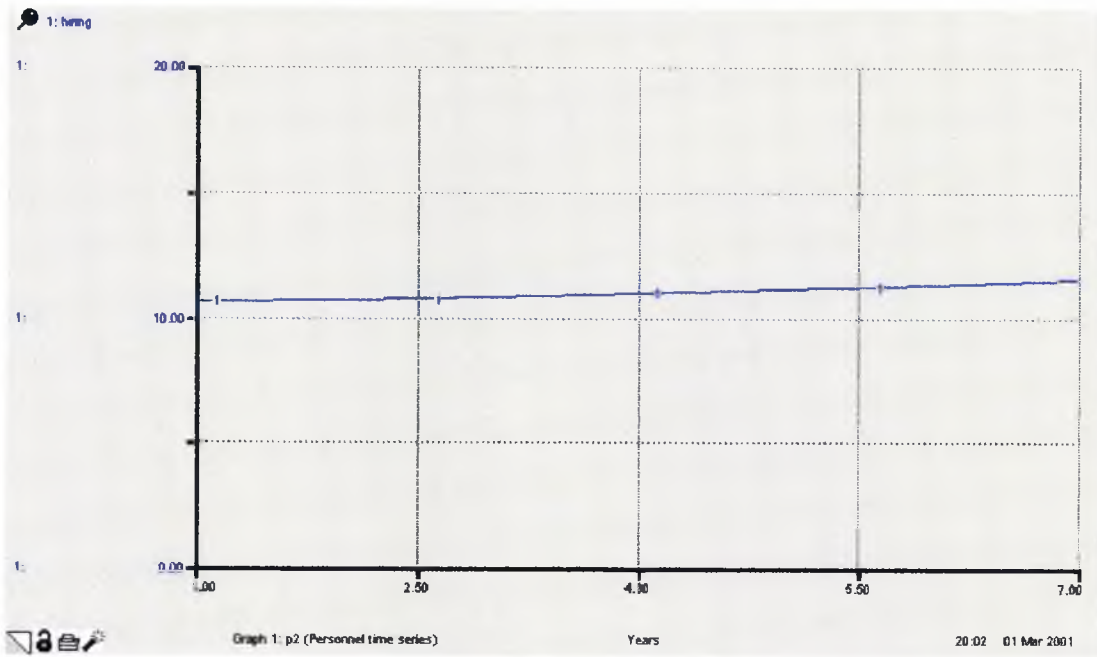


Figure (6.17): The graph of hiring when manpower discrepancy equals zero

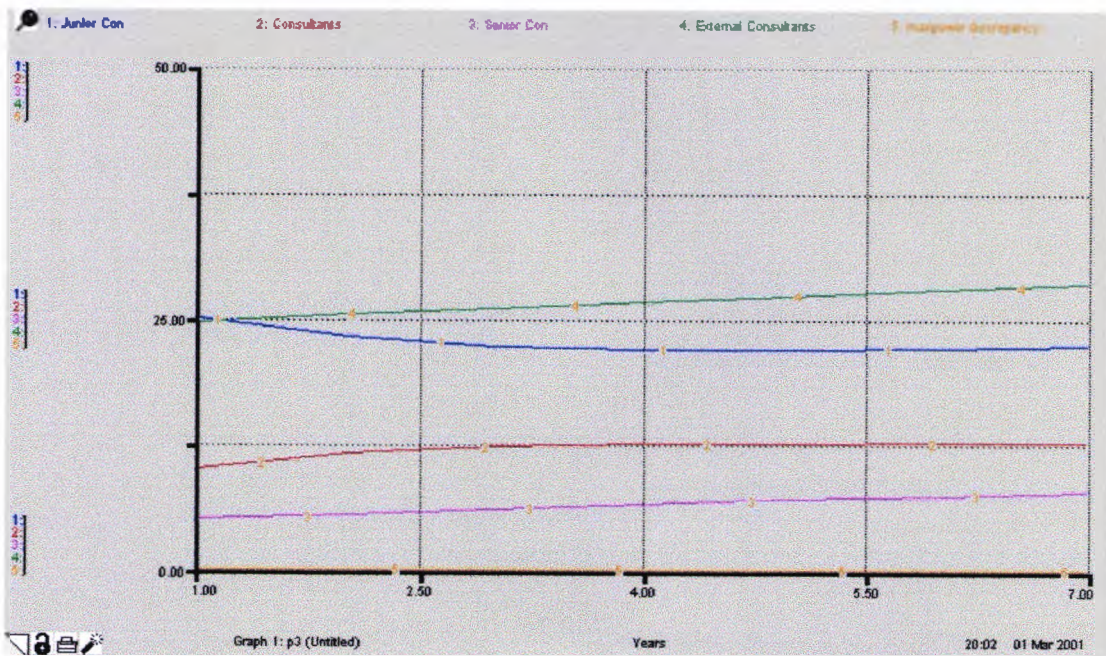


Figure (6.18): The graph of the four levels of resources when manpower discrepancy equals zero

| Years | Initial | 1 | 2 | 3 | 4 | 5 | 8 |
|--------------------------------------|---------|--------|--------|----------|----------|----------|----------|
| Junior Con | 25.0 | 23.1 | 22.2 | 21.9 | 21.9 | 22.0 | 22.3 |
| Consultants | 10.0 | 11.6 | 12.2 | 12.6 | 12.8 | 12.8 | 12.7 |
| Senior Con | 5.0 | 5.4 | 6.0 | 6.5 | 7.0 | 7.4 | 7.7 |
| External Consultants | 24.6 | 25.3 | 26.0 | 26.6 | 27.2 | 27.8 | 28.4 |
| con available for projects | 8.5 | 9.8 | 10.4 | 10.6 | 10.7 | 10.7 | 10.8 |
| senior con available for projects | 4.8 | 5.2 | 5.7 | 6.2 | 6.7 | 7.0 | 7.3 |
| hiring | | 10.6 | 10.7 | 10.8 | 10.9 | 11.1 | 11.3 |
| junior quitting | | 7.5 | 8.9 | 6.7 | 6.6 | 6.6 | 6.6 |
| promoting to con | | 5.0 | 4.8 | 4.4 | 4.4 | 4.4 | 4.4 |
| Turnover | 1.50 | 1.90 | 2.40 | 3.10 | 3.80 | 5.10 | 6.90 |
| average charging rate | 2.30 | 2.46 | 2.60 | 2.72 | 2.80 | 2.94 | 3.15 |
| required manmonths | 652.17 | 775.51 | 923.08 | 1,139.71 | 1,357.14 | 1,734.69 | 2,190.48 |
| required no of consultant equivalent | 59.3 | 70.5 | 83.9 | 103.6 | 123.4 | 167.7 | 199.1 |
| equivalent consultant workforce | 56.6 | 57.9 | 59.1 | 60.3 | 61.5 | 62.6 | 63.8 |
| manpower discrepancy | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| hiring externals | | 3.18 | 3.20 | 3.23 | 3.28 | 3.33 | 3.39 |

Table (6.2): Table of values when manpower discrepancy equals zero

- c. A third and final possibility is where the Required no of consultant equivalent is less than the Equivalent consultant workforce. In this case, the company will act (make people redundant or not) according to how big the discrepancy is. Again, this is not an instantaneous process. There is a need for such delays to be captured in the model and Tarek Abdel-Hamid deals with situations like these in his work [Abdel-Hamid, 1] and [Abdel-Hamid, 2], where he develops a model for dealing with the distribution of personnel in a project. Negative discrepancy implies that project members will be transferred out of the project and used in another one, which appears to be “struggling with time”. Below, the graphs of hiring and of the individual levels of consultants are depicted, assuming that the manpower discrepancy is negative for some years. The Figures 6.19 and 6.20 verify that the model behaves as expected and for the years with negative discrepancy, it gives highly reduced or even zero hiring for the year right after.

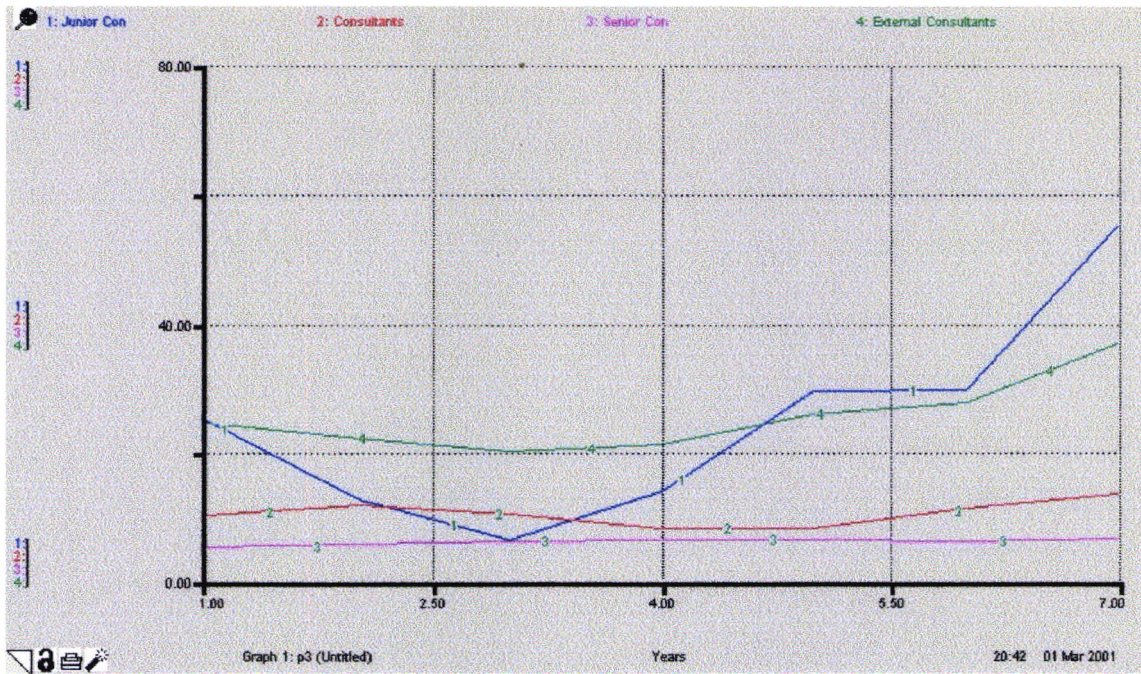


Figure (6.19): The graph of the four levels of resources when manpower discrepancy is negative

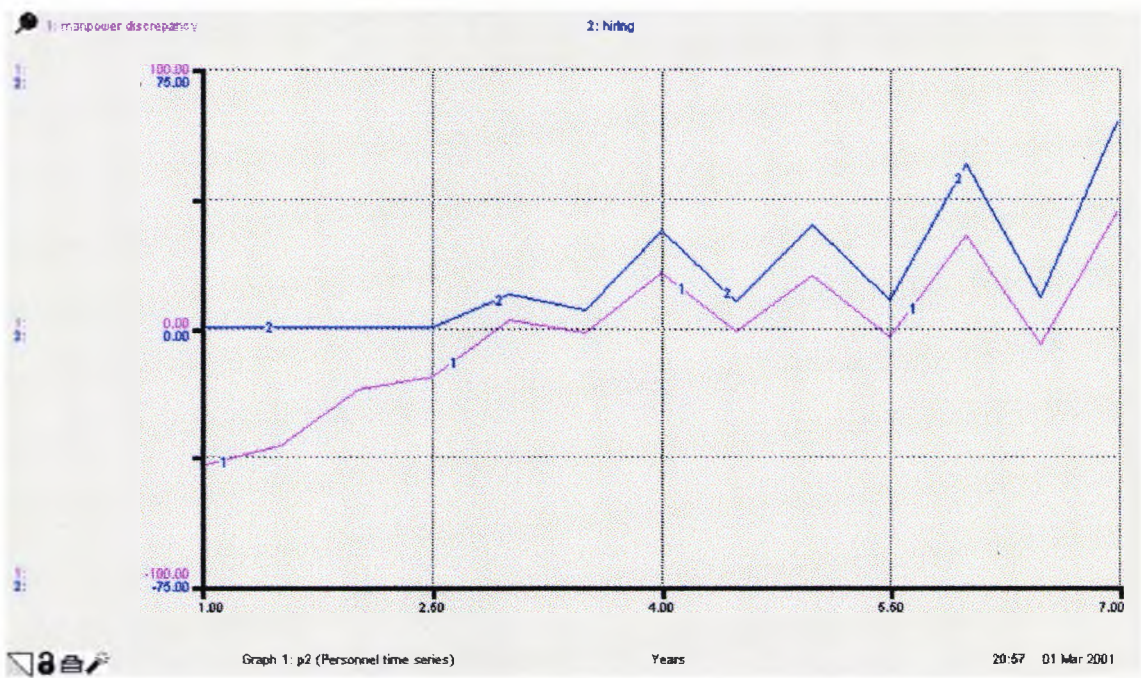


Figure (6.20): The graph of hiring when manpower discrepancy is negative

Such a formal model of the Human Resources management problem has aided in the understanding of the process through its formulation as well as the analysis of its behaviour. Since the System Dynamics model is able to provide a decision-support to answer what-if questions, one could further include more parameters in it, such as cost,

sickness and insurance issues, project delays, and so on, so as to evaluate closer to reality scenarios, or even, try to allocate staff between business development activities and client service delivery. The latter would require the modelling of the Business Development sector, where new business is developed and becomes the backlog of work performed in the Service Delivery sector. This thesis supports the belief that the development of optimisation methods and the research study on the capabilities of Petri Nets in dealing with problems such as the dynamics of the human resources management, would definitely add to the better manipulation of the issue, providing firms with a more powerful tool. It would also help to address and surpass the issue of time as stated in paragraph 6.4.5, by capturing the stochastic quality of processes. In addition, SD could help people to devise strategies for overall human capital management of the company, including training and use of external consultants.

6.10 General comments on System Dynamics

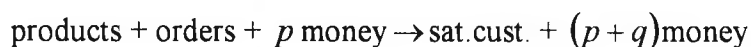
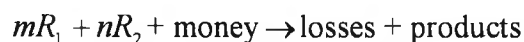
System Dynamics has proven out to be a very helpful tool in getting insight of the issue in question. As already stated, it has so far been applied to a very wide range of problems, dealing with the broad, overall behaviour of the system and how it can influence its own evolution into the future. System dynamics, being so relevant to problems of dynamic nature could even complement other management science approaches, which are more close to static problems. Nevertheless, the analyst is again faced with a blank piece of paper on which a model must be created, and as it happens to other system methodologies, such as the Soft Systems approach, System Dynamics has no fixed form. The saving grace is that all socio-economic systems contain physical flows of something, and getting the physics right is the key step to understanding a problem and getting a good model. The advice to think physics [Coyle, 2] was emphasised when the influence diagrams were discussed, and the principle of physical flow is, perhaps, the nearest thing System Dynamics has to a general theory for structuring and understanding problems.

The main advantage of using System Dynamics is that their models come in two equivalent forms – the influence diagram and the equation form – making it possible to communicate easily and effectively with the people interested in the problem. Particularly, the simulation technique makes it easy and quick to build models

that can afterwards be tested through a number of alternative policies. Hence, the structure of the system might even be redesigned so that its policies become more effective. This extensive freedom to redesign a system's structure makes System Dynamics not only a practical proposition, but also, a discipline capable of providing useful policy guidelines for problems of great practical and human significance. Finally, this methodology finds many applications when designing the specifications and the objectives of the problem due to the fact that even quite complicated models can run in seconds on a PC, maintaining the constant client involvement throughout the study periods.

Nevertheless, one of the difficulties that System Dynamics is asked to face during the testing of different policy scenarios under controlled conditions, is dealing with socio-economics systems, which most of the time contain significant non-linearities. Furthermore, socio-economic systems may well involve positive feedback loops, or badly behaved negative loops, capable of driving the system far outside the normal operating range within which an engineering system can be made to remain. Even further, socio-economic systems cannot always be represented as objects that are separate from the environment that causes the shocks. One consequence of that, is that System Dynamics' models of systems like that, quite frequently produce counter-intuitive behaviour, in which the result of a policy or structural change is different from, or even opposite to what logic might initially have predicted.

Finally, in the case of modelling processes, which have several inputs and outputs, the model we derive is analogue to chemical reactions. For example:



where R_1 and R_2 are two kind of materials and *sat.cust* are the satisfied customers (please refer to [LeFèvre, 2] for more details). System Dynamics is unable to represent such chemical reactions simply and intuitively. Due to the messy character of models derived in physiology, biochemistry, ecology, socio-economics, and management sciences, complex SD (System Dynamics) models are difficult to build and modify. As a result, many people build oversimplified models and avoid eventually fruitful reaction metaphors. Motivated by this fact, [LeFèvre, 1] developed a generalised SD called "Kinetic Bond Graphs (KGB)". [LeFèvre, 1] has elaborated a progressive methodology incorporating his multi-level diagrams into the soft system approach of [Wolst., 1] and

[Coyle, 2]. This is an interdisciplinary development. One starts from qualitative views and progressively refines them using not only the KGB Russian doll diagrams but also causal loop charts and systematic factor discovery.

6.11 The discrete-time model of the consulting firm's problem

As already stated in the Equations Level of the Ithink software (Appendix, Section 5), the mathematical model can be described by a set of simplified discrete-time equations. In this paragraph, a formal discrete-time model is produced to compensate for the “softness” of the equations already developed. Through this new model, mathematical theory can support a far more rigorous analysis of the problem. Parsing the narrative of the consulting firm's problem, as presented in chapter 2, four different levels of consultants are assumed; the Junior consultants (J), the Consultants (C), the Senior consultants (S), and the External consultants (E). It is also assumed that the company hires only junior and external consultants according to future demand for projects. The hiring of externals (u^β) is a fraction n of the hiring of internals (u^α), and the demand for consultants (y) is expressed in units of consultants. The system dynamics are described by the following set of equations:

$$u_k^\beta = n u_k^\alpha \quad (1)$$

$$J_{k+1} = \alpha(k)J_k + u_k^\alpha \quad (2)$$

$$C_{k+1} = b(k)C_k + \gamma(k)J_{k-\mu} \quad (3)$$

$$S_{k+1} = \delta(k)S_k + \varepsilon(k)C_{k-\lambda} \quad (4)$$

$$E_{k+1} = \omega(k)E_k + u_k^\beta \quad (5)$$

If W_k is the equivalent consultant workforce at consultant level at time k , and p is the relative productivity of junior consultants and of consultants and senior consultants, who supervise juniors while working on projects, (p is assumed to be a constant), then:

$$W_k = E_k + m(k)C_k + l(k)S_k + p(d(k)J_k + e(k)C_k + f(k)S_k) \quad (6)$$

which indicates that in the mathematical expression of workforce, a proportion $m(k)$ of the consultants and a proportion $l(k)$ of senior consultants are working only on projects, and they are therefore, 100% productive. The rest of these two types of consultants

(whose proportions are defined via parameters $e(k)$ and $f(k)$ respectively) are acting as supervisors and are, at the same time, working on projects. Factor $d(k)$, ($0 < d(k) < 1$) accounts for the reduced productivity of junior consultants relative to the other three types. Equation (6) can further be expressed as:

$$W_k = E_k + q_1(k)C_k + q_2(k)S_k + q_3(k)J_k \tag{7}$$

where $q_1(k) = m(k) + pe(k)$, $q_2(k) = l(k) + pf(k)$, and $q_3(k) = pd(k)$.

In equation (1), n is the external hiring ratio, here is assumed to be a constant. In equation (2), $\alpha(k)$ is the junior consultants growth parameter at time k , i.e., those *remaining-quitting-promoting*, in equation (3), $b(k)$ is the consultants growth parameter at time k , i.e., those *remaining-quitting-promoting*, $\gamma(k)$ denotes the junior-to-consultant promotion parameter, and μ is the delay of promoting from junior to consultant level; in equation (4), $\delta(k)$ is the senior consultants growth parameter at time k , while $\varepsilon(k)$ is the consultant-to-senior-consultant promotion parameter, and λ is the delay of promoting from consultant to senior level; finally, in equation (5), $\omega(k)$ indicates the external consultants growth parameter at time k .

In this model, it is assumed that the consulting company will not make people redundant in case the demand for projects falls, hence, u_k^α and u_k^β are always non-negative. Furthermore, J , C , S , and E are also non-negative integers, since they refer to employees.

Equations (2), (3), (4), and (5) can also be transformed into the following matrix form:

$$\begin{bmatrix} J_{k+1} \\ J_k \\ \vdots \\ J_{k-\mu+1} \\ C_{k+1} \\ C_k \\ \vdots \\ C_{k-\lambda+1} \\ S_{k+1} \\ E_{k+1} \end{bmatrix} = \begin{bmatrix} \alpha(k) & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ 1 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & \dots & 0 & \gamma(k) & b(k) & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & \dots & 0 & 0 & 1 & \dots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 & 0 \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 & \varepsilon(k) & \delta(k) & 0 & 0 \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \omega(k) & 0 \end{bmatrix} \begin{bmatrix} J_k \\ \vdots \\ J_{k-\mu+1} \\ J_{k-\mu} \\ C_k \\ \vdots \\ C_{k-\lambda+1} \\ C_{k-\lambda} \\ S_k \\ E_k \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ n \end{bmatrix} u_k^\alpha$$

This may be written more compactly as $x_{k+1} = A(k)x_k + Bu_k$, which is the state-space realisation of the model (x_k is the system's state-vector).

The optimal recruitment policy in this problem is the control law $\{u_k^a\}_{k=1}^m$ such that $\{W_k\}$ tracks the "demand signal" $\{y_k\}$ by minimising the performance index:

$$J = \min \sum_{k=1}^m |y_k - W_k| \quad (8)$$

In other words, one tries to minimise the sum of the discrepancy between the demand for projects and the equivalent consultant workforce, over a finite horizon of m time-periods. Note that this performance index places an equal weight on either positive or negative deviations from the demand level at each time-period, i.e. it penalises equally the firm's over-resourcing and under-resourcing in staff.

Before concluding this section on describing the discrete-time equations of the problem, it is interesting to depict the consulting firm's problem in its control equivalent form, which is nothing else but a closed-loop system, as shown in the figure below:

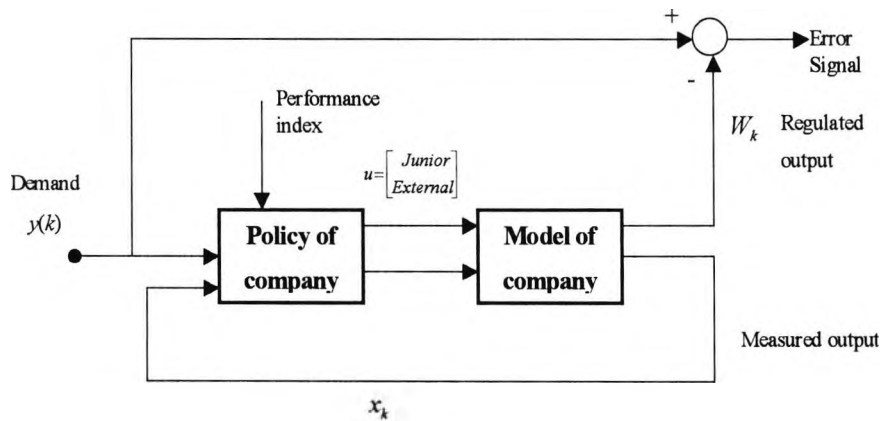


Figure (6.21): The control equivalent form of the consulting firm's problem

As can be seen from the above diagram, the company's hiring policy acts as a feedback controller. The inputs of the controller are: (i) the external demand for projects y_k , (ii) the measured output x_k , which is the system's state-vector at time k , i.e. the number of the consultants employed at various levels, and (iii) the performance index/constraints that the company wishes to optimise/satisfy (for example, not exceeding deadlines for project completion, minimising the cost of employing external resources/subcontractors, etc). The overall objective is to make the error signal "as

small as possible”, i.e. to match the company’s human resources to project demand, as closely as possible, over the specified time-horizon.

The discrete-time model described above may be used to run simulations forward in time and hence, serve as a “prediction” tool for different policy scenarios. These simulations could be used to evaluate, for example, the effects of different hiring policies or different average waiting times between promotions (parameters λ and μ), on meeting future project demand or on the future mixture of the various types of employees. Alternatively, the model is suitable for formulating and solving optimisation problems, in order to determine the “best” policy according to the specified criterion, by using dynamic programming or another optimal control method.

The present formulation of the problem is fully *deterministic*, i.e. the effects of different hiring policies are assumed to be exactly predictable. An important issue affecting optimal decisions is the *time when the demand for projects becomes known*. Consider the following two extreme scenarios: In the first, the demand for projects for *each* time-period becomes known only at the beginning of the period; in the second, the whole project-demand profile over the m time-periods is available at the beginning of the *first* period. Clearly, the first problem is easy to solve but not very interesting. In contrast, the second problem is far more complicated due to its *dynamic* nature: For example, it is conceivable that the optimal hiring policy in the early periods could exceed project demand over these periods, to compensate for a sharp rise in demand over the later periods (note that the employees’ growth is constrained by the dynamics of the system, including waiting time-intervals between promotions, or possibly constraints on the magnitude of the control signal $|u_k^a|$). Clearly, the optimal cost when project demand over the entire horizon is fully known in advance (i.e. under “preview” control), *can not exceed* the optimal cost when the demand of projects over each period becomes only known at the beginning of the corresponding period. The difference between these two optimal costs represents the “*value of information*”, which is a characteristic of the model and the project-demand profile.

One could in principle produce a more complex model by making the more realistic assumption that future project demand (and/or some model parameters) is known only probabilistically. In this case the model becomes stochastic and the cost function is a random variable (e.g. the *expected* sum of absolute deviations over a finite

horizon). Developing such a complex model, however, is beyond the scope of this thesis, and is only mentioned here as instigation for further research.

6.11.1 Dynamic Programming Implementation

Following the discussion on the formulation of a discrete-time model of the consulting firm's problem, a first, rather simplified form of the model is developed and simulated in Matlab. The main idea behind the development of the model is the implementation of a conceptually simple mathematical technique, called Dynamic Programming [Berts., 1] to optimise the performance index defined in equation (8). This technique deals with a very wide class of problems where decisions are made in stages. The outcome of each decision may not be fully predictable but can be observed before the next decision is made. The objective is to minimise a certain cost – a mathematical expression of what is considered as a desirable outcome.

In our case, the dynamic programming (DP) algorithm for deterministic problems is applicable. For further analysis of stochastic optimisation problems and the ramifications of the presence of uncertainty, the reader may refer to [Berts., 1]. Consider now a deterministic problem; then at any state x_k a control u_k can be associated with a transition from state x_k to state $f_k(x_k, u_k)$. Thus a finite state deterministic problem can be equivalently represented by a graph such as the one of Figure 6.22, where the nodes correspond to states, the arcs correspond to transitions ($x_{k+1} = f_k(x_k, u_k)$) between states at successive stages and each arc has a cost ($g_k(x_k, u_k)$) associated with it. There is also an artificial terminal node t added to the graph. Each arc connecting a state x_N at stage N to the terminal node has cost $g_N(x_N)$. Control sequences correspond to paths originating at the initial state (node s at stage 0) and terminating at one of the nodes corresponding to the final stage N . Hence, viewing the cost of an arc as its length, the deterministic problem is equivalent to finding a shortest path from the initial node s of the graph to the terminal node t [Berts., 1]. If we denote:

- $c_{ij}^k =$ cost of transition from state $i \in S_k$ to state $j \in S_{k+1}$, $k = 0, 1, \dots, N-1$
- $c_{it}^N =$ terminal cost of state $i \in S_N$,

the DP algorithm takes the form:

- $J_N(i) = c_{it}^N, i \in S_N,$
- $J_k(i) = \min_{j \in S_{k+1}} \{c_{ij}^k + J_{k+1}(j)\}, i \in S_k, k = 0, 1, \dots, N-1$

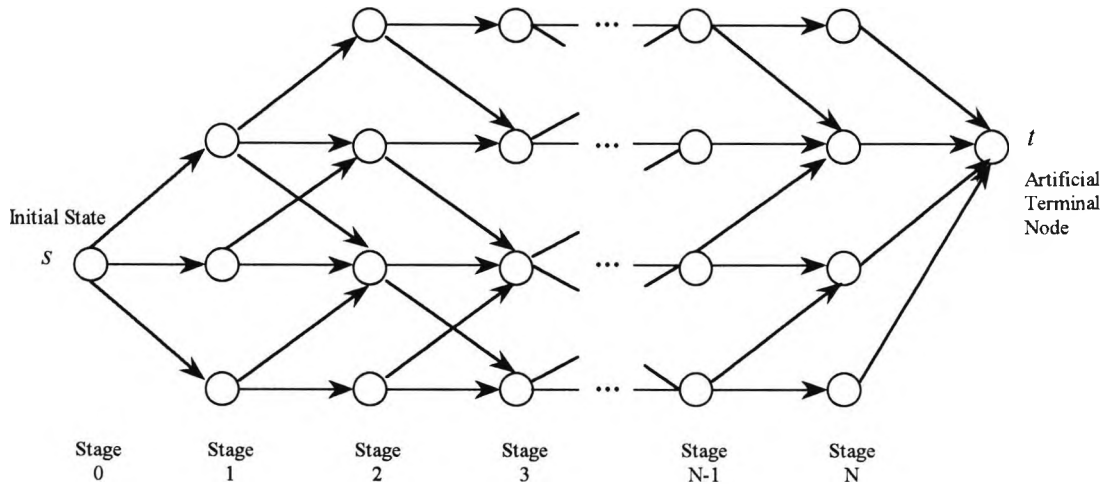


Figure (6.22): Transition graph for a deterministic finite state system [Kerts., 1]

The optimal cost is $J_0(s)$ and equals the length of the shortest path from s to t . $J_k(i)$ is actually the optimal cost for getting from i to t in $(N-k)$ moves (cost-to-go). The recursions implementing the DP algorithm are based on *Bellman's principle of optimality*, which states that any sub-path along an optimal trajectory must itself be optimal.

Returning to the present problem, two Matlab programs, (which may be found in the Appendix) have been developed, in order to simulate the simple case of a consulting firm that hires up to u_{\max} junior consultants per period. The first program is a subroutine defining the state transition map of the model; in other words, equations (1) to (6) shown in the previous paragraph are implemented, and the new state vector x_{new} is obtained, corresponding to the old state vector x_{old} and the control input u ; in addition the output variable w is also calculated, i.e. the equivalent workforce at consultant level corresponding to the (old) state-vector. The second program solves the optimisation problem (8), using the DP algorithm. Starting with an initial state x_0 , the state-space sets $S_k, k = 1, 2, \dots, N$ are first calculated recursively, proceeding forward in time. These are the "reachable" state sets at time k corresponding to all possible control

inputs $u \in \{0, 1, \dots, u_{\max}\}$, and thus contain at most $(u_{\max} + 1)^k$ distinct elements. Some additional information is also stored at this stage (the control input u_k leading to each state x_{k+1} and the corresponding “partial cost” $|W_k - y_k|$). Next, the DP recursions are implemented, starting from the terminal node N and proceeding backwards in time, producing the optimal “cost-to-go” associated with each state, and thus finally the optimal cost J_0 . A forward sweep is finally employed (using the stored information) to derive the optimal control sequence.

The programs were tested successfully for various demand-sequences in problems of low dimensionality ($u_{\max}=4$, $N=6$). As a further verification that the algorithm is implemented correctly the following simulation test was performed: Initially, for an arbitrary initial state-vector and an admissible control policy ($0 \leq u_k \leq u_{\max}$), the states (x_k) and outputs (W_k) produced by the state-transition function were obtained. Subsequently, the project demand sequence was set equal to the output sequence ($y_k = W_k$ for all k), and the DP Matlab routine was executed. As expected, the program produced zero optimal cost and the optimal control sequence was found to be identical with the original control sequence.

We conclude this section by making the following remarks:

- The implemented model is deterministic, time-invariant (parameters $\alpha(k)$, $b(k)$, $\gamma(k)$, $\delta(k)$, $\varepsilon(k)$, and $\omega(k)$ are constants). Minor modifications are needed for the algorithm to accept time-varying parameters, changing only slightly the complexity of the problem. Although equations (2)-(5) defining the model are linear, rounding non-linearities are introduced in their implementation, since state-variables correspond to people and must therefore be non-negative integers.
- The discrete-time model, as developed in this simplified form, gives one the ability to derive optimal policies for hiring, promoting, and outsourcing of consultants, when information on future demand for projects is provided. Even when the demand for projects is not exactly known, stochastic dynamic programming techniques can still be applied by introducing statistical expectations in the cost-function. Such a model may well serve as a starting point for further research to produce more realistic formulations of the problem.
- Although the programs were tested successfully for small-size problems, the implementation of medium/large-scale problems was found to be problematic due to

memory limitations and slow execution time. This is partly due to the software platform used (Matlab), whose capabilities are limited when heavy recursive calculations are required. Although implementation using a non-interpreted based computer language would certainly improve the situation, more efficient numerical algorithms are ultimately required. This means dealing efficiently with complexity issues (the number of nodes can become extremely large). Manipulating models with that many nodes can easily become a daunting task due to the computer time required for processing. Indeed, the nature of some shortest path problems is such that the solution becomes very simple once the nodes of the underlying graph are enumerated, and the real issue is how to solve the problem while avoiding a complete enumeration of all nodes [Berts., 1]. For issues related to the efficient implementation of the DP algorithm see [Berts. & Tsits., 1], [Berts. & Tsits., 2].

- The choice of an appropriate cost function is an important one, since the type of solution obtained ultimately depends on this choice. Cost function (8) places an equal penalty on either positive or negative deviations from demand, which may be inappropriate (a negative deviation implies that projects cannot be completed in time). This could be addressed by defining a cost function of the form:

$$J = \min \sum_k |L_k| \quad \text{where } L_i = \begin{cases} w_i - r_i & w_i \geq r_i \\ \beta(r_i - w_i) & \text{otherwise} \end{cases} \quad \& \beta > 1$$

so as to avoid having the same penalty in all cases, i.e., when we meet the demand and when we are not. An alternative formulation could be to use only external consultants as resources to the problem, and try to match the demand by hiring only externals. In that case, the algorithm would provide the optimal policy of hiring in order to minimise the cost of external consultants with respect to the cost of internals.

6.12 Conclusion

So far we have seen what System Dynamics is for, how it relates to the rest of management science, its main processes and how it supports top-level decision issues. Despite its origin in business management, System Dynamics is clearly applicable to social problems, and this has proved to be a fertile area of application in socio-economy,

ecology, biology, and other semi-quantitative fields. System Dynamics is quite successful in these fields because its main variables (flows and rates) are the only one easily measurable in these domains. Also, the graphical representation and theoretical background of System Dynamics is so simple that it may be easily understood by everybody without extensive training. Finally, System Dynamics may be used to represent any set of first order differential equations although sometimes in cumbersome ways.

The fundamental purpose of System Dynamics is to achieve comparable quality of design, and hence performance, in managed systems. This involves analysing a managed system so as to:

- Model the ways in which its information, action and consequences components interact to generate dynamic behaviour.
- Diagnose the causes of faulty behaviour.

The above analysis was made obvious through the manipulation of the consulting firm's problem, where we identified the key components (entities) of the model and built the structure of it by means of influence diagrams, we described the way dynamic behaviour is generated (increase in market demand versus the composition of equivalent consultant workforce), and run various policy decisions so as to evaluate the best possible behaviour of the model. We could have further made use of the optimisation methods but this would exceed the scope of this thesis, which is the evaluation of a number of modelling methodologies for business processes, and not the building of a perfect modelling solution for the human resources management issue. Nevertheless, such a formal model of the human resources management process aids in the understanding of the process through both its formulation as well as the analysis of its behaviour.

The development of a discrete-time model, emerging from the equations of the equation editor of Ithink, and the use of the dynamic programming algorithm have broken new ground of research in the area of soft systems. The example provided in previous paragraph shows that it is possible to "bring some order in the chaos" through the use of mathematical tools. Further work on the subject can justify the above statement.

System Dynamics share the same aim with the conceptual modelling approach; that is to capture knowledge about a specific problem and represent it in such a way so

as to enable the system developer to reason about this knowledge, specify the allowable structures and transitions, and communicate this understanding to the end users for validation. In our view, these two methodologies appear as complementary. There can be no definition of the problem (understanding of the users' needs, the business rules and structuring of the acquired knowledge) without the conceptual modelling methodology. The latter offers a more structured list of guidelines to help the analyst in the first two stages of System Dynamics approach. Having said that, System Dynamics serve to move the analysis a bit further. Dynamic (time evolving) systems are better manipulated with SD, where flows incorporate the time concept more effectively than the time-stamped relationships between objects do, and simulation, as well as optimisation methods add to the power of dealing with managed systems. In addition, SD can further benefit from the powerful machinery of discrete-time mathematics. The fields of application are now only limited by the wit and imagination of the analyst.

Chapter 7

ROLE BASED MODELS: ROLE ACTIVITY DIAGRAMS

7. ROLE BASED MODELS: ROLE ACTIVITY DIAGRAMS

7.1 Introduction


Nowadays, there are still quite a few organisations which do not recognise that they operate processes, even though they know perfectly well how they are structured into functions and what each of those functions is responsible for. Simply by modelling these processes, individuals and groups can be provided with a perspective on the organisation, promoting, in this way, a more collaborative spirit.


One of the methods used to model organisational processes is the Role Activity Diagrams (RADs), which represent the whole of a process as far as we wish to capture it. A RAD shows the roles, their component activities, and their interactions, together with external events. It concentrates on the “organisational behaviour” of the process being modelled and does not consider the “computational detail”. Once the RAD has been prepared, a number of questions can be asked of the resulting model and the process it describes. Of course, no process model can of itself provide the answer to a process problem; it can only act as a sort of searchlight on the process. After all, the purpose of a process model is to reveal the process, the roots of its problems, and the potential ways of attacking the trouble.


In the following pages of this chapter, the reader is provided with a brief summary of the basic concepts of RADs, and then this method is used to model the consulting firm’s problem. Finally, the chapter dwells on the pros and cons of using RADs to model business processes.


7.2 Basic Concepts


The key concepts sought in this process modelling language will derive from the needs of the process modeller. We will therefore use the concepts of *roles*, which are composed of *activities*, which produce and operate on *entities*, and which communicate, co-ordinate and collaborate through *interactions* [Ould, 1].


A *role* involves a set of actions that are generally carried out by an individual or group within the organisation. Each role in the process is represented by the contents of a shaded block (). Within each role there are a number of *activities*, indicated by

black boxes (■). The little spring-shaped symbol () indicates that the RAD continues but that we are not concerned with how.

One role can *instantiate* another role, i.e., start a new instance of that role: this is indicated by a square with a cross inside it (). In a RAD there is no separate symbol to represent the “ending” of a role instance once its work is done.

An *interaction* between roles is shown as a white box in one role connected by a horizontal line to a white box in another role (). An interaction is always annotated in a way that makes clear what is happening, and that might indicate whether any grams (entities) pass during the interaction. Additionally, an *entity* is the name one gives to anything which is the subject matter of an activity. An interaction can involve any number of parties. The overall *triggering* condition is effectively the state reached when all the participating roles are ready for the interaction. The modeller always shows whatever detail is appropriate to that model for that purpose.

In some situations, it is useful to show which party to an interaction takes the lead or is responsible for making it happen. We do this simply by shading in the part-interaction in the “driving role” ().

The vertical lines between activities represent *states* or *conditions* that the role can be in. Understanding of a RAD is greatly enhanced if lines are seen as states rather than just representing “flow” from one activity or interaction to another. To capture this idea (the activity starts), the method states that the state in front of the activity represents its *triggering condition*, that is, the condition of the role instance which will cause the activity to start ().

7.2.1 Representing alternative courses of action

At some points in the process, what happens next might depend on some condition or state. Such alternative courses of action are represented with *case refinement*. An example can be found in Figure 7.1 is *a consultant capable to carry on working?*. Using the token scheme, we can think of a token passing down the leg of the case refinement that corresponds to the predicate that is true; the role goes in different directions depending on the state of things. Figures 7.1, 7.2 and 7.3 illustrate this.

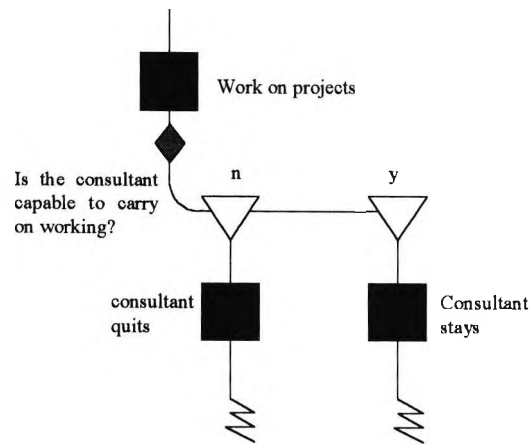


Figure (7.1): The state before a case refinement

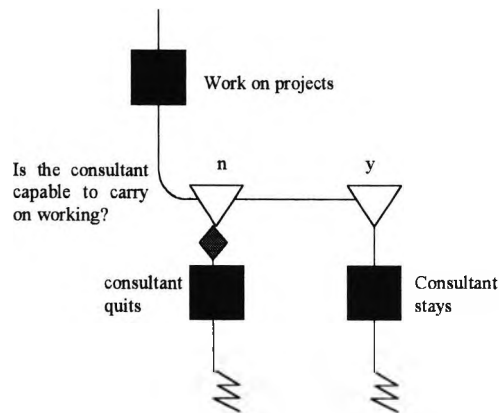


Figure (7.2): The state after a case refinement in one case

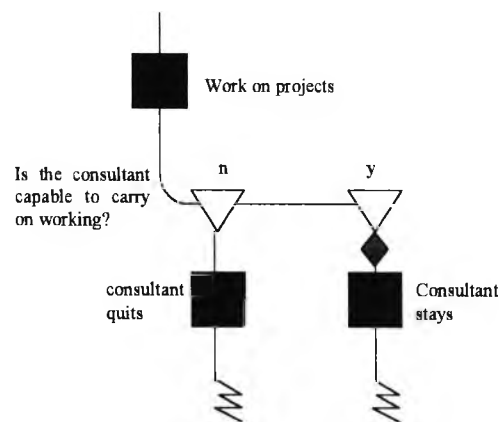


Figure (7.3): The state after a case refinement in the other case

Immediately after the activity *Work on projects*, imagine a token on the state line leading out to *Work on projects*'s black box as shown in Figure 7.1. The case refinement says that, if the answer to the predicate *Is the consultant capable to carry on working?* is yes, this is equivalent to Figure 7.3 with the token on the corresponding

state line; whilst, if the answer is no, it is equivalent to Figure 7.2 with the token on the other state line. The two-way case refinement generalises quite naturally to N -way case refinements.

7.2.2 Representing concurrent threads of action

There might be a point at which a role can start a number of separate threads of activity that can be carried out concurrently. *Part refinement* can involve any number of threads of concurrent activity, depending on just how much concurrency is possible in the work of the role. An example of part refinement is the split in the *Consultant* role in Figure 7.4 where a consultant carries out two concurrent threads of activity. On one thread the consultant supervises junior consultants; on the other he works on projects.

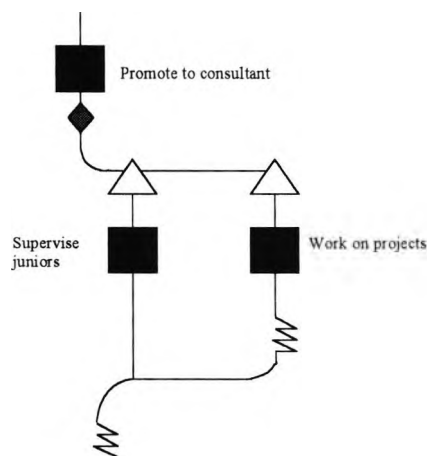


Figure (7.4): The state before a two-way part refinement

Using tokens once again, one can think of the single token that reaches the part refinement becoming a number of tokens, each of which passes down one thread of the part refinement. In Figure 7.4, the activity *Promote to consultant* has been completed and there is a state token on the state line coming out of *Promote to consultant* and before the part refinement. This state is entirely equivalent to that shown in Figure 7.5 where the token before the part refinement has turned into one of each of the separate part threads.

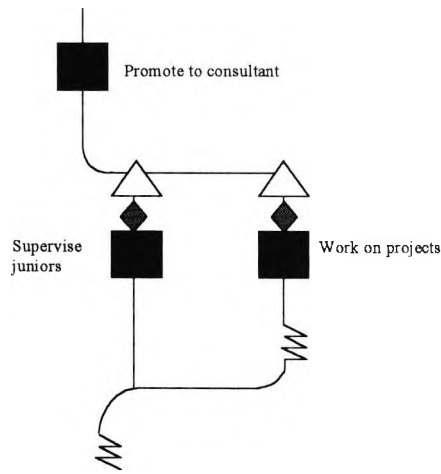


Figure (7.5): The equivalent state at the start of the part refinement

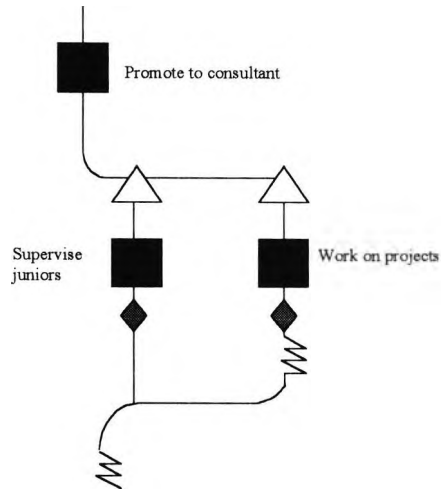


Figure (7.6): The state at the end of the part refinement

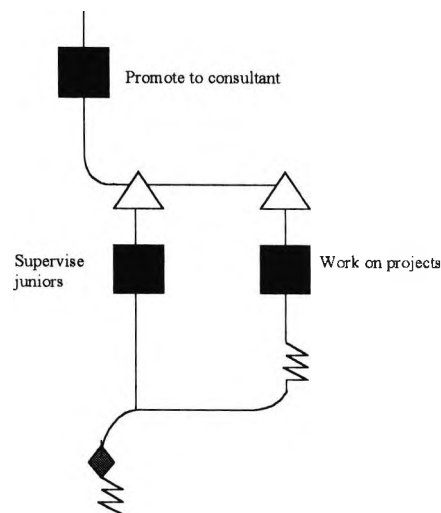


Figure (7.7): The equivalent after the part refinement

In the representation in Figure 7.6, the threads join once they have finished, i.e., at some point after the part threads have gone their separate ways, they will all have finished and hence there will be a token sitting on the state at the end of each thread. This is entirely equivalent to the state shown in Figure 7.7 where the part thread tokens have all been replaced by a state line immediately after the closure of the part refinement.

7.2.3 Interactions

A RAD always shows strict ordering. The rules of an interaction state that an interaction cannot begin until the participating roles are in their respective pre-states, and that when the interaction finishes they both move into new respective post-states [Ould, 1].

7.3 Modelling business processes

There are times when the process being modelled is relatively small and it can therefore be drawn to the level of detail on a single sheet of paper. But most of the time, especially where an entire business is being modelled, a number of issues arise as to how large processes should be captured and how detailed the model should be. Usually, people will want to be able to view a process from different levels, perhaps ranging from a high level, where the broad brush perspective is given, with detail removed, to a low level, where the minutiae of the process can be inspected [Ould, 1]. RAD notation offers exactly this feature, i.e., one can model a process from any perspective we choose.

Because organisational processes tend to be multi-dimensional networks, rather than hierarchical processes, RADs make use of three different relationships that a number of processes might have. These are:

1. *Composition*: many processes do not operate entirely separately from one another – there are points at which they intersect or synchronise in some way.
2. *Encapsulation*: one process is in some way an “expansion” of an element in the other.

3. *Activation*: one process starts another, which then operates independently, possibly meshing subsequently.

For an extensive account of these three relationships, refer to [Ould, 1]. What is important to point out here is that RAD methodology provides ways of producing free-standing RADs which are at the same time showing where they are connected. Sometimes, the goal will be to bring people into thinking about a process, hence a simplified version of a model can help. Other times, too much detail does not contribute anything extra; in fact it can get in the way of producing a quite accurate model of the process under study. A model can therefore be reduced in three ways:

1. by “bundling” process elements into a single activity or interaction
2. by combining related roles
3. by simply ignoring “unimportant” roles or interactions

7.3.1 Getting the “big picture” for the consulting firm’s problem

A RAD is not only suitable at the detailed level. It can as well be used to give the “big picture” of what a process does. Figure 7.8 is such a picture.

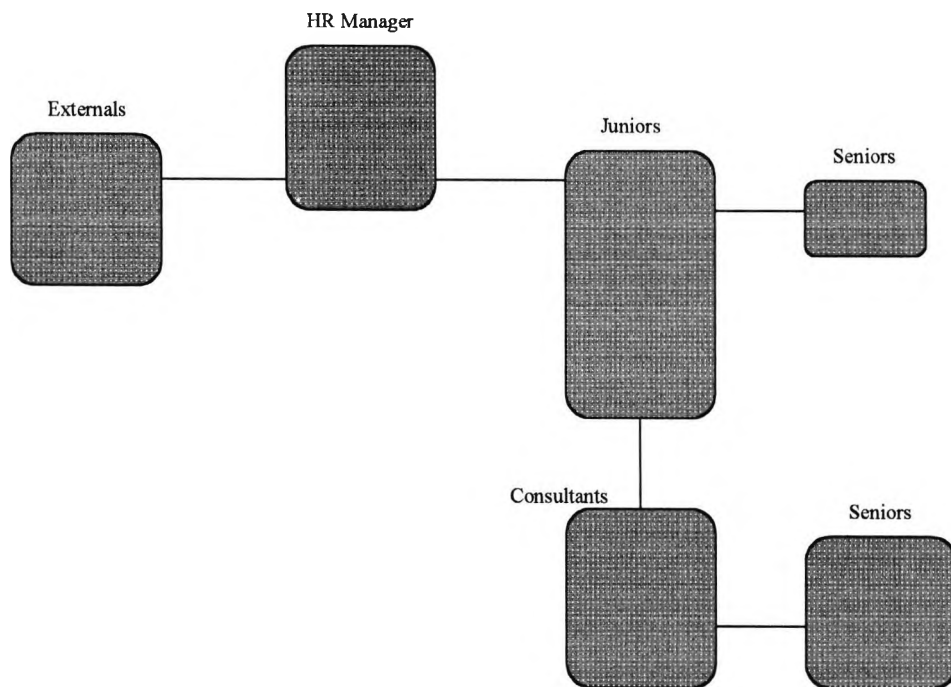


Figure (7.8): A summary RAD for the consulting firm’s problem

Even in such a big picture, the RAD preserves a degree of information about the way roles interact with one another. Here, we see that the Human Resources Manager has interactions with the External and Junior Consultants, Junior Consultants interact with the Consultants and Senior Consultants, and finally, Consultants have interactions with the Senior Consultants. Nothing else is said about the behaviour of the process or what is actually done. By no means, should abstraction be seen as some sort of summarising. The aim is to model the process in terms of intent rather than mechanism.

7.4 Drawing the RAD for the consulting firm's problem

In this paragraph, the effort will lie on using RAD's basic concepts and logic in order to model the consulting firm's problem, as it was formulated rather elaborately in chapter 2. The first task is to identify the roles for this specific problem, i.e., to identify the different areas of responsibility for some contribution to the process. As seen previously in the Figure 7.8, there are five main roles that interact with each other. The Human Resources Manager reviews the demand for consulting services every three months (see Figure 7.9). We use the event arrow \Rightarrow to indicate this kind of activity. The event arrow is a commonly used notation to show external events triggering activity in a role in the process we are modelling, or to represent the passage of time. The HR Manager then makes sure that the company personnel are sufficient to meet the demand. If the answer to the predicate *Personnel sufficient?* is no, then he issues the commencement of the hiring period.

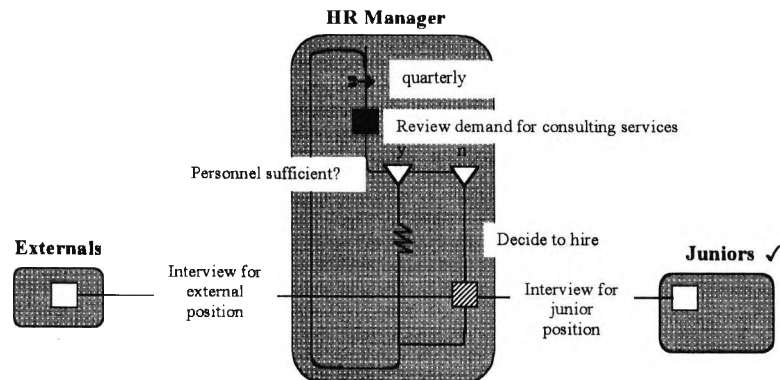


Figure (7.9): The RAD for the HR Manager role

Here, the assumption is that the activity of hiring takes place only in the external and junior consultants' levels. The Externals' role interacts with the HR Manager role, and as seen in Figure 7.10, the life cycle of an external consultant is pretty simple. After being hired, an external works on projects, but by the time the project finishes, he leaves the company. The little spring-shaped symbol indicates that the RAD continues but the reader is not in this process concerned with how.

The RAD for the Junior Consultants role is quite more complicated and interesting in the way Juniors interact with the HR Manager, the Consultants and the Senior Consultants (see Figure 7.11). After being hired, a junior starts working on projects while at the same time he receives supervision by the consultants and the senior consultants. The tick (✓) next to the name of the role indicates that this given role has a pre-existing instance.

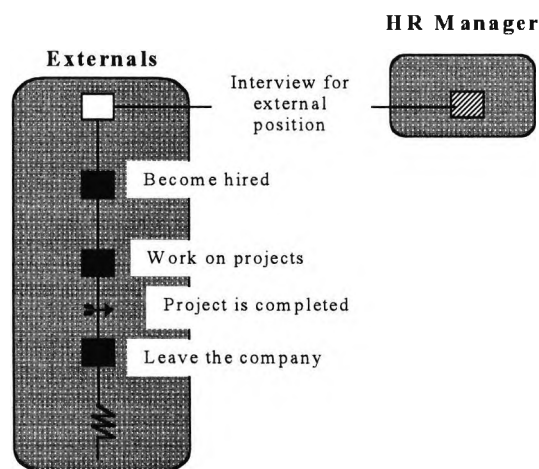


Figure (7.10): The RAD for the External Consultants role

When the evaluation time for junior consultants arrives, juniors either leave the company or are promoted to fully-fledged consultants, according to the result of the evaluation.

The newly promoted consultants begin to work concurrently as supervisors and in projects. Note the part refinement in Figure 7.12 where the activity *become consultant* breaks down into two separate threads *supervise juniors* and *work on projects*. These two activities are joined later on before the role can proceed to a further activity. Two successive case refinements follow. The first deals with the ability of the consultants to carry on working in the firm; if the answer to the predicate *capable to carry on working?* is yes, the second case refinement appears. A yes to the predicate

promote? moves the consultant to a higher level, the one of the senior consultants, while a no brings him back to what he has already been doing, i.e., supervising and working on projects.

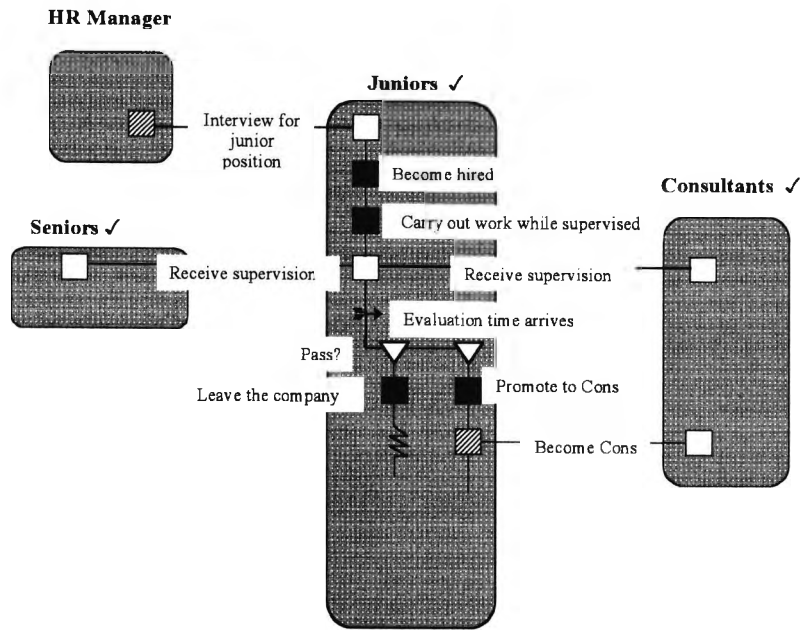


Figure (7.11): The RAD for the Junior Consultants role

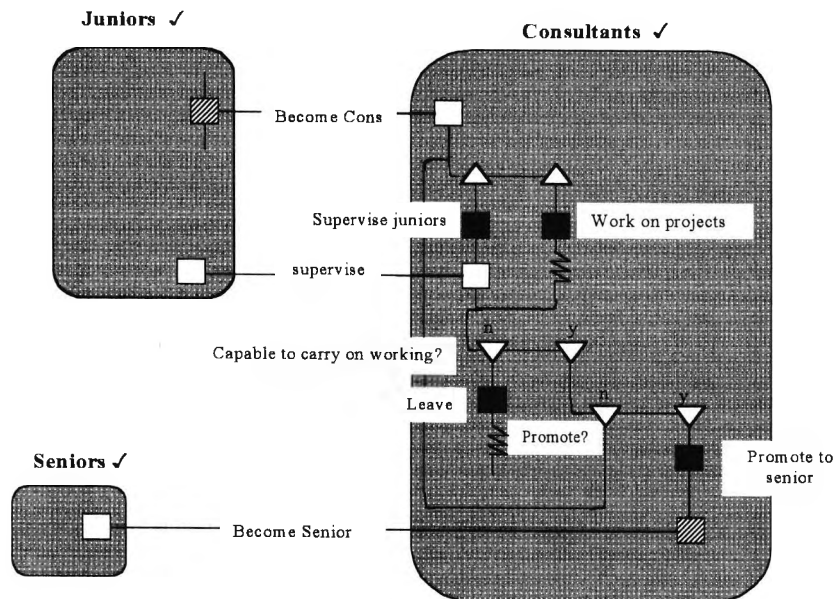


Figure (7.12): The RAD for the Consultants role

The Senior Consultants RAD is quite similar to the Consultants' one. Here there is only one case refinement. Senior consultants either stay with the company or decide to leave and pursue other career prospects (see Figure 7.13).

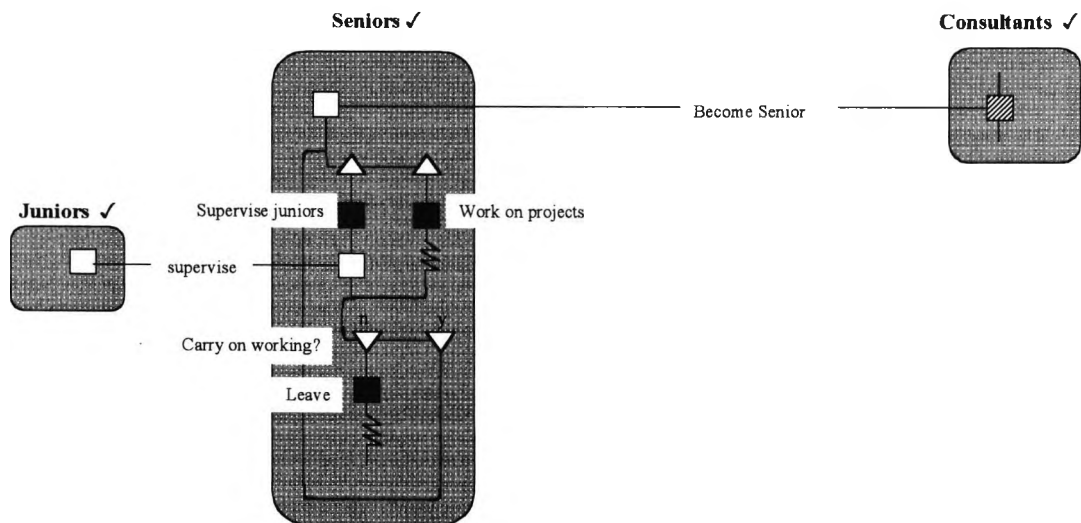


Figure (7.13): The RAD for the Senior Consultants role

The overall RAD for the whole process can be seen in Figure 7.14. In this, the order and concurrency between activities and external events is reported. It is in the hands of the analyst to come up with a more summarised or detailed version of things.

Just by examining the inputs and outputs of activities and the flow of grams in interactions, one can build the basis for a data flow model of the process. Also, once a RAD has been prepared, a number of questions can be asked of the resulting model and the process it describes. Those involved in modelling can look for features in the model, which might help them to understand or improve or re-engineer the process.

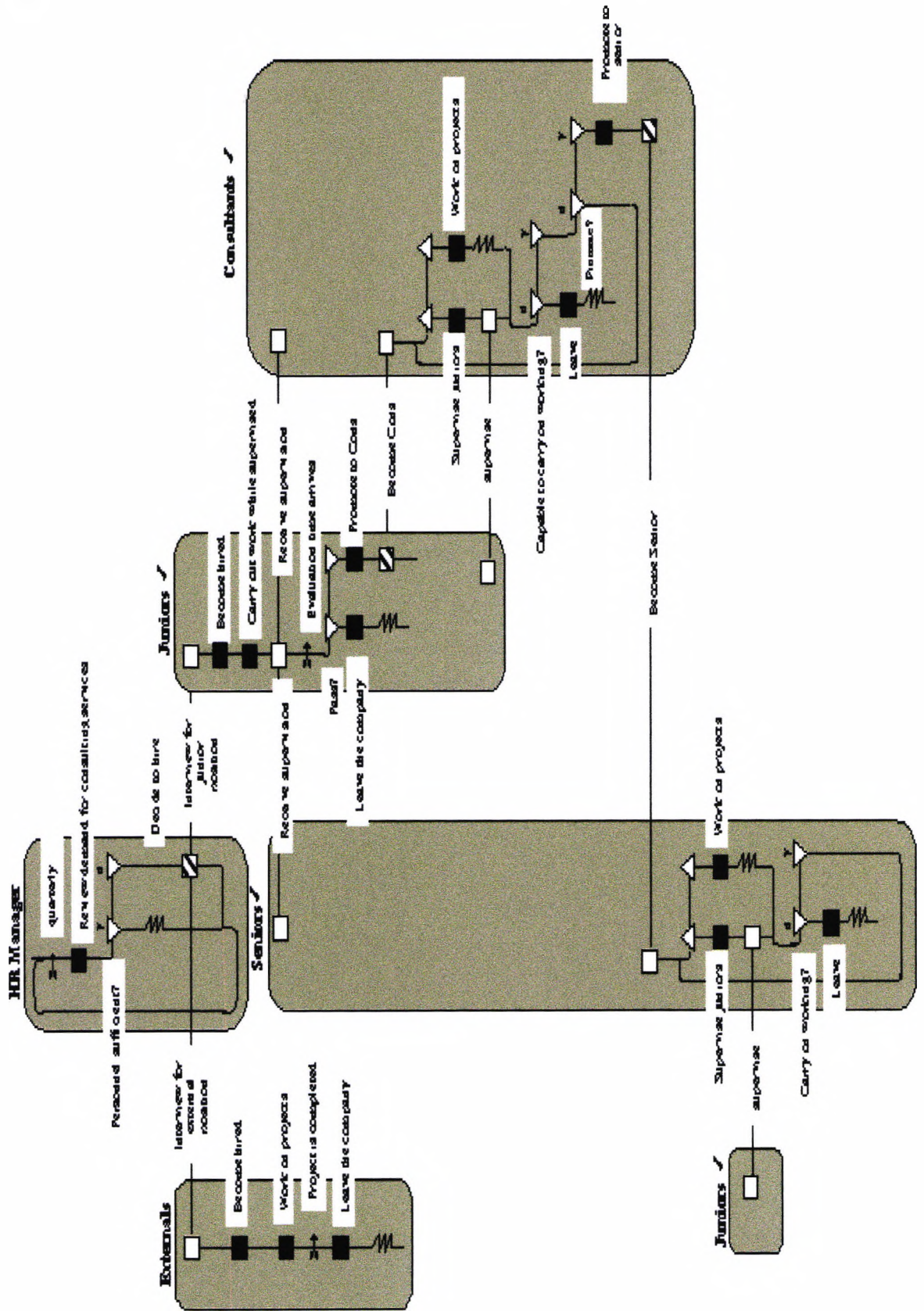


Figure (7.14): The RAD for the consulting firm's problem

7.5 General Comments on RADs

The Role Activity Diagrams are intuitively easy to read because they concentrate on people and on what people do. They provide an excellent intellectual machinery for working with processes, let alone the fact that they are very good in drawing pictures of the problem. As [Ould, 1] mentions: "...in this approach to process modelling, we concentrate unashamedly on what people *do*, rather than on what people do it with." Once a process model, i.e., a description of how the business does its business or plans to do its business, is given, then the investigation on the information needs of the process commences.

As stated, entities are the subject matter of a process and they are dealt with or created or consumed by the activities in the process. The state of these entities is represented in the form of data that people conventionally hold in databases. If a database system is going to be built then it is better to analyse the process being automated from the point of the entities this is concerned with. Conceptual modelling supports the modelling of rather hierarchically structured things. Since in a process there is usually not a very neat hierarchy of activities, RAD can act as a complementary tool for building models for business processes and for helping to create the necessary databases.

Role Activity Diagrams also seem to encompass some of the logic of Petri Nets. When in a previous paragraph, the way a RAD animates by following the movement of the token through the numerous activities was described, the notion of a Petri Net emerged. We showed the state of an instance by placing tokens on the appropriate state lines. The position of those tokens can be referred as the 'marking' of the role instance. Actually, a RAD can be recognised as a Petri Net with the activities, external events and interactions as transitions, and the state lines as places. Tokens then flow around the net, with the transitions taking the net from one marking to another in the traditional way.

In case of looking to find a reduced time-to-process, System Dynamics can help. As seen in Chapter 6, a process is perceived as a set of flows of material between stocks. These flows form a network that can include feedback loops and alternative paths of materials. Software tools for animating SD models allow flow rates to be made dependent on stock levels, on external variables, on the passage of time, on the date or

time, and even on each other; in this way one ends up with a model that can be animated allowing the analyst to determine the cycle time. This is a much richer representation, especially if the effort is to understand the variability of the time a process takes [Ould, 1]. A RAD can easily be converted into an SD model by turning states into stocks, and activities and interactions into flows. [Ould, 1] mentions that case refinements become split flows whose rates sum to one, whilst part refinements become joint in which the material is replicated on each flow.

Usually, RAD models contain a great amount of detail proving out to be a very accurate qualitative model. But this kind of detail does not need to be carried into a quantitative model such as the one System Dynamics can provide. To complete a quantitative model, one will need to collect information about other influencing factors that affect the behaviour of the process, as for example the rate at which junior consultants leave the company, staff productivity, the numbers of the different types of consultants and the annual change in turnover.

It is quite obvious that a quantitative model deals with factors that are strictly outside the sort of model a RAD produces. Still, it is better to think RAD and SD models as two complementary tools for the analyst despite some apparent overlap between them. Each model has its own things to tell; hence the analyst should use them alternately to satisfy the needs of the client.

7.6 Conclusion

The nature of the business can change. A process that was once adequate under one set of market conditions can become quite ineffective when those conditions change. Role Activity Diagrams model organisational behaviour in a way that is revealing and communicative. RADs allow the analyst to explore the relationship between the process and the organisation, they show the dynamics of a process and they can be used at a variety of levels, having at the same time sound theoretical underpinnings. Their models provide a step-by-step guide of the process mechanism revealing the roots of their problems and offering potential ways of attacking the trouble. Hence, RADs are a superb tool for helping in the re-engineering and improvement of business processes.

Chapter 8

PETRI NETS

8. PETRI NETS

8.1 Introduction

Petri Nets have been developed from the early work of Carl Adam Petri [Petri, 1]. In his research, Petri was particularly concerned with the description of the causal relationships between events. Since then, the use and study of Petri Nets have increased considerably. Petri Nets is a graphical and mathematical modelling tool designed to describe and study systems, characterised as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. As a mathematical tool, Petri Nets can be used to set up state equations, algebraic equations, and other mathematical models, as well as simulation models. As a graphical tool, Petri Nets provide a visual modelling technique where tokens are used to simulate the dynamic and concurrent activities of systems. The areas of application are many and diverse; communication protocols, real-time systems, workflow systems, flexible manufacturing systems, computer systems, production systems, logistic systems, distributed information systems, and administrative systems are only some of the areas where Petri Nets have been applied.

In this thesis, we focus on a specific class of Petri Nets (PNs) suitable for the representation, validation, and verification of business processes. We will show that it is not difficult to map a process onto a Petri net. This special class of PNs is called High-Level Petri Nets, and are based on the classical Petri net model introduced by Petri, but are extended with “colour”, “time”, and “hierarchy”. These extensions allow for the representation and study of complex business processes and, at the same time, inherit all the advantages of the classical Petri net, such as the graphical and precise nature, the robust mathematical foundation, and the abundance of analysis methods.

In the following pages, the main concepts of Petri Nets and their capabilities as tools for analysis are outlined. The reader is introduced to various subclasses and extensions of Petri nets, but the discussion is mainly focused on High-level Petri nets and their modelling power to deal with business-process-based issues. A list of simulation tools that make use of Petri nets is provided and then the application of Petri nets in the modelling of the consulting firm’s problem is illustrated. The software used for the application is Alpha/Sim[®], developed by Alphatech Inc. General comments on

the power of Petri nets, as well as a discussion on the linkage of Petri nets with the modelling techniques already mentioned, conclude the chapter.

8.2 Basic Concepts of Petri Nets

The classical Petri net is a directed bipartite graph with two node types called *places* (\circ) and *transitions* (\square). Directed *arcs* connect the nodes, but connections between two nodes of the same type are not allowed. Arcs can be labelled with their weights (positive integers), where a k -weighted arc can be interpreted as the set of k parallel arcs. The structure of a net is static. Assuming that the behaviour of the system can be described in terms of system state and its changes, defining its marking, creates the dynamics of a net structure. The *marking* of a net is the assignment of a nonnegative integer to each place. For example, if a marking assigns to place p the integer m , we say that p is marked with m tokens, and, pictorially, we place m black dots in place p . A formal definition of a Petri net is given below:

Definition (8.1): A Petri net is a 5-tuple, $PN = (P, T, F, W, M_0)$ where:

$P = \{p_1, p_2, \dots, p_m\}$ is a finite set of places,

$T = \{t_1, t_2, \dots, t_n\}$ is a finite set of transitions,

$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),

$W : F \rightarrow \{1, 2, 3, \dots\}$ is a weight function,

$M_0 : P \rightarrow \{0, 1, 2, 3, \dots\}$ is the initial marking,

$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$

A Petri net structure $N = (P, T, F, W)$ without any specific initial marking is denoted by N . A Petri net with the given initial marking is denoted by (N, M_0) .

A place p is called an *input place* of a transition t iff there exists a directed arc from p to t . Place p is called an *output place* of transition t iff there exists a directed arc from t to p . We use $\bullet t$ to denote the set of input places for a transition t . The notations $t\bullet$, $\bullet p$ and $p\bullet$ have similar meanings, e.g. $p\bullet$ is the set of transitions that have p as an input place. Note that in the context of workflow processes it makes no sense to have

other weights apart from 1 on the arcs, because places correspond to conditions [van der Aalst, 2].

The state or otherwise, the marking, is the distributions of tokens over places, and will be presented as follows: $2p_1 + 3p_2 + 0p_3 + 1p_4$. This notation means the state with two tokens in place p_1 , three tokens in p_2 , zero tokens in p_3 and one token in p_4 . The number of tokens may change during the execution of the net. Transitions are the dynamic components in a Petri net: they change the state of the net according to the following *firing rule* [Murata, 2]:

- (1) A transition t is said to be *enabled* iff each input place p of t contains at least one token.
- (2) An enabled transition may or may not *fire* (depending on whether or not the event actually takes place).
- (3) If transition t fires, then t *consumes* one token from each input place p of t and *produces* one token for each output place p of t .

A transition without any input place is called a *source transition*, and one without any output place is called a *sink transition*. A pair of a place p and a transition t is called *self-loop* if p is both an input and output place of t . A Petri net is said to be *pure* if it has no self-loops. A Petri net is said to be *ordinary* if all of its arc weights are 1's. For a detailed description on the firing procedure refer to [Murata, 2], [Jensen, 1] as well as to the Appendix.

Below, we mention a number of definitions that form the elementary 'vocabulary' of Petri nets. Hence, given a Petri net (P, T, F) and an initial state M_1 , we have the following notations:

- $M_1 \xrightarrow{t} M_2$: transition t is enabled in state M_1 and firing t in M_1 results in state M_2
- $M_1 \rightarrow M_2$: there is a transition t such that $M_1 \xrightarrow{t} M_2$
- $M_1 \xrightarrow{\sigma} M_n$: the firing sequence $\sigma = t_1 t_2 t_3 \dots t_{n-1}$ leads from state M_1 to state M_n ,
i.e., $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} \dots \xrightarrow{t_{n-1}} M_n$
- $M_1 \xrightarrow{*} M_n$: there is a firing sequence which leads from M_1 to M_n

Definition (8.2): A state M_n is called *reachable* from M_1 (notation $M_1 \rightarrow M_n$) iff there is a firing sequence $\sigma = t_1 t_2 t_3 \dots t_{n-1}$ such that $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} \dots \xrightarrow{t_{n-1}} M_n$.

Few descriptions of Petri nets concentrate simply on the basic definitions. The Computing Surveys tutorials by [Baer, 1] and [Peterson, 1] are probably the best way for continued introduction. Basic definitions on Petri nets can also be found in [Holt *et al.*, 1], [Holt & Comm., 1], [Hack, 1], [Hack, 2], [Keller, 1], [Murata, 1], and [Murata, 2].

8.2.1 Modelling with Petri Nets

Petri nets are used mainly for modelling different kinds of systems with independent components, as computer hardware, computer software, physical or social systems. Petri nets can model the occurrence of various events and activities in a system and in particular the flow of information or other resources within the system. The Petri net view concentrates on two basic concepts: *events* and *conditions*.

Events are actions, which take place in the system. The occurrence of them is controlled by the state of the system. The state of the system can be described as a set of conditions, which may either be true - *hold* - or false - *not hold*. For an event to occur, it may be necessary for certain conditions to hold. These are the *preconditions* of the event. The occurrence of the event may cause the preconditions to cease to hold and may cause other conditions, *post-conditions*, to become true.

In a Petri net, conditions are modelled by places, and events are modelled by transitions. The inputs of a transition are the preconditions of the corresponding event and the outputs are the post-conditions. The occurrence of an event corresponds to the firing of the corresponding transition. The holding of a condition is represented by a token in the place corresponding to the condition. When the transition fires it removes the enabling tokens representing the holding of the preconditions and creates new tokens, which represent the holding of the post-conditions.

One point about Petri nets and the systems, which they can model, is the *parallelism* or *concurrency*. Two events, which are both enabled and do not interact, may occur independently (see Figure 8.1). There is no need to synchronise events,

unless it is required by the underlying system, which is being modelled. When synchronisation is needed, it is easy to model this also. Thus, Petri nets would seem ideal for modelling systems of distributed control with multiple processes executing concurrently in time.

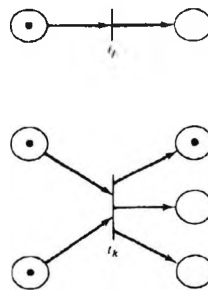


Figure (8.1): Concurrency. These two transitions can fire in any order. ([Peterson, 2])

Another point is when two enabled transitions are in *conflict*. For example t_1, t_2 , in Figure 8.2, are in conflict and only one can fire, since in firing, it removes the token in the shared input and disables the other transition.

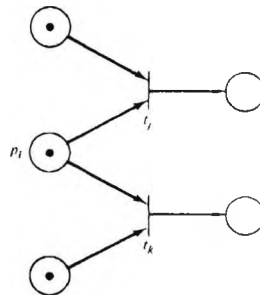


Figure (8.2): Conflict. Transitions t_j and t_k are in conflict since firing either will remove the token from p_i , disabling the other transition ([Peterson, 2])

After modelling systems with Petri nets, it is natural that we want to investigate what we can do with the models. As we already mentioned, a major strength of Petri nets is their support for analysis of many properties and problems associated with concurrent systems. There are two types of properties that can be studied with a Petri net model:

- (i) Those which depend on the initial marking and are called *behavioural properties*
- (ii) Those that are independent of the initial marking and are called *structural properties*.

In this thesis, we are more interested in the behavioural properties and their analysis problems. We use (PN, M) to denote a Petri net PN with an initial state M . Some basic properties for Petri nets are:

Definition (8.3): A Petri net (PN, M) is *live* iff, for every reachable state M' and every transition t there is a state M'' reachable from M' which enables t .

Definition (8.4): A Petri net (PN, M) is *bounded* iff, for every reachable state and every place p the number of tokens in p is bounded.

Definition (8.5): A Petri net is *strongly connected* iff, for every pair of nodes (i.e., places and transitions) x and y , there is a directed path leading from x to y .

Reachability is a fundamental basis for studying the dynamic properties of any system. The firing of an enabled transition will change the token distribution in a net according to the transition rule described in [Murata, 2]. Since places are often used to represent buffers for storing intermediate data, the verification of the boundness property guarantees that there will be no overflows in the buffers, no matter what firing sequence is taken. Moreover, the concept of liveness is closely related to the complete absence of deadlocks in operating systems.

For an overview of the many analysis methods developed for Petri nets, one should refer to [Murata, 2] and [Jen., 1]. These methods (like the coverability tree method, the matrix-equation approach, or the decomposition techniques) can be used to prove properties (safety properties, invariance properties, deadlock, etc.) and to calculate performance measures (response times, waiting times, occupation rates, etc.) In this way, the user is allowed to test and evaluate alternative designs.

8.2.2 Subclasses of Petri nets

There are a number of key structures characterising different subclasses of Petri nets [Murata, 2]:

- (1) A *state machine* is an ordinary Petri net such that each transition t has exactly one input place and exactly one output place, i.e., $|\bullet t| = |t\bullet| = 1$ for all $t \in T$.

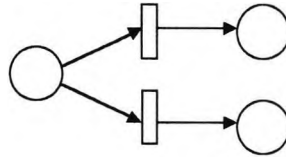


Figure (8.3): An example of a state machine

State machines admit no synchronisation.

- (2) A *marked graph* is an ordinary Petri net such that each place p has exactly one input transition and exactly one output transition, i.e., $|\bullet p| = |p\bullet| = 1$ for all $p \in P$.

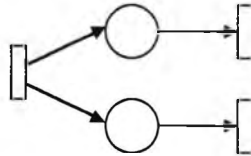


Figure (8.4): An example of a marked graph

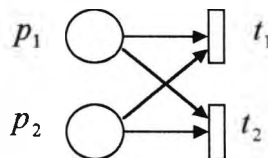
Marked graphs admit no conflicts.

- (3) A *free-choice net* is an ordinary Petri net such that every arc from a place is either a unique outgoing arc or a unique incoming arc to a transition, i.e.,

$$\text{for all } p_1, p_2 \in P, p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow |p_1 \bullet| = |p_2 \bullet| = 1$$

- (4) An *extended free-choice net* is an ordinary Petri net such that

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet = p_2 \bullet \text{ for all } p_1, p_2 \in P.$$

Figure (8.5): An extended free-choice net but not a free-choice one. Verify that $p_1 \bullet = \{t_1, t_2\}$ and

$$p_2 \bullet = \{t_1, t_2\}, p_1 \bullet \cap p_2 \bullet \neq \emptyset, \text{ and } p_1 \bullet = p_2 \bullet, \text{ but } |p_1 \bullet| = |p_2 \bullet| \neq 1.$$

Both free-choice nets and extended free-choice nets have the behavioural property that if t_1 and t_2 share a common input place, then there are no markings for which one is enabled and the other is disabled. In other words, we have a “free-choice” about which transition to fire [Murata, 2]. Hence, free-choice nets admit no confusion.

- (5) Finally, an *asymmetric choice net*, otherwise known as a *simple net*, is an ordinary Petri net such that

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet \subseteq p_2 \bullet \text{ or } p_1 \bullet \supseteq p_2 \bullet \text{ for all } p_1, p_2 \in P.$$

Asymmetric choice nets allow asymmetric confusion.

In this thesis, we will only be dealing with extended free-choice nets, which are a good compromise between expressive power and analysability [van der Aalst, 2].

8.2.3 Examples and Applications

We present as an example of modelling by means of Petri Nets, the producer / consumer problem. Here, the producer process creates objects, which are put in the buffer; the consumer waits until an object is put in the buffer, removes it, and consumes it. This can be modelled as shown in Figure 8.6. The place B represents the buffer; each token represents an item, which has been produced, but not yet consumed [Peterson, 2].

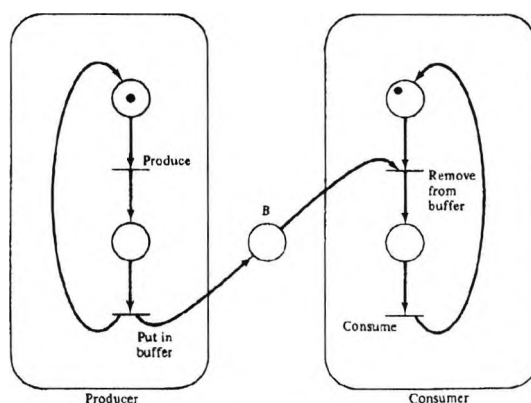


Figure (8.6): The producer / consumer problem modelled as a Petri net ([Peterson, 2])

Some other interesting examples of the application of Petri nets can be found in [Dijk., 1] where we see the dining philosophers problem, and in [Courtois *et al.*, 1], with the readers / writers problem.

Also, the *PERT charts* have long been used in the planning and scheduling of large projects. A PERT chart is a graphical representation of the relationships between the various activities which make up a large project. A project consists of a number of activities; some activities must be completed before other activities can start. The activities are represented graphically by a node; arcs are used to connect activity nodes to show precedence requirements [Peterson, 2].

These charts show the same type of scheduling constraints as Petri nets. We can easily convert a PERT chart to a Petri net. Each activity in a PERT chart is

represented by a place, while the precedence constraints are represented by the transitions. The extra advantage of PERT charts is that they also provide timing information. An example of a PERT chart and its equivalent Petri net is given below (Figure 8.7, 8.8).

Chemical systems are another example of a system, which can be modelled by Petri nets, where chemical equations are modelled by transitions; reactants are modelled by places. [Meld. & Holt, 1] have suggested that legal systems may be modelled by Petri nets, and queuing networks (where the queues would be presented by places and the jobs by tokens) can also be modelled with the use of Petri nets. The list of applications is only limited by the imagination of the modeller and not by the properties of the Petri net model. See also [Murata, 2], [Jen., 1],[van der Aalst, 1], [van der Aalst, 6], [van der Aalst, 7], and [van der Aalst, 3].

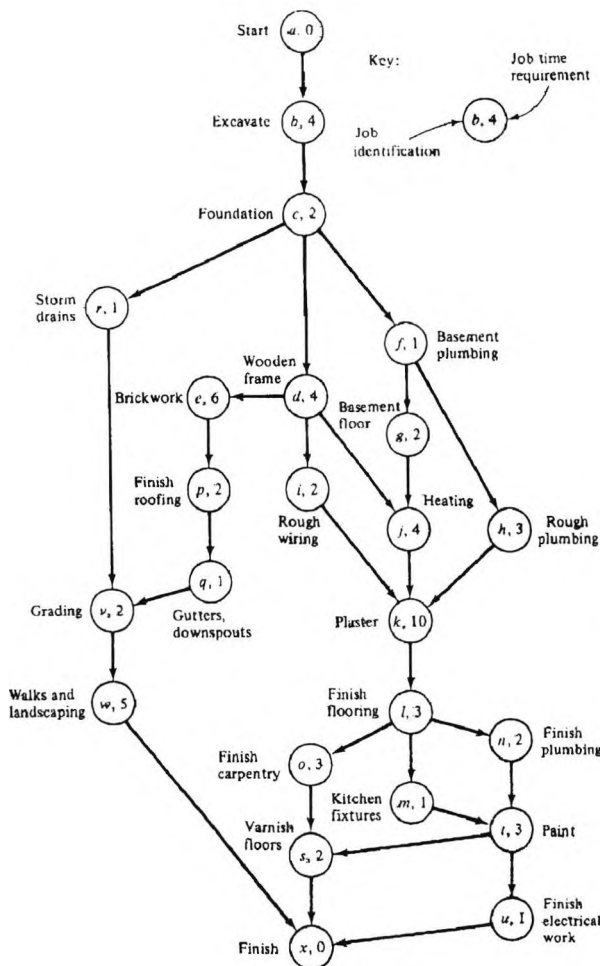


Figure (8.7): A PERT chart of the construction of a house [Levy et al., 1]

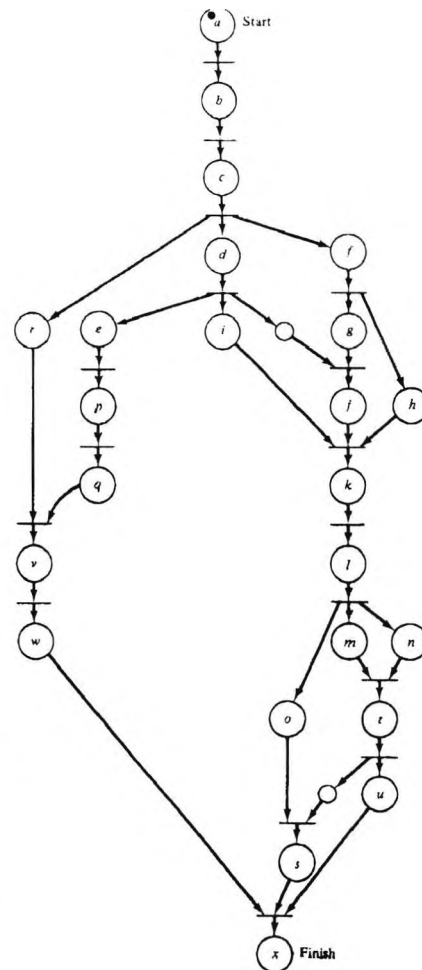


Figure (8.8): A Petri net representation of the PERT chart of figure 8.7 ([Peterson, 2])

8.3 Extensions of Petri Nets

In the case of complex systems, the Petri net graphs become large and therefore difficult to be analysed. In such cases, high-level Petri nets are introduced, such as *predicate/transition nets (Pr T-nets)* or *coloured Petri nets* as the basic modelling tools to express more complicated requirements. The difference between the basic and the high level Petri nets are:

- In coloured Petri nets, tokens have a *type*, which is also called *colour* and carry information to represent structured objects
- Formal expressions or inscriptions containing token occurrences, which are used as inputs or outputs of a transition, are attached to the arcs of the graphs. The information can be inspected and modified when a transition fires, thus imposing conditions on transition firing on the basis of token values.

A special set of coloured Petri nets is the *Updated Petri nets (UPN)*, which model/implement business rules and company policies of manufacturing information systems. Their primitives are an extension of coloured Petri nets and they are used to develop a procedure to automate and formalise the interpretation process from a model to a rule-specification language. An updated Petri net is a directed graph with three types of nodes: *places*, which represent facts of predicates, *primitive transitions*, which represent rules or implications, and *compound transitions*, which represent meta-rules (sub-nets).

A Petri net model enriched by control places, input transitions, output transitions, and related arcs is referred as *Extended Petri net (EPN)*.

Untimed Petri nets do not include any notion of time and are aimed to model only the logical behaviour of systems by describing the causal relations existing between events. The introduction of a timing specification is essential in cases of performance, scheduling or real-time control problems. In such problems the concept of time (time delays) can be introduced either by associating a time interpretation with places and/or with transitions, and thus we are talking about *place-timed Petri nets (p-TPN)* or about *transition-timed Petri nets (t-TPN)* respectively.

If the timing of transitions is not deterministic but characterised by the probability distribution function (PDF) of a random variable, we have the case of *Stochastic Petri nets*. In the case of one-phase stochastically timed transitions and

immediate transitions (firing in zero time), we have *generalised stochastic Petri nets* (GSPN). The use of stochastic Petri nets is limited in practice, by the fact that routing probabilities are not naturally expressible when one-phase firing is assumed for transitions.

Furthermore, there has been a tendency to extend a Petri net model even more to make it easier to use. There have been several types of extensions: [Patil, 1] suggested extension of Petri nets to include *constraints*. A constraint is a set of places. The firing rule is modified to allow a transition to fire if and only if the resulting marking does not have all of the places, which are in a constraint, simultaneously marked. For example, if $\{p_1, p_4\}$ is a constraint set, then either p_1 or p_4 must be empty at all times; if p_1 is marked, then a token cannot be put in p_4 until all tokens in p_1 are removed and vice versa [Peterson, 2].

[Noe, 1] introduced a different extension: the *exclusive-OR transition*. This transition can fire if and only if exactly one of its inputs has tokens and all the others have zero tokens. A similar extension was used by [Baer, 2], who introduced *switches*. A switch is a special transition with a special input called the *switch input* and exactly two outputs (one labelled e for empty and the other labelled f for full). A switch transition fires when it is enabled. When it fires a token is put in the output labelled e if the switch input is empty or a token is put in the output labelled f if the switch input is full. Thus, firing a switch transition will result in either one of two markings, depending on the state of the switch.

Finally, one limitation of classic Petri Nets is their inability to perform zero testing, i.e., the ability to test for the *absence* of tokens in an unbounded place [Moore & Bren., 1], [Patil, 1]. The most common extension that allows for zero testing, is *inhibitor arcs*. An inhibitor arc connects a place p_i to a transition t_j and is usually depicted as a line with a hollow circle at the end in place of the arrowhead. This arc disables a transition t_j when the input place p_i has as many tokens in it as the weight of the inhibitor arc from p_i to t_j (the tokens are not consumed along the inhibitor arc).

More details about the different types of Petri nets are given by [Dicesare *et al.*, 1].

8.4 High-level Petri nets and their role in the modelling of business processes

As we have seen so far, the classical Petri net allows for the modelling of states, events, conditions, synchronisation, parallelism, choice and iteration. Since a business process consists of a partially ordered set of tasks, it is quite easy to map a process onto a Petri net (see [van der Aalst *et al.*, 1]). Hence, tasks are modelled by transitions and precedence relations are modelled by places. After all, there are many reasons for choosing Petri nets as a modelling methodology for business processes. Petri nets have [van der Aalst, 6]:

- **Formal semantics:** classical Petri nets and several enhancements (colour, time, hierarchy) have been defined formally through mathematical semantics. This results to a model of a business process which is unambiguous, i.e., the meaning of each construction is clear, no more multiple interpretations, and whose properties can be identified through an abundance of analysis techniques, i.e., the absence or not of dynamic properties such as deadlocks [van der Aalst, 3]. In this way it is possible to evaluate alternative processes using standard Petri-net-based analysis tools.
- **Graphical nature:** Petri nets are intuitive and easy to learn, promoting the communication between the users.
- **Expressiveness:** Petri nets support all the primitives needed to model a business process.

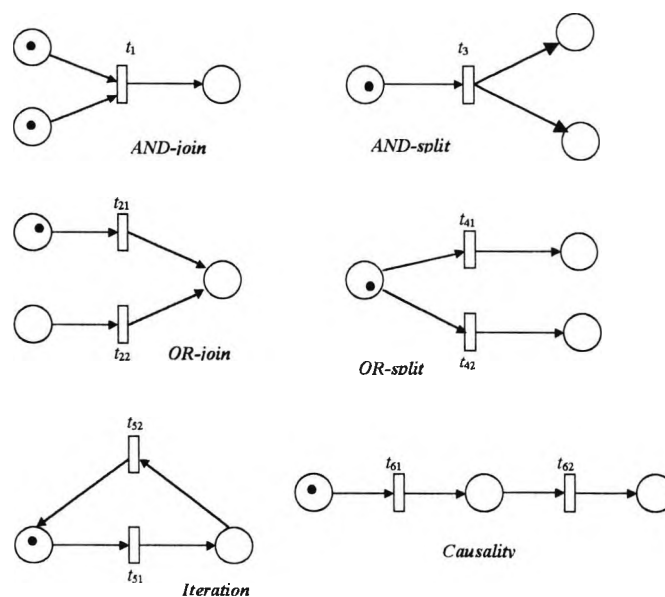


Figure (8.9): The basic activities (primitives) in workflow management ([van der Aalst, 4])

Moreover, as we will further see, the fact that states are represented explicitly, allows for the modelling of milestones and implicit choices.

However, a major weakness of Petri nets is the complexity problem, i.e., Petri-net-based models tend to become too large for analysis even for a modest-size process. Moreover, the classical Petri net does not allow for the modelling of data and time. To solve all these problems, Petri nets have been extended: (a) with colour to model data, (b) with time to model durations, and (c) with hierarchy to structure large models.

- (a) *Extension with colour*: Quite often, tokens represent objects (e.g. goods, humans) in the model, hence, we want to be able to represent the attributes of these objects. If an employee is modelled by a token in the consulting firm's Petri-net-model, we want to represent attributes such as the employee identification number, the name of the employee and his address. In a coloured Petri net, each token has a value often referred as 'colour'. This framework provides the classical Petri net with the modelling power of the conceptual modelling, as it has been derived from the entity-relationship model. Transitions describe the relations between the values of the 'input tokens' and the values of the 'output tokens'. One can also specify preconditions which take the colours of tokens to be consumed into account.
- (b) *Extension with time*: By introducing a timing concept in the classical methodology of Petri nets, we are capable to describe the temporal behaviour of the system. The user can now model durations and delays. Time is associated with tokens, places and/or transitions. The delay of a produced token can be described by a fixed value, an interval or a probability distribution.
- (c) *Extension with hierarchy*: When modelling a real system, models tend to become large and complex. This is the reason we extend Petri nets with a hierarchy construct, called *subsystem* or *subnet*, which is an aggregate of places, transitions, and subsystems. The user can now choose the level of detail for his/her modelled system. At one level, we give a simple description of the process and at another level, we can specify a more detailed behaviour. Hence, it is possible to decompose complex systems into smaller subsystems.

A Petri net extended with colour, time, and hierarchy is called a *high-level Petri net*. In [Jen., 1] one can find interesting examples of high-level Petri nets.

8.4.1 Modelling business processes

Petri nets, which model business processes, have some typical properties [van der Aalst, 2]. First of all, they always have two special places i and o , which correspond to the beginning and termination of the processing of a case respectively. Secondly, a Petri net, which represents a business process, is always an extended free-choice Petri net. Thirdly, for each transition t there should be a directed path from place i to o via t . All these three requirements are included in the following definition:

Definition (8.6): A Petri net $PN=(P, T, F)$ is a *BP-net* (*Business-Procedure net*) if and only if:

- (i) PN has two special places: i and o . Place i is a source place: $\bullet i = \emptyset$. Place o is a sink place: $o \bullet = \emptyset$.
- (ii) PN is a free-choice Petri net.
- (iii) If we add a transition t^* to PN which connects place o with i (i.e., $\bullet t^* = \{o\}$ and $t^* \bullet = \{i\}$), then the resulting Petri net is strongly connected.

A *BP-net* is required to be a free-choice one, because we cannot think of a sensible business process that violates this property, i.e., a process composed of AND-splits, AND-joins, OR-splits and OR-joins is free-choice. One can model parallelism, sequential routing, conditional routing and iteration (the workflow primitives) without violating the free-choice property. Another reason for choosing free-choice nets, is that in a non-free-choice Petri net, the choice of conflicting tasks may be influenced by the order in which the preceding tasks are executed, and the routing of a case should be independent of the order. The last requirement (a Petri net extended with t^* should be strongly connected) states that for each transition t there should be directed path from place i to o via t , making sure that all tasks in the net contribute to the processing of cases. Nevertheless, even if these three requirements are satisfied, it is still possible to construct a process with potential deadlocks and/or livelocks.

There is however a fourth property, which should be satisfied, and that is: For any case, the procedure will eventually terminate and the moment the procedure terminates there is a token in place o and all other places are empty. This property is called the *soundness property* [van der Aalst, 2].

Definition (8.7): A procedure modelled by a *BP-net* $PN=(P, T, F)$ is *sound* if and only if:

(i) For every state M reachable from state i , there exists a firing sequence leading from state M to state o . Formally:

$$(ii) \quad \forall_M (i \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} o)$$

(iii) State o is the only reachable from state i with at least one token in place o . Formally:

$$(iv) \quad \forall_M (i \xrightarrow{*} M \wedge M \geq 0) \Rightarrow (M = o)$$

The soundness property relates to the dynamics of a *BP-net*. The first requirement in definition 8.7 implies that starting from the initial state (state i), it is always possible to reach the state with one token in place o (state o). The second requirement states that the moment a token is put in place o , all the other places should be empty. We can use the coverability graph to decide whether a *BP-net* is sound, but this decision is really difficult to be made when business processes become complex. In [van der Aalst, 5] this soundness property is defined formally and a technique is presented to verify this property in polynomial time.

8.4.2 Triggering

In real processes, we often see a task being executed for a specific case, but this does not imply that the task is executed directly. For example, if a task is to be executed by an employee, then the employee has to be available to execute the task. If he is sick or away on holidays, the task will not take place. One of the main advantages of Petri nets, is that they allow a distinction between the *enabling* of a task and the *execution* of a task. Therefore, the concept of *triggering* is introduced. Trigger is an external condition which leads to the execution of an enabled task [van der Aalst, 6]. The task instance is executed the moment the task instance is triggered. A task instance can only be triggered if the corresponding case is in a state which enables the execution of the task. Tasks are divided into four different types depending on the kind of triggering that sets them into action:

1. *Automatic*: a task is triggered the moment it is enabled. This kind of triggering is used for tasks, which are executed by an application that does not require human interaction.
2. *User*: a task is triggered by a human participant, i.e., it is entirely up to the user to select an enabled task to take place.
3. *Message*: an external event triggers an enabled task instance. Examples of such external events are telephone-calls, fax messages, e-mails, and so on.
4. *Time*: a clock triggers the enabled task instance, i.e., the task is executed at a predefined time. For example, the task 'promote to consultant' is triggered if a case is trapped in a specific state for more than 3 years.

Only for automatic tasks do the enabling and the actual start of the execution coincide. In order to indicate the different types of tasks, users can use the symbols shown below.

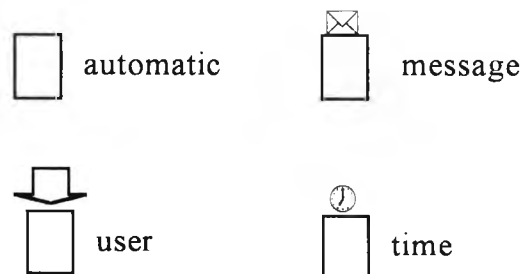


Figure (8.10): The four types of triggering ([van der Aalst, 6])

In order to present the use of these symbols in the modelling of a process, we give an example of an explicit representation of triggers, states, and the movement choice in Figure 8.11. Here, we see that in both process definitions the execution of task *A* is followed by the execution of *B* or *C*. In the first process definition, the moment of choice between the two tasks *B* and *C* is as late as possible. After the execution of *A*, tasks *B* and *C* compete with one another against the time (the case of two competitive tasks, i.e., both are enabled and only one of them may be executed). If the external message required for task *C* arrives before someone starts executing task *B*, then *C* is executed, otherwise *B*. In the second process definition, the choice is made after the execution of *A*. If task *B* is selected, then the arrival of the external message has no influence. If *C* is selected, then *B* cannot be used to bypass *C*.

To model this situation, we cannot use an event-based description such as the one the Role-Activity diagrams (RADs) provide. The choice to do task *B* or *C* is not

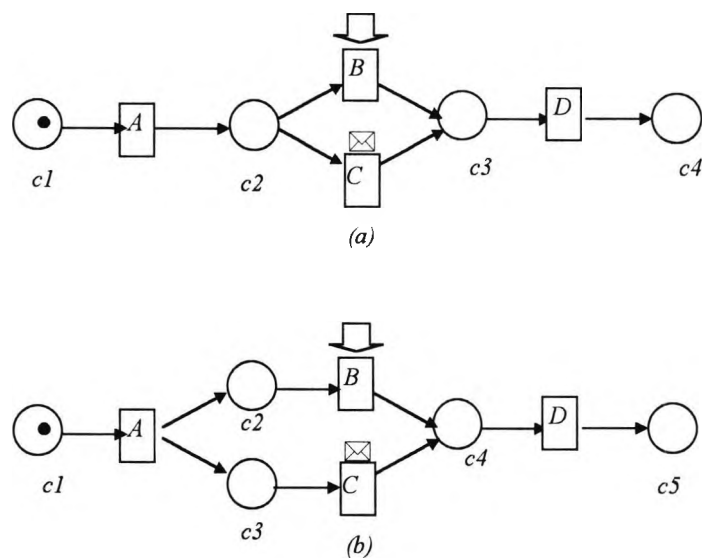


Figure (8.11): An example indicating the triggering mechanism ([van der Aalst, 3])

made during the execution of task A (see Figure 8.11a). The choice is implicitly made by the environment of the business process while the corresponding case marks place $c2$. Petri nets allow for the explicit representation of states in a very elegant manner.

8.5 Analysis of Petri nets

As we have seen Petri nets are capable of modelling a large variety of systems. But modelling by itself is of little use. It is necessary to analyse the modelled system in order to understand its behaviour better. A business process, that has 'mistakes' in its design, may prove catastrophic when implemented, e.g., high throughput times and low service levels. Hence, it is crucial that the business process is analysed before put into production. The analysis techniques that are used for this purpose are divided into the following categories:

1. Validation: testing whether the process behaves as expected.
2. Verification: investigating the correctness of the process.
3. Performance analysis: investigating the ability of the process to meet requirements with respect to throughput time, service and resources management.

Very briefly we can say that validation analysis is implemented through interactive simulation, where the user has the chance to test the behaviour of the system when a number of fictitious cases are processed. This type of analysis does not allow

properties to be proved, even if they might be of great help in understanding the modelled system. It is useful when we wish to know the response of the system described with a net in an environment, which is also defined by simulation.

Verification and performance analysis are implemented using certain advanced analysis techniques. For example, linear algebraic techniques (matrix-equation approach) can be used to verify many properties, such as place invariants, transition invariants, and non-reachability. Coverability graph analysis (representation of the net marking and the transition firings), model checking, and transformation or reduction techniques can be used to analyse the dynamic behaviour of a Petri net [Murata, 2], [Jen., 1]. Simulation and Markov-chain analysis can be used for performance evaluation. We will not elaborate further on the analysis techniques, because this is beyond the scope of this thesis. One point is crucial to understand here though. That all these mathematical analysis techniques make Petri nets a solver independent medium [van der Aalst, 6] between the design of the business process and the analysis of the process, and place them high in the list of all the methodologies we have investigated in this thesis; for being the most advanced, structured, and mathematically formulated modelling method.

8.6 Tools

When evaluating a business process, it is very important to have estimates of the key performance measures such as response times, waiting times and occupation rates of resources. Most of the Petri-net-based simulation tools provide a simulation facility that evaluates the performance of a particular procedure. At the moment, there are quite a few workflow management products that use Petri nets as a design language. Some of them are:

1. *ALPHA/Sim* (Alphatech, Inc.) a general purpose, graphical, discrete-event simulation tool that allows one to construct simulation models graphically using icons and data-form windows. The class of Petri Nets modelled are stochastic, timed and coloured nets. *ALPHA/Sim* can be used to perform a range of analyses, including timing and sizing, workload, queuing, and failure analyses, in support of system design and evaluation. Some areas of application include: re-engineering of business processes, design and analysis of manufacturing systems, design and

evaluation of service delivery facilities (such as banks, hospitals), design and evaluation of hardware/software systems, and so on. *ALPHA/Sim* will be used in this thesis to simulate the consulting firm's problem.

2. *COSA* (Software Ley/COSA Solutions, Pullheim, Germany) which consists of seven interrelating components, and supports design of workflow process definitions, classification of resources in terms of roles and organisational units, a simulation tool, a workflow client application that offers work items to the end user, and an administration and monitoring tool which can be used to handle abnormalities, execute changes, detect problems, and collect management information [van der Aalst, 6]. Although *COSA* is good in modelling complex processes, it does not support verification, it is difficult to use, and it does not allow detailed simulation.
3. *DesignCPN* (Meta Software) is a hierarchical Petri-net-based simulation tool, which provides the user with a graphical interface to create, modify and simulate high-level Petri nets.
4. *ExSpect* (Bakkenist Management Consultants, Diemen, the Netherlands) can be used to model and simulate processes modelled in terms of high-level Petri nets. The language *ExSpect* consists of two parts: a functional part and a dynamic part. The functional part is used to define types and functions needed to describe operations on the value of a token. In the dynamic part, a network of transitions and places, as well as their interaction, are specified. During the simulation of a process with *ExSpect*, the process is animated and key performance indicators (e.g. average throughput time of a project and occupation rate) are measured. The user is also capable to intervene, while the simulation is running, by adding tokens to places or changing the values of the tokens, hence changing completely the final outcome of the process and its behaviour. As a result, a number of diverse scenarios are evaluated and the tool supports also the modelling of sudden events or changes (i.e., adding tokens results in the change of the system's state). Both *ExSpect* and *DesignCPN* provide analysis tools and reporting facilities.
5. *INCOME* (Promatis Informatik GmbH & Co. KG) provides upper and lower CASE tool (Computer Aided Software Engineering) based on semantic data modelling and Petri nets.
6. *PACE* (Grossenbacher Elektronik AG) includes a Petri net tool plus simulation.
7. *Process Weaver* (Cap Gemini Innovation) supports Petri-net-based process modelling.

8. *ProModel 2.0* (ProModel Corp.) is a tool for process modelling and simulation. The last four software tools (*INCOME*, *PACE*, *Process Weaver*, and *ProModel 2.0*) can be used to document an existing process and analyse it, as well as, to define and analyse new process concepts. Additionally, they can support the construction of prototypes and detailed designs of new processes.
9. *PROTOS* (Pallas Athena, Plasmolen, the Netherlands) is mainly a Petri-net-based Business Process Reengineering tool. It is easy to use and can be useful in modelling and analysing business processes.
10. *Woflan* (Eindhoven University of Technology, the Netherlands) is perhaps the only Petri-net-based workflow analyser that can also be used to verify the correctness of a workflow. It makes use of standard Petri net analysis techniques and stresses the importance of the soundness property. If the process definition is not sound, then the user is guided in detecting the source of the error and support is given to repair the error. *Woflan* detects constructs, which violate the free-choice property, calculates invariants, detects unbounded places and reports dead transitions.

The list of software tools, based on Petri Nets and developed by both the commercial and academic worlds, grows even more. Here, only a selection of them is reported. Detailed descriptions and contact information of these and other Petri Net tools can be found on the Internet. The Centre de Recherche Informatique de Montreal (CRIM, Web address: http://www.crim.ca/Domaines_Services/GL/PETRI/README.html) and the Department of Computer Science at the University of Denmark (DAIMI, Web address: <http://www.daimi.au.dk/PetriNets/>) maintain World Wide Web sites devoted to Petri Nets and their tools.

8.7 Drawing the Petri net model for the consulting firm's problem

In this part of the chapter, we focus on the modelling of the Human Resources Management System of a consultancy firm. This time, the research vehicle for the study of the dynamics of the system is a Petri Net model developed using *ALPHA/Sim*'s platform. A detailed discussion of the model's structure as well as of its behaviour is provided.

ALPHA/Sim is a discrete-event simulation tool, where simulation models are constructed graphically using icons and data-form windows. These can be constructed

hierarchically, allowing one to build models from the top down, from the bottom up, or both. The tool also provides a rich assortment of built-in mathematical functions and statistical collection and reporting capabilities. With ALPHA/Sim, simulation can be used to evaluate the design of a new system, or to predict the impact of changes to an existing system. The graphical development environment means that the user never has to see or write a line of code.

Model development is accomplished through ALPHA/Sim's graphical user interface. The user creates a model by arranging icons from a palette on the drawing area. Each icon represents one of the fundamental ALPHA/Sim building blocks (i.e., places, transitions, arcs, and boxes). Double-clicking on an icon brings up a data-entry form that allows one to define timing rules, decision logic, attribute value assignments, and other parameters that define the characteristics of the system. Hence, ALPHA/Sim is based fully on the extended Petri Nets theory that incorporates timing rules at transitions, decision rules at places, attributes (colours) on tokens, additional arc types, and box nodes. More analytically, all the above extend the modelling power of Petri Nets as follows:

- Timing rules at transitions: There are three types of timing rules – (i) deterministic, where time of process is fixed at one value, (ii) stochastic, where time of process is randomly distributed according to user-specified probability distribution, and (iii) constructed, where time of process is based on a user-specified function. These represent the time required to complete an activity and are used for system performance analysis.
- Decision rules at places: There are three types of decision rules – (i) priority, where tokens flow along the arc with the highest priority, (ii) probability, where tokens flow along the arcs according to assigned probabilities (assuming all firing rules are met), and (iii) constructed, where tokens flow along arcs based on a user constructed rule. These rules resolve cases of conflicting transitions and govern the flow of tokens out of places.
- Attributes on tokens: These extend the modelling power of Petri Nets by making description and analysis of a system more compact and manageable.
- Arc types: There are three types of arcs – (i) the standard arc, (ii) the enable arc, that is used when there is no resource constraint, and (iii) the inhibit arc, which is used

when the presence of an object or condition prevents a process or event from occurring.

- Box nodes: They are used to group or cluster model segments that relate to particular subsystems and functions.

Using the description of the problem as stated in paragraph 2.7.1, the construction of our model proceeds through a bottom-up approach. Here, the basic transitions (events) are identified and graphically represented as rectangles. Places (locations of states) are soon added to the model, and the reader easily distinguishes the four main 'buffers', i.e., the Juniors, the Consultants, the Seniors and the External consultants (see Figure 8.12). Standard arcs are used to represent the paths of objects through the system. Two important places are defined in the model, and these are: (i) Projects, where a 'machine' generating tokens, i.e., man-hours, feeds the system, and (ii) New_con, where a queue of ready-to-be-hired consultants is slowly inserted into the system. The model assumes equal probability for projects to be undertaken by the four different types of consultants, though their processing differs due to the productivity factor; junior consultants are only half as productive as the other three types of consultants. Initial population for all the places, representing consultants, is defined – the number of Juniors is 20, the number of Consultants is 10, Seniors start with 5 people, and, finally the number of Externals is 25. Furthermore, it is assumed that the consulting company hires 15 people on average per year. As it is shown in Figure 8.12, the whole hiring-work-transfer-attrition process for the four different types of consultants is explicitly described.

Using ALPHA/Sim's modelling capability, Figure 8.12 can be further manipulated so as to construct a concise model, as the one shown in Figure 8.13. Here, the two main places, i.e., Projects and New_con, that drive the process remain, but all the unnecessary details of projects undertaken by a specific type of consultant, as well as the process of their attrition, are clustered in Box nodes, named as Type_of_Consultant_working. Transitions Prom_con_to_sen and Prom_jun_to_con are still visible in this representation of the system. Using the clustering mechanism of ALPHA/Sim once more, Figure 8.13 reduces to Figure 8.14, simply informing the reader that the company undertakes projects and hires people to cope with the on-going demand.

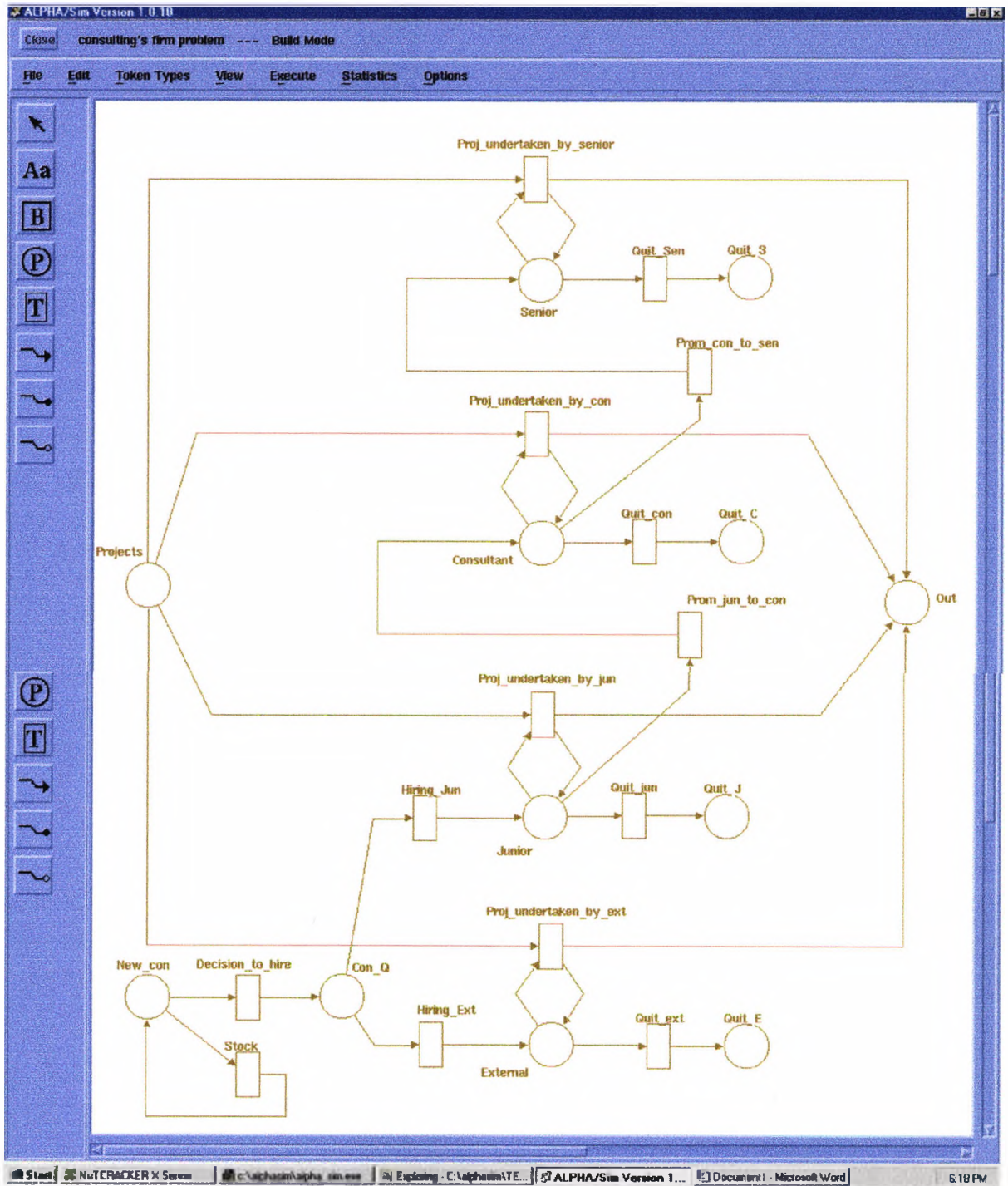


Figure (8.12): The complete Petri Net model of the consulting firm's problem

The arcs connecting the boxes represent information that is passed between the different processes; the nature of each underlying process is detailed inside the boxes.

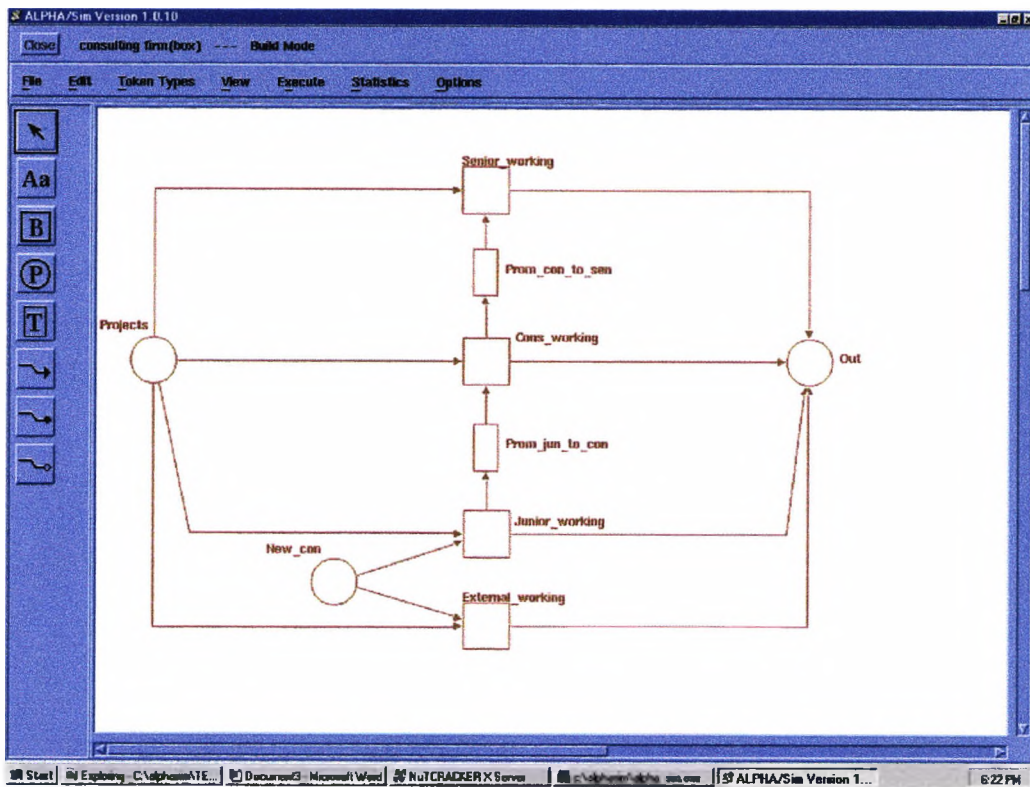


Figure (8.13): Introducing clustering in Figure 8.12

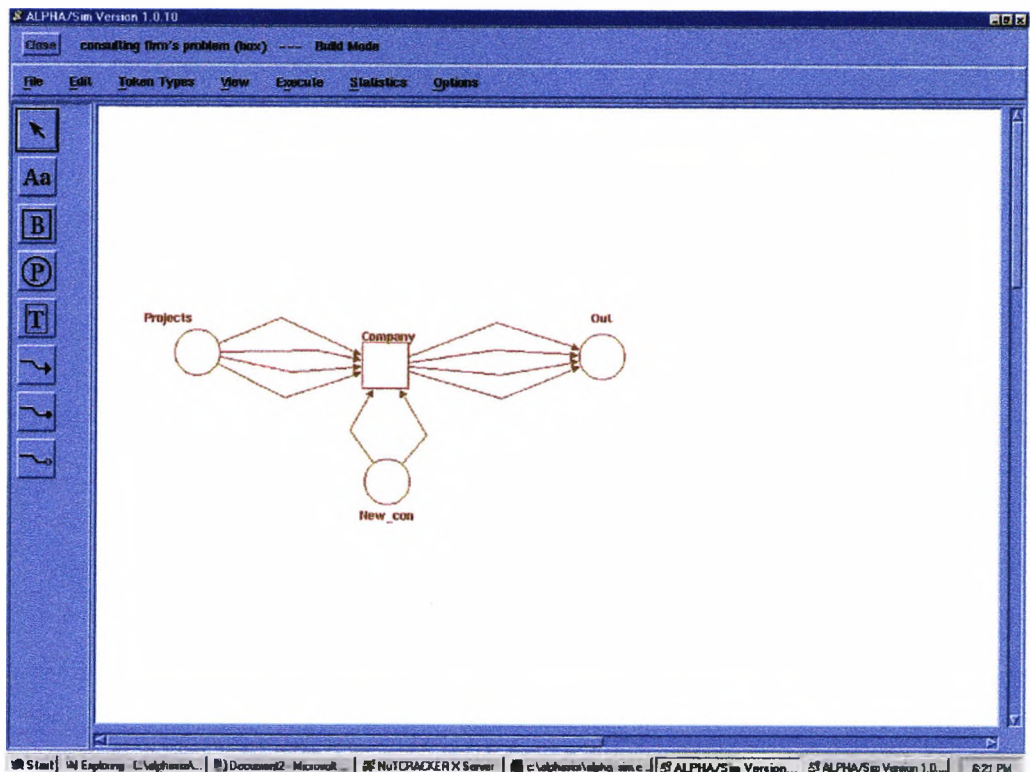


Figure (8.14): General description of the consulting firm's problem

ALPHA/Sim also offers another interesting view of the model, such as the one shown in Figure 8.15. In addition to displaying the overall hierarchical structure of a model, trees can be used to navigate through the model. All the above different representations of the problem make it easier and more intuitive to understand the system. Visual display of the model's logic eases construction, debugging and validation.

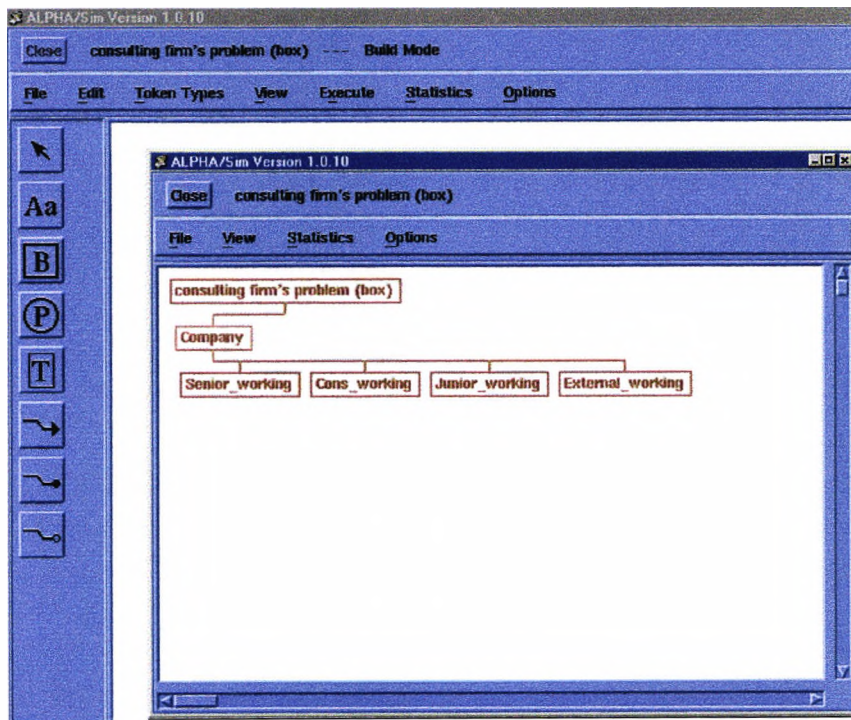


Figure (8.15): Drawing the tree of the system

Soon after the appropriate data (timing rules, decision logic, attribute value assignments, and other parameters that define the characteristics of the system) is inserted in the data-entry forms of the various building blocks, simulation is used to evaluate the performance of the system throughout a ten-year period. As already mentioned, the place called Projects contains a number of projects that have been transformed into man-hour units irrespectively of delays, requirements, or duration. Hence, a regular flow of projects that steadily increases throughout the ten-year period is assumed. Furthermore, it is assumed that the company hires people according to a steady flow of 15 hirings per year. These are distributed among junior and external consultants according to the 10:3 ratio.

When carrying on with the simulation, the ALPHA/*Sim* model provides valuable annual information on the number of employees that the company has at the respective levels of consultancy, as well as on the number of man-hours the company is capable of dealing with, i.e., projects the company is able to undertake. For example, during one simulation run, the following results were obtained: At the end of the ten-year period, the company's workforce increased from 60 people to 107, distributed as follows: External consultants: 47, Junior consultants: 28, Consultants: 21, and Senior consultants: 11. It is fairly easy from this point onwards, for someone to try different hiring patterns so as to closely match the required/expected demand for projects. The hiring pattern that has been followed in this example (stochastic with an average of 15 hirings per year), can easily be modified so as to take different average values (hirings per year) to the extent of creating a completely custom-made hiring function. Furthermore, ALPHA/*Sim* offers one the ability to make modifications easily, by adding/ deleting icons to explore alternative structures, or by entering new values to explore parametric variations.

Although Petri Nets have proven a useful methodology for modelling flows of resources and information effectively, one cannot fail to observe the inherent inability of Petri Nets to provide optimisation runs, and thus, optimal solutions. Instead, the user has to compromise with a trial-and-error logic, i.e., individual simulation runs for each setup under test.

8.8 General Comments on High-Level Petri nets

Petri nets are well suited to model flows of goods, resources, and information in a unifying way. Modelling these flows by tokens appears very natural. A place either represents a medium through which something is sent or a temporary storage space. The fact that flows are represented graphically (some PN tools even allow the animation of tokens while the simulation is running) makes the overall structure extremely comprehensible, explicit, and, at the end of the day, very appealing. This description of a modelled system can be used as a specification (of a system which we want to construct) or as a presentation (of a system which we want to explain to other people).

High-level Petri nets is a modelling language developed for systems in which communication, synchronisation and resource sharing play an important role. They are

characterised by some very important properties, thus, they are extremely valuable as a modelling language for the design, specification and analysis of many different types of systems. Their main properties are their graphical nature and their formal semantics. Business logic can be represented by a formal as well as a graphical language. Hence, the basic structure of a complex high-level Petri net is easily explained, i.e., how the individual processes interact with each other, and the formal description of the net resolves any ambiguity and conflict over the interpretation of common workflow procedures. This last property is what makes Petri nets a “high-level” modelling methodology, when compared with the other methods mentioned in the previous chapters. It is the only methodology, which supports a well-defined syntax and semantics, forming in this way the foundation of the formal analysis methods. As we have seen, there is an abundance of formal analysis techniques by which properties of high-level Petri nets can be proved (reachability graphs, interpretation of system invariants, reductions, and checking for structural properties).

One very important aspect of Petri nets is that, in contrast to other process modelling techniques (Role Activity Diagrams, dataflow diagrams, for example), the state of a case can be modelled explicitly in a Petri net. Process modelling techniques are event-based, i.e., transitions are modelled explicitly and the states between subsequent transitions are suppressed. The advantages of using a state-based description¹ seem to be of utmost importance. This kind of description allows for a clear distinction between the enabling of a task and the execution of the task. We have shown, through the concept of triggering, that the enabling of the task does not imply that the task will be executed immediately. A task instance can only be triggered if the corresponding case is in a state that enables the execution of the task. In a RAD for example, we assume that the task is executed (the activity starts) immediately, as soon as the role instance is in the triggering condition. But in a Petri net, tasks might not be performed immediately after a task has been completed. Their start could depend on the state of the system. Through a state-based description, one can also model the possibility of competitive tasks (Figure 8.11a) or the case of a withdrawn case, by simply removing all the tokens and triggers that correspond to the cancelled case.

¹ The term state-based simply denotes that states are modelled explicitly. A state-based description also incorporates state transitions, i.e., events.

Furthermore, high-level Petri nets allow for the depiction of the environment, the enabling conditions and the effects of an action on a single sheet of paper [Jen., 1]. Many other graphical description languages (for example, Soft Systems and System Dynamics) work with graphs that only describe the environment of an action; the detailed behaviour is specified separately. All this information is then embedded in a variety of software tools that support the drawing, simulation and formal analysis of the nets. Through simulation, users gain insight and detailed knowledge of a system, which is often as important as the results that the users get from the actual simulation runs.

Last but not least, another interesting property of high-level Petri nets, is their hierarchical nature. This means that one can construct a large Timed Coloured Petri net by relating smaller nets to each other, in a well-defined way, thus, one is able to structure large specifications. Hierarchy constructs were also supported in the System Dynamics methodology.

8.9 Petri Nets and the other methodologies

So far, the various methodologies for modelling business processes have been presented extensively. The whole research effort started from the identification of the most elementary activities deployed in the early phases of business modelling, and reached the point where well-structured, well-defined, and rather formal models of business processes could be developed. As Figure 8.16 indicates, the Soft Systems approach forms, historically and conceptually speaking, the initial attempt of the model developer to gather data and structural information for the process under observation. The essence of this effort is to identify one or more problem areas in a system and then to model only those business processes that are relevant to that problem area, i.e., to create the "relevant" system. Elicitation of knowledge is based mainly on the goals and objectives of the modelling effort, and on these will all the methodologies mentioned in this thesis, rely to drive the development of business models. The process, suggested by the Soft Systems approach, is iterative: the model is refined again and again, until all participants agree that the model represents the ideal processes of this soft system. The aim is to tackle both the intellectual problems of interpretation, analysis and synthesis involved in conceiving ideas for change. The model produced, at this phase, is simply the product of combining graphics and natural language, in order to organise and depict

all relevant information. It is a static and purely qualitative description of the business process.

Conceptual modelling, as been used by the fields of Artificial Intelligence and Software Engineering, adds to the representation of a process by creating semantic models that can suppress all the irrelevant details and emphasise only the essential ones. The abstraction mechanisms used are aggregation, grouping, classification, and generalisation.

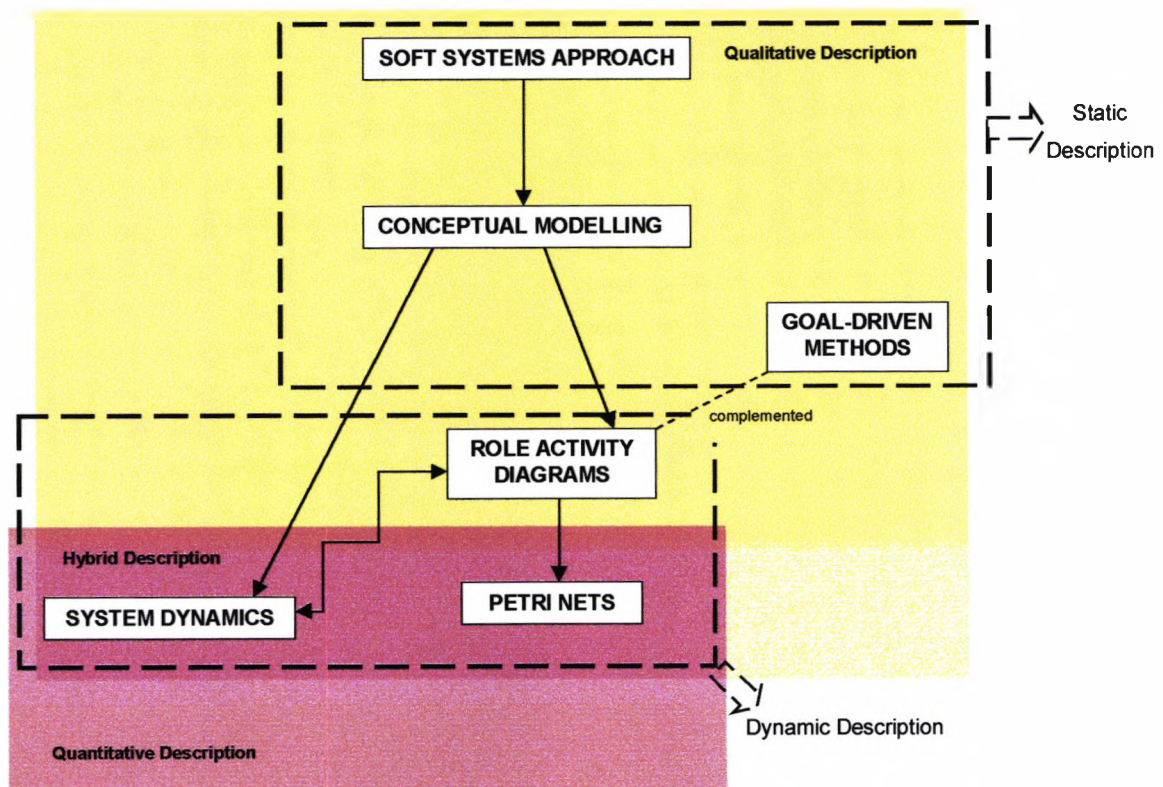


Figure (8.16): Classification of the various methodologies according to the nature of model description

At this phase of the development, the main entities, objects, as well as their interrelationships in a process are identified. The system developer is now enabled to reason about the business process, communicate this understanding to end-users, and specify the allowable structures and transitions on the information base. However, neither Soft Systems approach nor Conceptual modelling provides a modelling support for the behaviour of systems. Additional research has led to the extension of the models produced so far, so as to create behavioural models. These provide a conceptual

formalism for expressing how and when changes occur to entities and relate with one another.

The modelling process now moves to the dynamic description of the system, but once again the path to follow differs depending on the type of information one wishes to extract from his/her model. For an intuitively easy to read representation of the process, where entities are again the subject matter of a process, and the emphasis lies on the description of what people *do*, rather than on what people do it with, the Role Activity Diagrams (RADs) methodology suits its purpose (Figure 8.16). RADs are flexible in modelling not very neat hierarchies of activities (an issue that Conceptual modelling failed to address), and, thus, providing the end user with a very accurate qualitative model, where its behaviour is studied through the movement of a simple token around the numerous activities. A richer representation of workflow models could benefit from modelling the intentional dimension of organisational work. Goal-driven methodologies introduce a modelling framework that explicitly represents the goals behind every business activity. By adding a *goal* attribute to activities, the modeller allows the activity to be selected as a candidate for meeting the final goal. The distinctive characteristics of this approach of reasoning are the use of methods and correlation rules that produce contributions towards goals.

System Dynamics (SD), on the other hand, are relevant to dealing with problems of dynamic nature, i.e., with the overall behaviour of the system and how it can influence its own evolution into the future. Nevertheless, System Dynamics could easily complement other management science approaches, which are more close to static problems, as RADs are for example. If the aim of the modelling process is to find a reduced time-to-process, Systems Dynamics can help. A process is now seen as a set of flows of material between stocks. These flows form a network that can include feedback loops and alternative paths of materials. SD models come into two equivalent forms – the influence diagram and the equations form- making them easy to communicate (this provides a qualitative description of the model). There is a formalism when building models and that makes SD methodology a more rigorous one. Software tools for simulating SD models allow flow rates to be made dependent on stock levels, on external variables, on the passage of time, on the date or time, and even each other; in this way the analyst can determine the cycle time (this provides a quantitative description of the model). This is a much richer representation than the one offered from RADs, especially when trying to understand the variability of time a process takes.

A RAD can easily be converted into an SD model, making it possible to deal with factors which are strictly outside the sort of model a RAD produces. Role Activity Diagrams and System Dynamics are, thus, two complementary tools for the model developer to use, despite some apparent overlap between them.

Petri Nets provide a very accurate conceptual formalism for expressing how and when changes occur to entities. As with RADs, an occurrence of an external event triggers the execution of a number of processes, hence reflects the effects of the event on the system and produces a state transition. RADs also seem to encompass some of the logic of Petri Nets, by acknowledging the position of tokens on a RAD as an indication of the marking of the role instance. However, Petri nets are particularly suited to describe an intended system behaviour where precise process synchronisation is required. Particularly, High-level Petri nets succeed in encompassing both qualitative and quantitative information, thus allowing the development of business models of 'hybrid' nature. Also as a mathematical tool, Petri Nets can be used to set up state equations, algebraic equations, and other mathematical models, as well as simulation models. Their formalism offers a plethora of analysis tools that test whether the process behaves as expected (validation), check the correctness of the process (verification), and investigate the ability of the process to meet requirements with respect to throughput time, service and resources management (performance analysis).

Petri Nets are extremely good at modelling well-structured processes so as to detect – using all the abundance of analysis tools – delays, throughput times, bottlenecks and problems in design in general. In this thesis, Petri nets are considered as the most advanced modelling technique for business processes, in terms of formalism and of tools for behavioural and structural analysis. They provide the analyst with helpful hints for the re-engineering of a process. However, the nature of business processes is diverse and complicated. When the description of a problem is vague and the relationships between the various entities and processes are not very well defined (as it often happens in the case of business process modelling), System Dynamics can overcome fuzziness and propose a better interlinking of the entities and processes involved. This is because System Dynamics incorporate optimisation logic where Petri Nets fail to address. Nevertheless, research has shown that optimisation by itself requires different tools and methods so as to complement the modelling power of System Dynamics and Petri Nets. There is no such thing as a total modelling solution. Each model, deriving from a different modelling methodology, has its own things to tell

end-users, hence the analyst should use them alternately to satisfy the needs of the client.

8.10 Conclusion

Petri nets are designed to model systems with interacting concurrent components. Mostly, they are suited to model and analyse discrete event dynamics systems, which exhibit parallel evolutions and whose behaviour is characterised by synchronisation and sharing phenomena. Using Petri nets to represent concurrency, conflict and mutual exclusion in a system is convenient and straightforward, but problems arise when the system to be modelled is very complex. In these cases the final models will be large and difficult to analyse. Thus, various extensions of Petri nets are proposed and this thesis considers the use of High-level Petri nets, i.e., hierarchical, timed, coloured Petri nets, as the appropriate extension when modelling business processes.

Modelling a system using Petri Nets has many advantages:

- the overall behaviour is easier to understand due to the graphical and precise nature of the representation scheme; Petri nets are the only modelling technique which supports a well-defined syntax and semantics, forming in this way the foundation of the formal analysis methods.
- they can be interactively executed and analysed using the Petri Net theory; there is an abundance of formal analysis techniques by which properties of high-level Petri nets can be proved (reachability graphs, interpretation of system invariants, reductions, and checking for structural properties).
- they can be synthesised using bottom-up and top-down approaches.

Petri nets are used mainly for modelling different kinds of systems with independent components, as computer hardware, computer software, physical or social systems. They prove out to be a suitable methodology for modelling very well structured processes, but research shows that Petri Nets need to be complemented by System Dynamics and optimisation techniques in order to cope with more 'fuzzy' structures.

Chapter 9

CONCLUSIONS

9. CONCLUSIONS – (Ideas for future work)

9.1 General approach of the thesis

In this thesis, a thorough investigation of the methodologies and approaches for the modelling of business processes has been performed. This has revealed a number of tools and modelling aspects that need to be taken into consideration, as well as a large number of open issues. The important role that business processes play nowadays has been explicitly stressed out in the thesis, in the sense that the set of business processes in an organisation essentially defines what the organisation does. All organisations depend for their competitiveness on the efficiency with which their business processes operate. In addition, using a process view highlights the fact that there are compelling reasons for moving from traditional, hierarchical, organisational structures to those where the importance of business processes is recognised. These reasons include:

- the ability to deliver a better service to customers
- the creation of increased flexibility of operations
- improved job satisfaction by providing a goal-oriented focus and enabling business transformation to flatter more effective structures.

[Stev., 1] quotes: "...A parallel may be drawn with science and engineering. In designing a new chemical plant or a passenger jetliner, for example, engineers recognise the complexity of the task; design cannot be left to experience or rules of the thumb. Before even a prototype is built, computer models are used to *simulate* system behaviour over a wide range of conditions. Managers and engineers work together to analyse the outcomes of simulations and to discuss design implications. Modifications are proposed, more simulations are conducted, designs are progressively improved. The design team learns about relationships between system *structure* and system *behaviour*...". So far, the cluster of problems overshadowing the modelling of business processes has been seen as one problem, whose solution was narrowed down to building a 'nice' picture of the problem, and then decision-making was performed, based wholly on heuristics and intuition of the people involved. Here, the problem of moving from a simple verbal description of the process to its robust representation, based on a formal, mathematical notation has been addressed.

Particularly, this research work provided a natural path for moving from a conceptual level to a formal one, enabling decision-making and driving the analysis

away from experience and intuition. Particular emphasis is given on the objectives of the modelling exercise, which after all, define the description of the process, i.e., static or dynamic, quantitative or qualitative description (Figure 9.1). The aim is to evaluate various modelling techniques and then, make use of them at different stages of analysis. The classification of these modelling techniques is made in Figure 9.1. The figure clarifies the process of how to build models of business processes. Historically- and conceptually-speaking, Soft Systems form the initial attempt of the model developer to gather data and structural information for the process under observation. Elicitation of knowledge is based mainly on the goals and objectives of the modelling effort, and on these will all the methodologies mentioned in this thesis, rely to drive the development of business models. Conceptual Modelling adds to the representation of a process by creating semantic models that can suppress all the irrelevant details and emphasise only the essential ones. The abstraction mechanisms used are aggregation, grouping, classification, and generalisation.

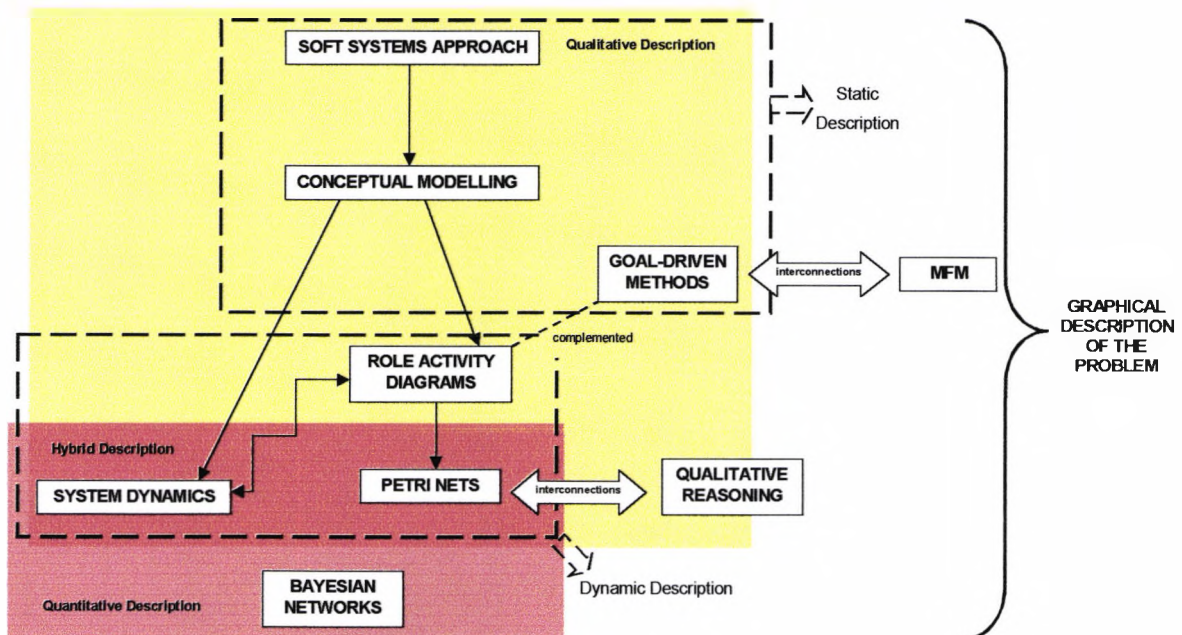


Figure (9.1): Classification of the various methodologies according to the nature of model description and in the wider effort of deployment of conceptual and quantitative information

However, neither Soft Systems approach nor Conceptual modelling provides a modelling support for the behaviour of systems. Additional modelling techniques provide behavioural models, that are able to express how and when changes occur to

entities and relate with one another. Role Activity Diagrams produce extremely detailed qualitative representations of the process and enhance communication. Petri Nets are very good at modelling well-structured processes, where as System Dynamics can overcome fuzziness and propose a better interlinking of the entities and processes involved, when the description of the problem is vague. The choice of any of the above mentioned techniques is instigated by the amount of information available and the complexity level of the problem.

The evaluation of the various modelling techniques is accomplished through their application to a very specific problem, the management of the human resources in a consulting company. This is a problem of resource allocation so that the company is able to meet the increasing demand for projects. The system is of dynamic nature, based on an information-feedback system. There are time delays between the creation of policies, decisions and actions. In this thesis, the development of such a model has been introduced and the application of the proposed framework (Figure 9.1) has provided us with the understanding of requirements for modelling analysis and decision-making for business processes. The advantages and the limitations of the alternative modelling techniques have been addressed and the type of models that these techniques are suitable to provide is discussed. This exploratory model can be extended to help organisations not only to forecast the ramifications of policy and environmental changes, but also to give decision-makers information about the possible effects of policy choices.

In addition, an alternative philosophy, in the overall design of business processes has been presented, in terms of use of control and “more” engineering notions. We have introduced a conceptual abstract and control theoretic framework for general systems, which is, nevertheless, suitable for discussing many issues arising in the analysis and design of business processes. Control theory concepts have been used to support decision-making. After all, process changes in organisations – particularly the radical changes implied by process engineering – should be carried out from a basis of *understanding* of organisational behaviour derived from systemic principles. As Forrester [For., 3] has recently proposed, the future role of top management will be that of corporate *designers* rather than corporate *operators*. Much of the time, it is the design of the organisation which is defective – usually in respect of misalignment of its core processes. The thesis instigates a new approach to Business Process Re-engineering (BPR), indicating that BPR is driven by the intense effort to overcome any possible

deadlocks in the decision-making procedure, when designing the process. This approach empowers BPR, which, till recently, was simply performed in companies, for the sake of improvement and creation of business processes that would meet specific time, quality, and economical criteria. In future work, there is need to clarify the process of business process re-engineering, where a distinction has to be made between system structure redesign and redesign of the decision-making mechanism.

Two very important modelling techniques have been thoroughly investigated. When process models are geared to delivering quantitative results, such as the average total time to process a project, or the percentage of time an employee is idle, System Dynamics and Petri nets provide the answers. System Dynamics abstracts over events to describe processes in terms of rates and accumulations, placing the emphasis on the feedback loops. System Dynamics' basic thesis is that several interconnected feedback loops can easily produce counter-intuitive behaviour, and therefore, simulation can greatly increase the understanding of such systems. An alternative to abstracting to the continuous quantities of rates and levels is to use Petri nets to provide a quantitative model, through discrete event simulation. And this is what makes the approach adopted in this thesis quite unique. Petri nets have so far been used to model real-time systems, workflow systems, flexible manufacturing systems, computer systems, production systems, and many others, but there has not been any effort to apply Petri nets in a human resources management problem before. Through them, the development of models for processes such as the one investigated in our case-study, acquire the mathematical notation and the plethora of analysis tools that test whether the process behaves as expected (validation), whether it is correct (verification), or whether it meets requirements with respect to throughput time, service and resources management (performance analysis).

Both modelling methodologies have provided promising results that can drive research on business processes further ahead. However, the intention is not to present any of these methodologies as the total business modelling solution. The nature of business processes is diverse and complicated. Each model, deriving from a different modelling methodology, has its own things to tell end-users, hence the analyst should use them alternately to satisfy the needs of the client.

During this research effort, business modelling literature has shown cases where business processes are modelled by an open system, i.e., a net exchanging tokens with some environment, in case Petri nets is used as a modelling method. Hence, it is

possible to determine what a business process should do by observing the relevant interactions with the environment, i.e., input and output are investigated to determine the prime objectives. Nevertheless, the study of processes modelled by a close-loop system seems to be a far more intriguing and challenging research topic. In chapter 6, the mathematical discrete-time equations describing the model of the case study were provided and a dynamic programming algorithm was used to design an optimal solution for the simplified case of the problem. The study revealed that the company's hiring policy acts as a feedback controller. The inputs of the controller are: (i) the external demand for projects y_k , (ii) the measured output x_k , which is the system's state-vector at time k , i.e. the number of the consultants employed at various levels, and (iii) the performance index/constraints that the company wishes to optimise/satisfy (for example, not exceeding deadlines for project completion, minimising the cost of employing external resources/subcontractors, etc). The overall objective is to make the error signal "as small as possible", i.e. to match the company's human resources to project demand, as closely as possible, over the specified time-horizon. Nevertheless, more efficient numerical algorithms are ultimately required to deal efficiently with complexity issues. Optimisation techniques are needed to enable decision-making – thus, driving the analysis away from experience, intuition, and informal debate. This is also an important direction for research.

Finally, to summarise, the main contributions of this research work are:

1. An introduction of a conceptual, systemic and control theoretic framework that is suitable to address a number of issues arising in the analysis and design of business processes.
2. An extended review and evaluation of various modelling methodologies and tools that are relevant to the modelling of business processes. Their basic concepts were presented and possible extensions for some of the techniques were suggested.
3. Classification of modelling methodologies in terms of the nature of the models that are capable to provide. Creation of a natural path for moving from a purely conceptual level of describing a process to a more formal one; integration of these methods.
4. Development of various models for the particular case study, making use of the integrated framework.

5. Specification of requirements for a system's modelling and control approach to business processes. Complexity and quantity of information drive the selection of a specific modelling technique.
6. Identification of needs for future work.

9.2 Future Work

From the systematic reviewing of many issues, it emerges that the challenge is to develop a new conceptual framework by which knowledge will be captured, and at the same time provide the designer with a powerful methodology with which he/she will be able to model the business processes. The new methodology must act as a unifying basis for modelling business processes, linking issues of design, or redesign, and providing a means of integrating abstract notions, heuristics and objectives to be achieved. Some other characteristics, the modelling technique should combine, are [Ould, 2]:

- the models should be diagrammatic rather than textual, since that makes them easier to comprehend and manipulate.
- the modelling notation should have formal, mathematical syntax and semantics so that it can be analysed and, possibly, enacted.
- it should be possible to handle *complexity*.

It is also interesting to investigate whether there is a new modelling method that combines some of the advantages of the already mentioned in the thesis techniques, and produces models that suffice three criteria: 1. they are understandable, 2. changeable, and 3. executable. Subsequently, the development of tools to address the specifics of the methodology also has to be addressed.

Furthermore, there is a need for further formal experiments with Petri nets and System Dynamics so as to test their ability to scale up and cope with real business processes. The evaluations provided in Chapters 6, 7, and 8, demonstrate the feasibility of the techniques to model a very simplified version of the problem addressed in the case study; however, to date, the techniques have not been tested against any large (or medium-sized) business processes. One of the limitations of studying a particular case study, is that much of the information gathered is context-dependent and subjective. Through this research work, it has been verified that System Dynamics and Petri nets

can be effectively applied to real business processes. However, a balanced range of case studies would greatly increase our confidence in any results.

It is also very difficult to directly compare the effectiveness of alternative notations at coping with real business processes – there is no simple way of separating inherent differences in the techniques from differences in the expertise of the modeller. We believe that with a reasonable number of case studies it would be possible to identify any correlation of the alternative notations and results.

Another possible area of research is the extension of the discrete-time model to incorporate more realistic information. One could in principle produce a more complex model by making the more realistic assumption that future project demand (and/or some model parameters) is known only probabilistically. In this case, the model becomes stochastic and the cost function is a random variable (e.g. the *expected* sum of absolute deviations over a finite horizon). Development of more efficient numerical techniques to deal with various optimisation issues is also required.

In this thesis, a classification of methodologies according to the nature of the modelling description of the problem (qualitative versus quantitative, static versus dynamic description) has been provided. An alternative area of future work would be the classification of processes and the understanding of which processes are best modelled by which techniques.

Last but not least, the specification of the model context of the Decision-Making Engine requires a classification of formal models and a first attempt of such a classification is given below. The diagram is taken by [Milne, 1], where it was used to classify the different types of diagnostic reasoning. Here, we have added some aspects to that so as to represent the four different levels of knowledge representation and the families of formal models. These levels are structural, behavioural, functional, and pattern matching. From each knowledge representation we are able to derive the next higher level of representation. The levels above correspond to families of reasoning (deep and shallow) type models.

Of the four levels, we can enter with knowledge at any level and exit with knowledge at the same or any other level. Every level contains a family of alternative models. The fact that knowledge at a given level may be deduced from knowledge at a lower level, implies that models are related in terms of projections; however, such mechanisms are not yet well understood.

The system can, therefore, be modelled as a whole entity in terms of an interconnected set of activities, which may be individually modelled on a different level. The process of modelling takes place after the *conceptual model* is formed. This represents the minimum necessary set of activities, at a particular level of detail, that the system must do [Wilson, 1].

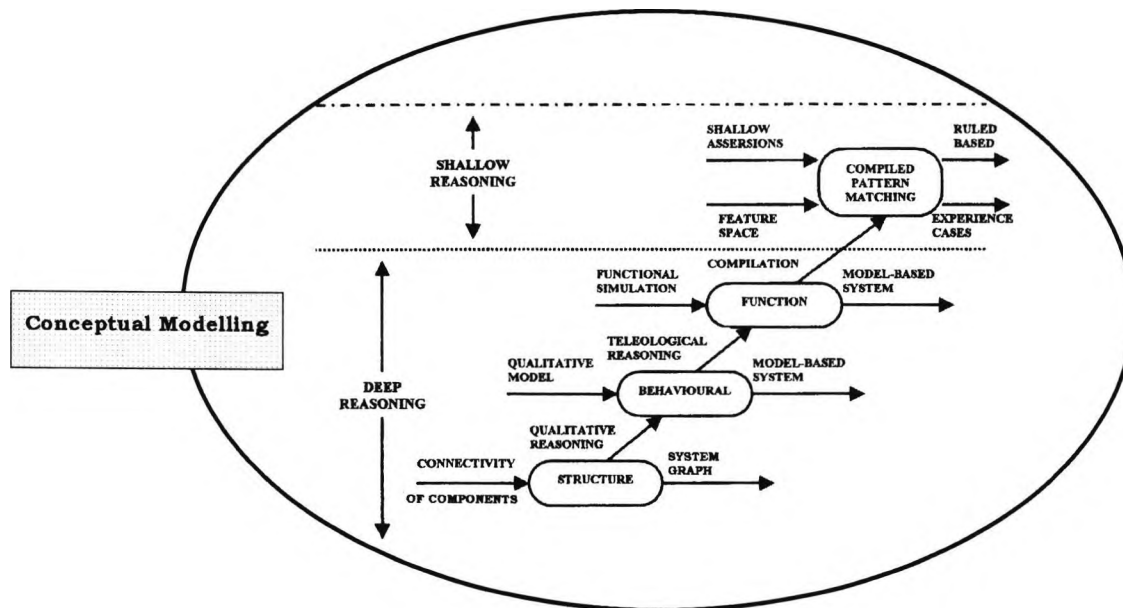


Figure (9.2): Nesting of formal modelling

A possible future work, hence, could be the proper nesting of models, i.e., the overall description of the mechanism the designer needs to follow in order to move around the different levels of reasoning, as well as the identification of the level of reasoning, to which each of the methodologies mentioned belongs.

REFERENCES:

- [Abdel-Hamid, 1] Abdel-Hamid, T. K. "The Dynamics of Software Project Staffing: A System Dynamics Based Simulation Approach" *IEEE Transactions on Software Engineering*, Vol. 15, No. 2, pp. 109-119, February 1989.
- [Abdel-Hamid, 2] Abdel-Hamid, T. K. *The dynamics of software development project management: An integrative system dynamics perspective*, Ph.D. dissertation, Sloan School of Management, MIT, January 1984.
- [Abramson *et al.*, 1] Abramson, B., J. Brown, W. Edwards, A. Murphy, and R. Winkler, "Hailfinder: A Bayesian system for forecasting severe weather" *International Journal of Forecasting*, 1996.
- [Agger, 1] Agger, S. *Grafisk Design af Skærbilleder*. Report from the SIP project, 1990.
- [Amsterdam, 1] Amsterdam, J. "Estimating order from causal ordering", *Proc. 7th Int. Workshop on Qualitative Reasoning about Physical Systems*, Orcas Island, WA. 1993.
- [Ander. & Richar., 1] Andersen, D. F. and G. P. Richardson, "Toward a pedagogy of System Dynamics" *TIMS Studies in the Management Sciences*, Vol. 14, 1980.
- [Andreassen *et al.*, 1] Andreassen, S., F. V. Jensen, S. K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A. Sørensen, A. Rosenfalck, and F. Jensen, "MUNIN- an expert EMG assistant" *Computer-aided electromyography and expert systems*, J. E. Desmedt (ed.), Elsevier Science, pp.255-277, 1989.

- [Antsaklis *et al.*, 1] Antsaklis, P., W. Kohn, A. Nerode and S. Sastry, (eds), "Hybrid Systems III". *Lecture Notes in Computer Science*, Springer-Verlag, 1995.
- [APQC, 1] *Process Classification Framework*, developed by APQC's International Benchmarking Clearinghouse in partnership with A. Anderson & Co., SC, 1995. (Cited in World Wide Web:
<http://www.apqc.org/b1/b1pubs/pubs2.htm#Framework>).
- [Backx, 1] Backx, A. C. P. M. "Engineering Aspects of Industrial Applications of Model-Based Control Techniques and System Theory", *Essays on Control; Perspectives in Theory and Its Applications*, H. Trentelman and J. C. Willems (eds), pp. 79-109, Birkhauser, Berlin.
- [Baer, 1] Baer, J. "A survey of Some Theoretical Aspects of Multiprocessing" *Computing Surveys*, Vol.5, No.1, pp.31-80, March 1973.
- [Baer, 2] Baer, J. "Modelling for Parallel Computation: A case Study", *Proceedings of the 1973 Sagamore Computer Conference on Parallel Processing*, IEEE, NJ, pp.13-22, Aug. 1973.
- [Beeri *et al.*, 1] Beeri, C., R. Fagin, D. Maier, and M. Yannakakis, "On the desirability of acyclic database schemes", *Journal of the Association for Computing Machinery*, Vol.30, No.3, pp.479-513, 1983.
- [Bennett, 1] Bennett, S. W. "Approximation in mathematical domains" *Proceedings AAAI-87*, Seattle, WA, 1987.

- [Berl. & Kuipers, 1] Berleant, D. and B. J. Kuipers, *Bridging the gap from qualitative to numerical simulation*, Technical Report AI91-158, Artificial Intelligence Lab. Dept. of Computer Sciences, University of Texas, 1991.
- [Berl. & Kuipers, 2] Berleant, D. and B. J. Kuipers, "Combined qualitative and numerical simulation with Q3", *Recent Advances in Qualitative Physics*, B. Faltings and P. Struss (eds.), MIT Press, Cambridge, MA, 1992.
- [Berl. & Kuipers, 3] Berleant, D. and B. J. Kuipers, "Qualitative and quantitative simulation: bridging the gap", *Artificial Intelligence*, Vol.95, pp.215-255, 1997.
- [Berleant, 1] Berleant, D. *Probabilities of qualitative behaviours from probability distributions on inputs*, Technical Report TR AI90-136, Artificial Intelligence Lab., Dept. of Computer Sciences, Austin, Texas, 1990.
- [Bert. & Brios., 1] Bertele, U. and Brioschi, *Nonserial dynamic programming*, London: Academic, 1972.
- [Berts., 1] Bertsekas, D. P. *Dynamic Programming: Deterministic and Stochastic Models*, Prentice-Hall, USA, 1987.
- [Berts. & Tsits., 1] Bertsekas, D. P. and J. N. Tsitsiklis, *Neuro-dynamic programming*, Athena Scientific, Belmont, MA, 1996.
- [Berts. & Tsits., 2] Bertsekas, D. P. and J. N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*, Athena Scientific, Belmont, MA, 1997.
- [Bobrow, 1] Bobrow, D. *Qualitative Reasoning about Physical Systems*,

-
- MIT Press, Cambridge, Mass., 1985.
- [Borch.,1] Borchardt, G. "Understanding causal descriptions of physical systems", *Proceedings 10th National Conference on Artificial Intelligence*, Cambridge, MA, MIT Press, 1992.
- [Bronowski, 1] Bronowski, J. "The Logic of the Mind", in *Man and the Science of Man*, W. R. Coulson and C. R. Rogers (eds.), Charles E. Merrill Publishing Company, Ohio, 1968.
- [Celier, 1] Cellier, F. E. *Continuous Systems Modelling*, Springer-Verlag, New York, 1991.
- [Chand. et al., 1] Chandrasekaran, B., J. W. Smith, and J. Sticklen, "Deep Models and their Relation to Diagnosis" *Artificial Intelligence in Medicine*, Vol.1, pp.29-40, 1989.
- [Chand. et al., 2] Chandrasekaran, B., J. Josephson, and A. Keuneke, "Functional Representation as a Basis for Explanation Generation" *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, pp.726-731, 1986.
- [Check. & Scholes, 1] Checkland, P. and J. Scholes, *Soft Systems Methodology in Action*, Wiley, 1990.
- [Check., 1] Checkland, P. *Systems Thinking, Systems Practice*, Wiley, 1981.
- [Chen, 1] Chen, P. P.-S. "The Entity-Relationship Model – Toward a Unified View of Data", *ACM Transactions on Database Systems*, Vol. 1, No. 1, pp. 9-36, March 1976.

- [Chin *et al.*, 1] Chin, R. C. Y., G. W. Hedstrom, and F. A. Howes, "Considerations on solving problems with multiple scales", *Multiple Time Scales*, J. U. Brackbill and B. I. Cohen (eds.), Academic Press, New York, 1985.
- [Chung, 1] Chung, K. L. *Representing and Using Non-Functional Requirements for Information System Development: A Process-Oriented Approach*, Ph.D. Thesis, Dept. of Comp. Sci. Univ. of Toronto, June 1993.
- [Courtois *et al.*, 1] Courtois, P., F. Heymans, and D. Parnas, "Concurrent Control with 'Readers' and 'Writers'" *Communications of the ACM*, Vol.14, No.10, pp.667-668, Oct. 1971.
- [Coyle, 1] Coyle, R. G. *Management System Dynamics*, John Wiley & Sons, Chichester, 1977.
- [Coyle, 2] Coyle, R. G. *System Dynamics Modelling, a practical approach*, Chapman & Hall, 1996.
- [Coyle, 3] Coyle, R.G. "The optimisation of defence expenditure" *European Journal of Operational Research*, Vol. 56, pp.304-18, 1992.
- [Croft & Lefk., 1] Croft, W. B. and L. S. Lefkowitz, *A Goal-Based Representation of Office Work, Office Knowledge: Representation, Management, and Utilisation*, W. Lamersdorf (ed.), Elsevier, pp.99-124, 1988.
- [D'Ambrosio, 1] D'Ambrosio, B. "Extending the Mathematics in Qualitative Process Theory", *Proceedings 6th National Conf. on Artificial Intelligence*, San Mateo, CA. Morgan Kaufmann, pp.595-599, 1987.

- [D'Ambrosio, 2] D'Ambrosio, B. "Extending the mathematics in qualitative process theory", *Artificial Intelligence, Simulation, and Modelling*, Wiley-Interscience, New York, pp.133-158, 1989.
- [Dague *et al.*, 1] Dague, P., O. Raiman, and P. Devès, "Troubleshooting: When Modelling is in Trouble." *Proceedings AAAI-87*, pp.600-605, 1987.
- [Dague, 1] Dague, P. "Numeric reasoning with relative orders of magnitude", *Proc. 11th National Conf. on Artificial Intelligence*, Cambridge, MA, MIT Press, 1993.
- [Dalle Molle *et al.*, 1] Dalle Molle, D. T., T. F. Edgar and B. J. Kuipers, "Qualitative modelling and simulation of dynamic systems." *Comput. Chem. Engng.*, Vol. 12, pp.853-866, 1988.
- [Daven., 1] Davenport, T. H. *Process Innovation: Reengineering Work through Information Technology*, Harvard Business School Press, Boston, MA, 1993.
- [Dav. & Short, 1] Davenport, T. and J. Short, "The New Industrial Engineering: Information Technology and Business Process Redesign" *Sloan Management Review*, pp.11-28, 1990.
- [deKleer, 1] deKleer, J. *Causal and Teleological Reasoning in Circuit Recognition*, MIT AI-TR-529, September, 1979.
- [deKleer, 2] de Kleer, J. "How circuits work" *Artificial Intelligence*, Vol. 24, pp.205-280, 1984.

- [deKleer & Brown, 1] deKleer, J. and J. S. Brown, "A qualitative physics based on confluences" *Artificial Intelligence*, Vol.24, pp.7-84, 1984.
- [deKleer & Brown, 2] deKleer, J. and J. S. Brown, "Mental Models of Physical Mechanisms and their Acquisition" in *Cognitive Skills and their Acquisition*, J. R. Anderson, ed. Lawrence Erlbaum Associates, Hillsdale, N. J, pp.285-309, 1980.
- [Dicesare *et al.*, 1] Dicesare, F., G. Harhalakis, M. J. Proth, M. Silva, and B. F. Vernadat, *Practice of Petri nets in Manufacturing*, Chapman & Hall, 1993.
- [Dijk., 1] Dijkstra, E. "Cooperating Sequential Processes" *Programming Languages*, F. Genuys (editor), NJ, pp.43-112, 1968.
- [Douglas *et al.*, 1] Douglas, J. M., M. F. Malone and M. F. Doherty, "The interaction between separation system synthesis and process synthesis." *Comput. Chem. Engng.*, Vol. 9, pp.447-462, 1985.
- [Douglas, 1] Douglas, J. M. "Process Operability and Control of Preliminary Designs" *Process Control*, Vol. 2, p.497, 1982.
- [Douglas, 2] Douglas, J. M. *Conceptual Design of Chemical Processes*, McGraw-Hill, New York, 1987.
- [Duncker, 1] Duncker, K. "On problem solving psychological monographs" Vol. 58, No. 5, 1945.
- [Dvorak & Kuipers, 1] Dvorak, D. and B. Kuipers, "Model-based Monitoring of

- Dynamic Systems”, *Proc. 11th Int. Joint Conf. on Artificial Intelligence (IJCAI-89)*, Los Altos, CA. Morgan Kaufmann, pp.1238-1243, 1989.
- [Dvorak & Kuipers, 2] Dvorak, D. and B. Kuipers, “Process monitoring and diagnosis: a model-based approach”, *IEEE Expert*, Vol.6, No.3, pp.67-74, 1991.
- [Dvorak, 1] Dvorak, D. *Monitoring and Diagnosis of Continuous Dynamic Systems Using Semiquantitative Simulation*, Ph.D. Dissertation, Dept. of Computer Sciences, University of Texas, Austin, Texas, 1992.
- [Fanni & Giua, 1] Fanni, A. and A. Giua, “A formal methodology for representing qualitative equations with Petri nets” *Proc. 2nd IMACS Symp. Mathematical Modelling MATHMOD*, pp. 217-222, Vienna, Austria, Feb. 1997.
- [Fanni & Giua, 2] Fanni, A. and A. Giua, “Discrete Event Representation of Qualitative Models Using Petri Nets” *IEEE Transactions on Systems, Man, and Cybernetics – part B: Cybernetics*, Vol.28, No.6, pp.770-780, December 1998.
- [Fanni *et al.*, 1] Fanni, A., A. Giua, and D. Y. Lee, “Petri nets in the qualitative modelling of systems” *Proc. 1994 IEEE Int. Conf. Systems, Man, Cybern.*, pp.2316-2321, San Antonio, Oct. 1994.
- [Fenton & Hill, 1] Fenton, N. and G. Hill, *Systems Construction and Analysis: A mathematical and logical framework*, McGraw Hill International, Software Engin. Series, 1993.
- [For. & Senge, 1] Forrester, J. W. and P. M. Senge, “Tests for building

- confidence in System Dynamics models.” *TIMS Studies in the Management Sciences*, Vol. 14, 1980.
- [For., 1] Forrester, J. W. “System Dynamics-Future Opportunities” *TIMS Studies in the Management Sciences*, Vol. 14, pp.7-21, 1980.
- [For., 2] Forrester, J. W. *Industrial Dynamics*, MIT Press, 1961.
- [For., 3] Forrester, J. “The CEO as Organisational Designer”, *McKinsey Quarterly*, No.2, 1992.
- [Forbus, 1] Forbus, K. “Qualitative reasoning about physical systems” *Proc. International joint conference on artificial intelligence (IJCAI 7)*, California, pp.326-330, 1981.
- [Forbus, 2] Forbus, K. D. “Qualitative process theory” *Artificial Intelligence*, Vol. 24, pp.85-168, 1984.
- [Gin. & Gog.,1] Ginali, S., Goguen, J. (1978), “A categorical approach to General Systems, Developments and trends”, [Proceedings of NATO conference held in Binghamton, NY, Aug. 15-17 1977], New York: Plenum Press, pp 257-70,1978.
- [Gog., 1] Goguen, J. A. “Mathematical representation of hierarchically organised systems” in the *Global Systems Dynamics International Symposium* Charlottesville 1969, Karger, Basel/ New York, pp. 112-128, 1970.
- [Gog., 2] Goguen, J. A. “Systems and minimal realisation”, *Proceedings I.E.E.E Conference on Decision and Control*, pp. 42-46, 1972.

- [Gold. & Charn., 1] Goldman, R. and E. Charniak, "A language for construction of belief networks" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.15, No.3, pp.196-208, 1993.
- [Golumbic, 1] Golumbic, M. C. *Algorithmic graph theory and perfect graphs*, London: Academic, 1980.
- [Graham, 1] Graham, A. K. "Parameter estimation in System Dynamics modelling" *TIMS Studies in the Management Sciences*, Vol. 14, 1980.
- [Grosz & Roll., 1] Grosz, G. and C. Rolland, "Using Artificial Intelligence Techniques to Formalise the Information Systems Design Process" *Proceedings of the Int. Conference on Database and Expert Systems Applications*, pp.374-380, 1990.
- [Guarino, 1] Guarino, N. "Formal ontology, conceptual analysis and knowledge representation", *Journal of Human – Computer Studies*, Vol. 43, pp.625-640, 1995.
- [Guarino, 2] Guarino, N. "Formal Ontology and Information Systems" in *Formal Ontology in Information Systems, Proceedings of the 1st International Conference (FOIS'98)*, N. Guarino (ed.), IOS Press, pp.3-15, 1998.
- [Gulla & Lindl., 1] Gulla, J. A. and O. I. Lindland, "Modelling Cooperative Work for Workflow Management", *Proceedings of Advanced Information Systems Engineering (CaiSE 94), Lecture Notes in Computer Science*, Vol.811, Springer-Verlag, pp.53-65, 1994.
- [Hack, 1] Hack, M. "Decision Problems for Petri Nets and Vector

- Addition Systems” *Computation Structures Group Memo 95, Project MAC*, Massachusetts, 1974.
- [Hack, 2] Hack, M. “Petri Net Languages” *Computation Structures Group Memo 124, Project MAC*, Massachusetts, June 1975.
- [Hammer, 1] Hammer, M. “Reengineering Work: Don’t Automate, Obliterate” *Harvard Business Review*, pp.104-112, July – August 1990.
- [Hammer & Champy, 1] Hammer, M. and J. Champy, *Reengineering the Corporation: A Manifesto for Business Revolution*, Harper Collins Publishers, New York, 1993.
- [Hecker., 1] Heckerman, D. “Causal independence, knowledge acquisition, and inference”, *Proceedings of the 9th conference on uncertainty in artificial intelligence*, D. Heckerman and A. Mamdani (eds.), CA: Morgan Kaufmann, pp.122-127, 1993.
- [Hempel, 1] Hempel, C. G. “The Logic of Functional Analysis” *Symposium on Sociological Theory* (Ed. Llewellyn Gross), Row, Peterson and Company, New York, 1959.
- [Holt & Comm., 1] Holt, A. and F. Commoner, “Events and Conditions” *Record of the project MAC Conference on Concurrent Systems and Parallel Computation*, NJ, pp.1-52, 1970.
- [Holt *et al.*, 1] Holt, A., H. Saint, R. Shapiro, and S. Warshall, “Final report of the Information System Theory Project” *Technical Report RADC-TR-68-305*, Rome Air Development Center, NJ, Sept. 1968.

- [Horn, 1] Horn, W. *Causal AI Models: Steps Toward Applications*, Hemisphere Publishing Company, New York, 1990.
- [Howard & Matheson, 1] Howard, R. A. and J. E. Matheson, "Influence diagrams", *Readings on the principles and applications of decision analysis*, R. A. Howard and J. E. Matheson (eds.), Vol.2, CA: Strategic Decisions Group, pp.719-762, 1984.
- [Jac. & Holt., 1] Jacobs, S. and R. Holten, "Goal Driven Business Modelling – Supporting Decision Making within Information Systems Development" *Proceedings of the Conference on Organisational Computing Systems, (COOCS'95)*, Milpitas, CA, Aug 1995.
- [Jen., 1] Jensen, K. *Coloured Petri Nets: Basic concepts, analysis methods and practical use*, Vol. 1, Springer-Verlag, 1992.
- [Jensen *et al.*, 1] Jensen, F. V., K. G. Olesen, and S. K. Andersen, "An algebra of Bayesian belief universes for knowledge-based systems", *Networks*, Vol.20, pp.637-659, 1990.
- [Jensen *et al.*, 2] Jensen, F., F. V. Jensen, and S. L. Dittmer, "From influence diagrams to junction trees", *Proceedings of the 10th conference on uncertainty in artificial intelligence*, R. L. de Mantaras and D. Poole (eds.), pp.367-373, Morgan Kaufmann, 1994.
- [Jensen, 1] Jensen, H. *The Interface, Design and Structure*, Report from the SIP project, 1990.
- [Jensen, 2] Jensen, F. V. *An Introduction to Bayesian Networks*, Springer-Verlag, England, 1996.

- [Jørgensen, 1] Jørgensen, S. S. *Generic MFM models for use in fault diagnosis of ship system's machinery*, Ph.D. thesis, Technical University of Denmark, Institute of Automatic Control, 1993.
- [Jowett, 1] Jowett, B. *The Dialogues of Plato*, Oxford University Press, Vol. 3, 1953.
- [Iwasaki *et al.*, 1] Iwasaki, Y., R. Fikes, M. Vesconi, and B. Chandrasekaran, "How things are supposed to work: Capturing functional knowledge in device design", *Proc. 13th Int. Joint Conf. on Artificial Intelligence*, San Mateo, CA. Morgan Kaufmann, pp.1516-1522, 1993.
- [Karc., 1] Karcaniyas, N. "Systems and Control Issues in the Overall Production and Design of Industrial Processes" *Report of ESPRIT Network ICIMS-NOE*, Nov. 1996.
- [Karc., 2] Karcaniyas, N. "Global Process Instrumentation: Issues and Problems of a Systems and Control Theory Framework", *Measurement*, Vol.14, pp 103-113, 1994.
- [Karc., 3] Karcaniyas, N. "The Integration Problem in Advanced Manufacturing: Systems and Control Issues", *Proceedings of ASI'2000*, Bordeaux, France, Sep. 18-20, 2000.
- [Karc.*et al.*, 1] Karcaniyas, N., M. Newby and I. Leontaritis, "Modelling of the integrated business and operational issues of industrial processes: A systems based approach" *Proceedings of ASI'98*, Conference Bremen, June 14-17 1998.
- [Keller, 1] Keller, R. "Generalised Petri Nets as Models for System Verification" *Technical Report 202*, Dept. of Electrical

- Engineering, Princeton University, New Jersey, 1975.
- [Keuneke, 1] Keuneke, A. M. "Device representation – The significance of functional knowledge" *IEEE Expert*, pp.22-25. April 1991.
- [Keuneke, 2] Keuneke, A. *Machine Understanding of Devices: Causal Explanation of Diagnostic Conclusions*, Ph.D. thesis, Ohio State University, Columbus, 1989.
- [Kingston, 1] Kingston, J. K. C. "X-MATE: Creating an interpretation model for credit risk assessment", *Expert Systems 91*, *British Computer Society*, Cambridge University Press, 1991.
- [Kingston, 2] Kingston, J. K. C. "Re-engineering IMPRESS and X-MATE using CommonKADS", *Expert Systems 93*, *British Computer Society*, Cambridge University Press, 1993.
- [Kingston, 3] Kingston, J. K. C. "Modelling Business Processes using the Soft Systems Approach", *International Symposium on the Management of Industrial and Corporate Knowledge (ISMICK' 94)*, Compiègne, France, October 1994. Also available from AIAI as *AIAI-TR-157*, January 1995.
- [Klir, 1] Klir, G. J. *An approach to General Systems Theory*, Van Nostrand-Reinhold, Princeton, New Jersey, 1967.
- [Klir, 2] Klir, G. J., "The polyphonic General Systems Theory", in *Trends in General Systems Theory*, Edit. G. Klir, Chichester, Wiley-Interscience, New York (1972).
- [Kreiss, 1] Kreiss, H. O. "Problems with different time scales",

-
- Multiple Time Scales*, J. U. Brackbill and B. I. Cohen (eds.), Academic Press, New York, 1985.
- [Kuipers & Berleant, 1] Kuipers, B. and D. Berleant, "Using incomplete quantitative knowledge in qualitative reasoning", *Proc. 7th National Conf. on Artificial Intelligence*, 1988.
- [Kuipers, 1] Kuipers, B. "Commonsense reasoning about causality: deriving structure from behaviour", *Artificial Intelligence*, Vol.24, pp.169-204, 1984.
- [Kuipers, 2] Kuipers, B. "The limits of qualitative simulation", *Proc. 9th Int. Joint Conf. Artificial Intelligence*, Los Angeles, pp.128-136, 1985.
- [Kuipers, 3] Kuipers, B. "Qualitative simulation", *Artificial Intelligence*, Vol.29, pp.289-338, 1986.
- [Kuipers, 4] Kuipers, B. "Abstraction by Time-Scale in Qualitative Simulation", *Proceedings AAAI-87*, pp.621-625, 1987.
- [Kuipers, 5] Kuipers, B. *Qualitative Reasoning: Modelling and Simulation with Incomplete Knowledge*, MIT Press, Cambridge, MA., 1994.
- [Larsen, 1] Larsen, M. N. *Modelling start-up tasks using functional models. Interactive planning for integrated supervision and control of complex plant*. CRCJSP Ispra, Final Report Project 4937-92-08-ED ISP DK, 1993.
- [Larsson, 1] Larsson, J. E. *Knowledge-based methods for control systems*, Ph.D. thesis. Lund Institute of Technology, Dept. of Automatic Control, Sweden, 1992.

- [Lauritzen, 1] Lauritzen, S. L. *Graphical Models*, Oxford University Press, 1996.
- [Lee, 1] Lee, J. *SIBYL: A Qualitative Decision Management System, Artificial Intelligence at MIT: Expanding Frontiers*, P. Winston (ed.), MIT Press, MA Cambridge, June 1990.
- [LeFèvre, 1] LeFèvre, J. "Bond Graphs are not the proper tools to model the world but Bond + Kinetic Graphs could very well be!" *Proceedings of the 3rd International Conference on Bond Graph Modelling and Simulation*, Simul. Series, Vol.29, No.1, pp.3-9, SCS Publ., San Diego, Calif. USA, 1996.
- [LeFèvre, 2] LeFèvre, J. "Applications of Kinetic Bond Graphs in Corporate and Production Management Systems" in "*Commande des systèmes industriels*", Belfort AFCET-IMACS Symposium, pp.226-232, 1997.
- [Leo. & Karc., 1] Leontaritis, I. and N. Karcianas, "A unified approach to Systems Theory, Data Modelling and Databases based on Object Oriented methodology" *Proceedings of ASI'98*, Bremen, June 14-17, 1998.
- [Levy et al., 1] Levy, F., G. Thompson, and J. Wiest, "Introduction to the Critical-Path Method" *Industrial Scheduling*, J. F. Muth and G. L. Thompson, Prentice-Hall, p.335, 1963.
- [Lind & Marcus, 1] Lind, D. and B. Marcus, *Symbolic Dynamics and Coding: An Introduction*, Cambridge University Press, Cambridge, 1995.
- [Lind, 1] Lind, M. et al., "Human-Machine Interface for Diagnosis

- based on Multilevel Flow Modelling" 2nd *European Meeting on Cognitive Science Approaches to Process Control*, Siena, Italy, Oct. 1989.
- [Lind, 2] Lind, M. *Representing Goals and Functions of Complex Systems*, Report presented in the Technical University of Denmark, 90-D-381, Nov. 1990.
- [Lind, 3] Lind, M. "Abstractions – Description of classes and their use" *Report IACS 90-D-380*, Technical University of Denmark, 1990.
- [Lind, 4] Lind, M. *Functional architectures for systems management and control. Interactive Planning for Integrated Supervision and Control of Complex Plant*, CECJRC Ispra, Final report project 4937-92-08-ED ISP DK, 1993.
- [Lind, 5] Lind, M. "Modelling Goals and Functions of Complex Industrial Plants" *Applied Artificial Intelligence*, Vol. 8, pp.259-283, 1994.
- [Lind, 6] Lind, M. "Status and Challenges of Intelligent Plant Control", 4th *IFAC Symposium on Dynamics and Control of Chemical Reactors, Distillation Columns and Batch Processes (DYCORD + '95)*, Helsingør, 7-9 June 1995.
- [Lindley, 1] Lindley, D. V. *Making Decisions*, Wiley, New York, 1971.
- [Louc. & Zic., 1] Loucopoulos, P. and R. Zicari, *Conceptual Modelling, Data Bases and CASE*, John Wiley and Sons, Chichester, England, 1992.
- [McLel., 1] McLellan, T. "Data Modelling: Finding the Perfect Fit",

- Dispatch*, Computing and Network Services, University of Alberta, April 1995.
(<http://www.islandnet.com/~tmc/html/articles/datamodl.htm>)
- [MacFarlane, 1] MacFarlane A. G. J. "Information, Knowledge and Control", *Essays on Control; Perspectives in Theory and Its Applications*, H. I. Trentelman and J.C. Willems (eds), pp:1-28, Birkhauser, Berlin, 1993.
- [MacFarlane, 2] MacFarlane, D. C. "Holonc Manufacturing Systems in Continuous Processing: Concepts and Control Requirements", *ASI '95*, Portugal, 1995.
- [Mavro. & Steph., 1] Mavrovouniotis, M. L. and G. Stephanopoulos, "Reasoning with Orders of Magnitude and Approximate Relations" *Proceedings AAAI-87*, pp.626-630, 1987.
- [Mavro. & Steph., 2] Mavrovouniotis M. L. and G. Stephanopoulos, "Formal Order-of-Magnitude Reasoning in Process Engineering" *Comp. Chem. Engng.*, Vol. 12, pp.867-880, 1988.
- [Mavro. et al., 1] Mavrovouniotis M. L., G. Stephanopoulos, and G. N. Stephanopoulos, "Order-of-Magnitude analysis and reasoning in the modelling of biochemical pathways" *AICHE 1987 Annl. Mtg., American Institute of Chemical Engineers*, New York, 1987.
- [Meld. & Holt, 1] Meldman, J. and A. Holt, "Petri Nets and Legal Systems" *Jurimetrics Journal*, Vol.12, No.2, pp.65-75, Dec. 1971.
- [Mes., 1] Mesarovic, M., D., "A mathematical theory of General Systems", in *Trends in General Systems Theory*, Edit. G. Klir, Chichester, Wiley-Interscience, New York (1972).

- [Mes., 2] Mesarovic, M., D. and Y. Takahara, *Abstract Systems Theory*, Springer, 1989.
- [Milne, 1] Milne, R. "Strategies for Diagnosis" *IEEE Trans. on Systems, Man and Cybernetics*, Vol. **SMC-17**, pp.333-339, 1987.
- [Moh. & Sim., 1] Mohammed, J. and R. Simmons, "Qualitative simulation of semiconductor fabrication", *Proceedings AAAI-86*, MIT Press, Cambridge, MA, pp.794-799, 1986.
- [Moore, 1] Moore, R. E. *Interval Analysis*, Prentice-Hall, Inc., Englewood Cliffs, NJ., 1966.
- [Moore, 2] Moore, R. E. "Methods and Applications of Interval Arithmetic", *Studies in Applied Mathematics*, SIAM, Philadelphia, 1979.
- [Moore & Bren., 1] Moore, K. E. and J. E. Brennan, "Petri Nets and Simulation: A Tutorial", *Proceedings of the 1995 Summer Computer Simulation Conference*, T. I. Ören and L.G. Birta (eds.), pp. 44-50, Canada, July 1995.
- [Morari *et al.*, 1] Morari, M., Y. Arkun and G. Stephanopoulos, "Studies in the Synthesis of Control Structures for Chemical Processes" *AICHE Journal*, Vol.26, Vol.27, (Parts I to V).
- [Murata, 1] Murata, T. "Petri Nets, Marked Graphs, and Circuit-System Theory" *IEEE Circuits and Systems Society Newsletter*, Vol.11, No.3, pp.2-12, 1977.
- [Murata, 2] Murata, T. "Petri nets: Properties, analysis and applications" *Proc. IEEE*, Vol.77, pp.541-580, Apr. 1989.

- [Murthy, 1] Murthy, S. "Qualitative reasoning at multiple resolutions" *Proceedings AAAI-88*, St. Paul, MN, 1988.
- [Mylop. et al., 1] Mylopoulos, J., L. Chung, and B. Nixon, "Representing and Using Non-Functional Requirements: A Process-Oriented Approach" *IEEE Trans. Soft. Eng.*, Vol. 18, No. 6, June 1992.
- [Mylop., 1] Mylopoulos, J. "Conceptual Modelling and Telos" In *Conceptual Modelling and Data Bases and CASE*, edited by P. Loucopoulos and R. Zicari, pp.49-68.
- [Nagel, 1] Nagel, E. "A Formalisation of Functionalism" *Logic without Metaphysics*, Free Press, pp.247-283, 1956.
- [Naug., 1] Naughton, J. *The Soft Systems Approach. Soft Systems Analysis: An introductory guide*, The Open University Press, Portsmouth, 1984.
- [Nayak, 1] Nayak, P. P. "Order of magnitude reasoning using algorithms" *Proceedings 7th International Workshop on Qualitative Reasoning*, WA, 1993.
- [Ner. & Kohn, 1] Nerode, A. and W. Kohn, "Models for hybrid systems: Automata topologies, controllability, observability", R.L. Grossman, A. Nerode, A. P. Ravn and H Rischel (eds) *Hybrid Systems*, pp:317-356, Springer Verlag, New York, 1993.
- [Noe, 1] Noe, J. "A Petri Net Model of the CDC 6400" *Proceedings ACM SIGOPS Workshop on System Performance Evaluation*, NJ, pp.362-378, April 1971.

- [Non. & Takeuchi, 1] Nonaka, I. and Takeuchi, H. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*, Oxford Univ Press, 1995.
- [Ochard, 1] Ochard, R., A., "On an approach to General Systems Theory", in *Trends in General Systems Theory*, Edit. G. Klir, Chichester, Wiley-Interscience, New York, 1972.
- [Okuda & Ushio, 1] Okuda, K. and T. Ushio, "Petri net based qualitative simulation" *Proc. IASTED Int. Symp. Expert Systems Theory Applications*, Los Angeles, Dec. 1990.
- [Okuda & Ushio, 2] Okuda, K. and T. Ushio, "Hierarchical qualitative simulation for large scale dynamic systems" *Applications of Artificial Intelligence in Engineering VI*, G. Rzevski and R. A. Adey (eds.), Amsterdam, Elsevier, pp.301-317.
- [Open, 1] Maddison, R. *et al.*, *A. Conceptual Modelling B. Logical Modelling*, The Open University Press, 1980.
- [Osman, 1] Osman, A. *The Interface Substrate*, Report from the SIP project, Institute of Automatic Control Systems, 1990.
- [Osman, 2] Osman, A. *Graphical control environment (Grace)*, Ph.D. thesis, Technical University of Denmark, Institute of Automatic Control Systems, 1992.
- [Ould, 1] Ould, M. A. *Business Processes: Modelling and Analysis for Re-engineering and Improvement*, John Wiley & Sons, Chichester, England, 1995.
- [Ould, 2] Ould, M. A. "Process Modelling: Why, What and How", *Software Assistance for Business Re-Engineering*, K.

- Spurr, P. Layzell, L. Jennison, and N. Richards (eds.), Wiley, Chichester, 1993.
- [Oyel. & Kramer, 1] Oyeleye, O. O. and M. A. Kramer, "Qualitative simulation of process plants" *Proc. 10th IFAC World Cong. Autom. Control*, Vol. 6, pp.324-329, 1987.
- [Pan & Tenen., 1] Pan, J. Y. and J. M. Tenenbaum, "P.I.E.S.: An engineering do-it-yourself knowledge system for interpretation of parametric test", *Data. Proc. Am. Ass. Artificial Intelligence*, Philadelphia, pp.836-843, 1986.
- [Pars. & Sul., 1] Parsael, H. R. and W. G. Sullivan, *Concurrent Engineering*, Chapman and Hall, London, 1993.
- [Patil, 1] Patil, S. *Coordination of Asynchronous Events*, Ph.D. dissertation, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Massachusetts, May 1970.
- [Pearl, 1] Pearl, J. "Reverend Bayes on inference engines: a distributed hierarchical approach", *Proceedings of American Association for Artificial Intelligence National Conference on AI*, Pittsburgh, pp.133-136, 1982.
- [Pearl, 2] Pearl, J. "A constraint-propagation approach to probabilistic reasoning", *Uncertainty in Artificial Intelligence*, L. M. Kanal and J. Lemmer (eds.), Amsterdam, pp.357-370. 1986.
- [Pearl, 3] Pearl, J. "Fusion, propagation, and structuring in belief networks", *Artificial Intelligence*, Vol.29, No.3, pp.241-288, 1986.

- [Perel., 1] Perelman, L. J. "Time in System Dynamics" *TIMS Studies in the Management Sciences*, Vol. 14, 1980.
- [Peterson, 1] Peterson, J. L., "Petri Nets" *Computing Surveys*, Vol.9, No.3, pp.223-252, Sept. 1977.
- [Peterson, 2] Peterson, J. L. *Petri Net Theory and the Modelling of Systems*, Englewood Cliffs, NJ: Prentice-Hall, 1981.
- [Petri, 1] Petri, C. *Kommunikation mit Automaten*, Ph.D. dissertation, University of Bonn, West Germany, 1962.
- [Poh & Chew, 1] Poh, H. L. and W. W. Chew, "Business Process Reengineering: Definitions and Models Revisited",
- [Pohl *et al.*, 1] Pohl, K., R. Domges, and M. Jarke, "Decision Oriented Process Modelling" *Proceedings of the 9th Int. Software Process Workshop*, 1994.
- [Porter & Millar, 1] Porter, M. E. and V. E. Millar, "How Information Gives You Competitive Advantage", *Harvard Business Review*, pp.149-160. July 1985.
- [PRIMA, 1] PRIMA: Process Industries Manufacturing Advantage, E.U. Information Technologies Project (8234), 1996.
- [Ram. & Wonh., 1] Ramadge, P. J. and W. M. Wonham, "The control of discrete event systems" *Proc. IEEE*, Vol.77, pp.81-98, Jan. 1989.
- [Raiman, 1] Raiman, O. "Order of magnitude reasoning" *Proceedings AAAI-86*, pp.100-104, 1986.

- [Raiman, 2] Raiman, O. "Order of magnitude reasoning" *Artificial Intelligence*, Vol. 51, pp.11-38, 1991.
- [Rasmus., 1] Rasmussen, J. *Information Processing and Human-Machine Interaction*, North Holland, New York, 1986.
- [Richar. & Pugh, 1] Richardson, G. P. and A. L. Pugh, *Introduction to System Dynamics Modelling with DYNAMO*, MIT Press, 1981.
- [Rieger & Grinb., 1] Reiger, C. and M. Grinberg, "The declarative representation and procedural simulation of causality in physical mechanisms", *Proc. 5th Int. Joint Conf. On Artificial Intelligence*, San Mateo, CA. Morgan Kaufmann, pp.250-256, 1977.
- [Rijnsdorp, 1] Rijnsdorp, J. E. *Integrated Process Control and Automation*, Elsevier, Amsterdam, 1991.
- [Risch, Troy. & Ster., 1] Risch, J. D., L. Troyano-Bermúdez, and J. D. Sterman, "Designing corporate strategy with system dynamics: a case study in the pulp and paper industry" *System Dynamics Review*, Vol. 11, No. 4, pp.249-274, Winter 1995.
- [Rober. et al., 1] Roberts, N. et al. *Introduction to Computer Simulation: A System Dynamics Modelling Approach*, Addison-Wesley, Reading MA, 1983.
- [Rolland & Cauvet, 1] Rolland, C. and C. Cauvet, "Trends and Perspectives in Conceptual Modelling", in *Conceptual Modelling, Data Bases and CASE*, ed. Loucopoulos & Zicari, John Wiley & Sons, Chichester, England, pp.27-48, 1992.

- [Rumbaugh *et al.*, 1] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy & W. Lorensen, *Object-Oriented Modelling and Design*, Prentice Hall Int., 1991.
- [Sassen *et al.*, 1] Sassen, J. M. A., P. C. Riedijk, and R. B. M. Jaspers, "Using multilevel flow models for fault-diagnosis of industrial processes" *3rd European Conference on Cognitive Science Approaches to Process Control*, Cardiff, pp.207-216, 1991.
- [Schein, 1] Schein, E. H. *Organisational Psychology*, Englewood Cliffs, NJ: Prentice-Hall, 1980.
- [Semb. & Chand., 1] Sembugamoorthy, V. and B. Chandrasekaran, "Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems" in *Experience, Memory, and Reasoning*, J. L. Kolodner and C. K. Riesbeck, eds., Lawrence Erlbaum Associates, Hillsdale, N. J., pp.47-73, 1986.
- [Shachter, 1] Shachter, R. D. "Evaluating influence diagrams", *Operations Research*, Vol.34, No.6, pp.871-882, 1986.
- [Shenoy, 1] Shenoy, P. P. "Valuation-based systems for bayesian decision analysis", *Operations Research*, Vol.40, No.3, pp.463-484, 1992.
- [Simmons & Davis, 1] Simmons, R. and R. Davis, "Generate, test, and debug: Combining associational rules and causal models", *Proc. 10th Int. Joint Conf. on Artificial Intelligence*, San Mateo, CA. Morgan Kaufmann, 1987.
- [Skyt., 1] Skyttner, L. *General Systems Theory: An introduction*,

MacMillan Press, London, 1996.

- [Spieg. & Knill-Jones, 1] Spiegelhalter, D. J. and R. P. Knill-Jones, "Statistical and knowledge-based approaches to clinical decision-support systems", *Journal of the Royal Statistical Society, Series A*, Vol.147, pp.35-77, 1984.
- [Starr, 1] Starr, P. J. "Modelling Issues and Decisions in System Dynamics" *TIMS Studies in the Management Sciences*, Vol. 14, pp.45-59, 1980.
- [Stev., 1] Stevenson, R. "Strategic Business Process Engineering: A Systems Thinking Approach Using itthink", *Software Assistance for Business Re-Engineering*, K. Spurr, P. Layzell, L. Jennison, and N. Richards (eds.), Wiley, Chichester, 1993.
- [Stover, 1] Stover, J. G. "Including future events in System Dynamics Models" *TIMS Studies in the Management Sciences*, Vol. 14, 1980.
- [Suchman, 1] Suchman, L. A. *Plans and Situated Actions*, Cambridge University Press, 1987.
- [Tak. & Takah., 1] Takahashi, S. and Y. Takahara, *Abstract Systems Theory*, Lecture Notes in Control and Information Sciences, Springer-Verlag, Vol.116, 1989.
- [Tak. & Takah., 2] Takahashi, S. and Y. Takahara, *Logical Approach to Systems Theory*, Lecture Notes in Control and Information Sciences, Springer-Verlag, Vol.204, 1995.
- [Tar. & Yan., 1] Tarjan, R. E. and M. Yannakakis, "Simple linear-time

- algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs”, *SIAM Journal of Computing*, Vol.13, No.3, pp.566-579, 1984.
- [Taylor, 1] Taylor, A. E. *Plato: The man and his work*, Methuen & Co. Ltd., London, 1929.
- [Theod. et al., 1] Theodoulidis, C. B. Wangler, and P. Loucopoulos, “The Entity – Relationship – Time Model”, in *Conceptual Modelling, Data Bases and CASE*, ed. Loucopoulos & Zicari, John Wiley & Sons, Chichester, England, pp.87-114, 1992.
- [Theod. & Louc., 1] Theodoulidis, C., and P. Loucopoulos, “The Time Dimension in Conceptual Modelling”, *Information Systems*, Vol. 16, No. 3, 1991.
- [Usch. & Grun., 1] Uschold, M. and M. Gruninger, “Ontologies: principles, methods and applications” *The Knowledge Engineering Review*, Vol. 11, No. 2, pp.93-136, 1996.
- [van der Aalst, 1] van der Aalst, W. M. P. “Putting high-level Petri nets to work in industry”, *Computers in Industry*, Vol.25, No.1, pp.45-54, 1994.
- [van der Aalst, 2] van der Aalst, W. M. P. “A class of Petri nets for modelling and analyzing business processes”, *Computing Science Reports 95/26*, Eindhoven University of Technology, Eindhoven, 1995.
- [van der Aalst, 3] van der Aalst, W. M. P. “Three Good Reasons for Using a Petri-net-based Workflow Management System”,

- Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*, S. Navathe and T. Wakayama (eds.), pp. 179-201, Cambridge, Massachusetts, Nov. 1996.
- [van der Aalst, 4] van der Aalst, W. M. P. "Petri-net-based Workflow Management Software", *Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems*, A. Sheth (ed.), Athens, Georgia, May 1996.
- [van der Aalst, 5] van der Aalst, W. M. P. "Verification of Workflow Nets", *Application and Theory of Petri Nets 1997*, P. Azema and G. Balbo (eds.), Vol. **1248** of *Lecture Notes in Computer Science*, pp. 407-426, Springer – Verlag, Berlin, 1997.
- [van der Aalst, 6] van der Aalst, W. M. P. "The Application of Petri Nets to Workflow Management", *Journal of Circuits, Systems and Computers*, Vol. **8**, No. **1**, pp. 21-66, 1998.
- [van der Aalst, 7] van der Aalst, W. M. P. "Logistics: A systems oriented approach", *Proceedings of the 3rd International Working Conference on Dynamic Modelling and Information Systems*, pp. 169-189, the Netherlands, June 1992.
- [van der Aalst & Hee, 1] van der Aalst, W. M. P. and K. M. Hee, "Business process redesign: A Petri-net-based approach", *Computers in Industry*, Vol. **29**, pp. 15-26, 1996.
- [van der Aalst *et al.*, 1] van der Aalst, W. M. P., K. M. Hee, and G. J. Houben, "Modelling workflow management systems with high-level Petri nets", *Proceedings of the 2nd Workshop on Computer-Supported, Cooperative Work, Petri nets and related formalisms*, G. D. Michelis, C. Ellis, and G. Memmi (eds.),

- pp. 31-50. 1994.
- [Varal. & Kursh., 1] Varalya, P. and A. B. Kurshanzki, *Discrete Event Systems: Models and Applications*, Springer-Verlag, 1988.
- [Vasil., 1] Vasileiadou, S. and N. Karcianas, "Modelling Approaches and System Theoretic Issues in the Integrated Operations of Industrial Processes", *Proceedings of ASI'99 Conference*, Sept. 22-24, 1999.
- [Venkat., 1] Venkatraman, N. *IT-Induced Business Reconfiguration, The Corporation of the 1990's – Information Technology and Organisational Transformation*, M. S. Morton (ed.), pp.122-158, 1991.
- [Verma, 1] Verma, T. S. "Causal networks: semantics and expressiveness", *Proceedings of the 3rd workshop on uncertainty in artificial intelligence*, pp.352-359, 1987.
- [Vermuri, 1] Vermuri, V. *Modelling of Complex Systems: An introduction*. Academic Press, New York, 1978.
- [Vesconi & Robles, 1] Vesconi, M. and J. Robles, "Fuzzy diagnosis of continuous processes", *Proc. 10th European Conf. on Artificial Intelligence*, John Wiley and Sons, Chichester, England, 1992.
- [Vesconi et al., 1] Vesconi, M., Y. Iwasaki, R. Fikes, and B. Chandrasekaran, "CFRL: A language for specifying the causal functionality of engineered devices", *Proceedings 11th National Conference on Artificial Intelligence*, Cambridge, MA., MIT Press, 1993.

- [von Neum. & Morg., 1] von Neumann, J. and O. Morgenstein, *Theory of games and economic behaviour*, Wiley, New York, 1953.
- [VonWright, 1] VonWright, G. H. *Norm and action – A logical enquiry*, New York: Routledge & Kegan Paul, 1963.
- [Wellman, 1] Wellman, M. P. “Fundamental concepts of qualitative probabilistic networks”, *Artificial Intelligence*, Vol.44, pp.257-303, 1990.
- [Wielinga *et al.*, 1] Wielinga, B., W. van de Velde, G. Schreiber, and H. Akkermans, *Expertise Model Definition Document*, CommonKADS Project Report, University of Amsterdam, June 1993.
- [Williams, 1] Williams, B. “Qualitative analysis of MOS circuits”, *Artificial Intelligence*, Vol.24, pp.281-346, 1984.
- [Wilson, 1] Wilson, B. *Systems: Concepts, Methodologies and Applications*, 2nd edition, John Wiley & Sons, Chichester, September 1996.
- [Winter. & Edwards, 1] Winterfeldt, D. and W. Edwards, *Decision analysis and behavioural research*, Cambridge University Press, 1986.
- [Wolst., 1] Wolstenholme, E. F. *System Enquiry: A System Dynamics Modelling Approach*, John Wiley & Sons, Chichester, 1990.
- [Woodger, 1] Woodger, J. H. *Biological Principles – A critical study*, Kegan Paul, New York, 1929.
- [Woods, 1] Woods, E. A. *The hybrid phenomena theory*, Ph.D. thesis,

- Technical University of Norway, Institute of Engineering Cybernetics, 1993.
- [Wym., 1] Wymore, A. W. *Model-based system engineering*, CRS Press, London, 1993.
- [Yip, 1] Yip, K. M. "Model simplification by asymptotic order of magnitude reasoning" *Artificial Intelligence*, Vol. 80, pp.309-348, 1996.
- [Yip, 2] Yip, K. M. "Model simplifications in fluid mechanics" *Proceedings AAAI-93*, Washington, DC, 1993.
- [Yu & Mylop., 1] Yu, E. S. K. and J. Mylopoulos, "Using Goals, Rules, and Methods to Support Reasoning in Business Process Reengineering" *Proceedings of the 27th Annual Hawaii Int. Conference on System Science*, Vol. 4, pp.234-243, 1994.
- [Yu & Mylop., 2] Yu, E. and J. Mylopoulos, "An Actor Dependency Model of Organisational Work – With Application to Business Process Reengineering" *Proc. Conference on Organisational Computing Systems (COOCS 93)*, Milpitas, Calif., Nov. 1993.
- [Yu, 1] Yu, E. "Modelling Organisations for Information Systems Requirements Engineering" *Proceedings of First IEEE Symposium on Requirements Engineering*, pp.34-41, California, 1993.
- [Zadeh, 1] Zadeh, L. "Fuzzy sets", *Information and Control*, Vol.8, pp.338-353, 1965.

APPENDIX

1. Qualitative Reasoning

Below, we have included some basic definitions as they appear in the book of [Kuipers, 5].

A *model* is a (small) finite description of an infinitely complex reality, constructed for the purpose of answering particular questions.

A *quantity* is a real-valued attribute of a physical object. *Landmark* values are the “natural joints” that break a continuous set of values into qualitatively distinct regions. A landmark value is a symbolic name for a particular real number, whose numerical value may or may not be known. For example, a natural set of qualitative regions for the temperature of water is defined by the following landmarks:

AbsoluteZero...Freezing...Boiling... ∞

A *qualitative value* is either equal to a landmark value, or refers to the open interval between adjacent landmarks. The *qualitative state* of a parameter consists of its ordinal relations with the landmark values (its qualitative value) and its direction of change [Dalle Molle *et al.*, 1].

Fuzzy values are qualitative descriptions without precise boundaries. For example, when describing values of a continuous scalar quantity such as the amount of water in a tank, there are no meaningful landmark values representing the boundaries between *low* and *normal*, or between *normal* and *high*. Fuzzy sets were originally developed by [Zadeh, 1] to formalise qualitative concepts without precise boundaries.

Qualitative representation is complex; models can be built at different levels of detail and for different purposes; values can be described in terms of landmarks or fuzzy descriptors; qualitative descriptions can be augmented with quantitative information. Qualitative reasoning provides a level of description between discrete state graphs and the continuous world [Kuipers, 5]: continuous change is described symbolically, but in a way that obeys the constraints of continuity. Recent work on causal reasoning builds on qualitative representations to clarify the roles and properties of causal knowledge ([Horn, 1], [Borch., 1], [Amsterdam, 1], [Vesconi *et al.*, 1], and [Iwasaki *et al.*, 1]).

Definition 1: A *quantity space* is a finite, totally ordered set of symbols, the *landmark values*: $l_1 < l_2 < \dots < l_k$

Definition 2: A time-point $t \in [a, b]$ is a *distinguished or landmark time-point* the reasonable function f if t is a boundary element of the set $\{t \in [a, b] / f(t) = x\}$, where $x \in \mathfrak{R}^*$ is represented by a landmark value of f .

Definition 3: The *qualitative value* of $f(t)$, $QV(f, t)$, with respect to the quantity space $l_1 < l_2 < \dots < l_k$, is the pair $\langle qmag, qdir \rangle$, where

$$qmag = \begin{cases} l_j & \text{if } f(t) = l_j, \text{ a landmark value} \\ l_{j+1} & \text{if } f(t) \in (l_j, l_{j+1}) \end{cases}$$

$$qdir = \begin{cases} inc & \text{if } f'(t) > 0 \\ std & \text{if } f'(t) = 0 \\ dec & \text{if } f'(t) < 0 \end{cases}$$

Definition 4: For adjacent distinguished time-points t_i and t_{i+1} define $QV(f, t_i, t_{i+1})$, the *qualitative value* of f on (t_i, t_{i+1}) , to be $QV(f, t)$ for any $t \in (t_i, t_{i+1})$.

Definition 5: The *qualitative behaviour* of f on $[a, b]$ is the sequence of qualitative values of f ,

$$QV(f, t_0), QV(f, t_0, t_1), QV(f, t_1), \dots, QV(f, t_{n-1}, t_n), QV(f, t_n),$$

alternating between qualitative values at distinguished time-points, and qualitative values on intervals between distinguished time-points.

Definition 6: A *qualitative differential equation (QDE)* is a tuple of four elements $\langle V, Q, C, T \rangle$, each of which will be defined below:

- V is a set of variables, each of which is a “reasonable” function of time;
 - Q is a set of quantity spaces, one for each variable in V ;
 - C is a set of constraints applying to the variables in V . Each variable in V must appear in some constraint;
 - T is a set of transitions, which are rules defining the boundary of the domain of applicability of the QDE.
- The state of a system at a time t is described in terms of the values of some set of variables $\{x, y, \dots\}$, each of which is a reasonable function of time. The relationships among these variables are expressed by qualitative constraints that hold for each t . The first few —*add*, *mult*, *minus*, *d/dt*, and *constant*— are familiar mathematical relations:

$$(\text{add } x \text{ y } z) \equiv x(t) + y(t) = z(t)$$

$$(\text{mult } x \text{ y } z) \equiv x(t)y(t) = z(t)$$

$$(\text{minus } x \text{ y}) \equiv y(t) = -x(t)$$

$$(d/dt \ x \ y) \equiv \frac{d}{dt} x(t)=y(t)$$

$$(\text{constant } x) \equiv \frac{d}{dt} x(t)=0$$

- The functional constraints $-M^+$ and M^- express incomplete, qualitative knowledge about a functional relationship. The function relating the variables is known only to lie in the class M^+ of monotonically increasing functions.

$$(M^+ \ x \ y) \equiv y(t)=f(x(t)), f \in M^+$$

$$(M^- \ x \ y) \equiv y(t)=-f(x(t)), f \in M^+$$

- For example:

The constraint level $=M^+$ (amount) in a model of a bathtub encapsulates the incompletely known geometry of the container:

((M^+ amount level) (0 0) (full top) (inf inf)))

The corresponding value pairs of the constraint encode the following information:

- (0, 0) means the amount=0 iff level=0. That is, the implied monotonic function $f \in M^+$ must pass through (0, 0).
- (full, top) constrains the meanings of the landmarks in their respective quantity spaces to refer to different aspects of the same event.
- (∞ , ∞) constrains the function $f \in M^+$ to eliminate the possibility of horizontal or vertical asymptote.

The domain of Signs

Zero (0) is the prototypical landmark, dividing the positive numbers (+) from the negative numbers (-) on the real line. This set of qualitative values is called the *Domain of Signs*:

$$S = \{+, 0, -\}.$$

In the following chart, we summarise the fundamental definitions of the sign of values in terms of intervals, the compact bracket notation used to avoid confusion with other symbols, the single-character abbreviated notation used where no confusion is possible, and two notations when signs are used for directions of change. The notations are mathematically equivalent "syntactic sugar" used for clarity in different contexts.

| Fundamental | Signs (Compact) | Signs (Abbreviated) | QSIM qdirs | Plot symbols |
|----------------------|-----------------|---------------------|------------|--------------|
| $(0, \infty)$ | [+] | + | inc | ↑ |
| 0 | [0] | 0 | std | ○ |
| $(-\infty, 0)$ | [-] | - | dec | ↓ |
| $(-\infty, +\infty)$ | [?] | ? | ign | * |

Sign-Valued Operators

Certain qualitative operators are applied to values in \mathcal{R}^* and return signs as values:

- $[x]_0 = \text{sign}(x) = \begin{cases} [+] & \text{if } x > 0 \\ [0] & \text{if } x = 0 \\ [-] & \text{if } x < 0 \end{cases}$
- $[x]$ as an abbreviation for $[x]_0$ is acceptable where no ambiguity is possible
- $[x]_{x_0} = \text{sign}(x - x_0)$, where x_0 serves as a reference value for the variable x .
- $\left[\dot{x} \right] = [dx/dt] = \text{sign}(dx/dt)$.
- $[x]_{\infty} = \begin{cases} [+] & \text{if } x = +\infty \\ [0] & \text{if } x \text{ is finite} \\ [-] & \text{if } x = -\infty \end{cases}$

Confluences

Confluences are equations in signs, $S' = \{+, 0, -, ?\}$, or qualitative expressions evaluating to signs, such as combinations of the qualitative operators $[x]_0$, $[x]_{x_0}$, and $\left[\dot{x} \right]$. Confluences are the form of qualitative constraints developed by [de Kleer & Brown, 1] and used in their qualitative physics.

For example, consider the familiar law $F = ma$, where we know that the mass m is positive and constant: $[m]_0 = [+]$ and $[m] = 0$. We can transform this equation into a confluence to draw several useful qualitative conclusions:

- Force and acceleration are in the same direction:

$$[F]_0 = [ma]_0 = [m]_0 [a]_0 = [a]_0$$

- A change to either force or acceleration results in a change to the other in the same direction:

$$[F] = [d(ma)/dt] = [m]_0 [a] + [a]_0 [m] = [a]$$

- Perturbations to force and acceleration (with no perturbation to mass: $[m]_* = [0]$) are in the same direction:

$$[F]_* = [ma]_* = [a]_0 [m]_* + [m]_0 [a]_* = [a]_*$$

Although confluences are similar in many ways to the qualitative constraints as we have defined them, they are not strictly identical. In particular, a monotonic function constraint, such as

((M+ amount level) (0 0)),

means that some monotonic function f exists, such that $\text{level} = f(\text{amount})$ and $f(0) = 0$. We know very little about f , but we do know that it represents a functional relationship.

The corresponding confluences are

$$[\text{level}]_0 = [\text{amount}]_0 \text{ and } [d(\text{level})/dt] = [d(\text{amount})/dt],$$

which say that the sign of the magnitude and the sign of the derivative of $\text{level}(t)$ and $\text{amount}(t)$ must always be identical. They do not, however, say that the two values are related by a function. Thus, observations of the relationship between $\text{amount}(t)$ and $\text{level}(t)$ at one time do not necessarily support conclusions about their relationship later.

2. Order-of-Magnitude Reasoning

Semantics of O(M) relations

A relation $A r B$ is equivalent to $(A/B) r 1$ and signifies an interval for the (A/B) ratio, as shown in Figure 2.1.

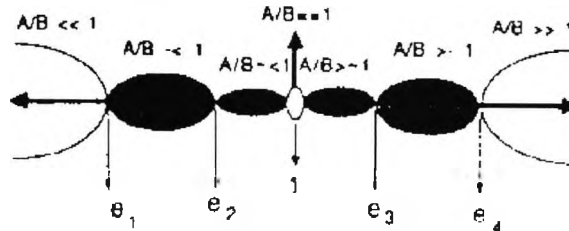


Figure appendix (2.1): Strict interpretation of the relation $A r B$ ([Mavro. & Steph., 2])

To sanction the symmetry of the relations:

$$A >\sim B \Leftrightarrow B \sim< A$$

$$A >> B \Leftrightarrow B << A$$

we impose the restrictions $e_3 = 1/e_2$ and $e_4 = 1/e_1$. Finally, to sanction the intuition that for $A > B > 0$:

$$A - B << B \Leftrightarrow A >\sim B$$

we further impose $e_3 - 1 = e_1$. Under this strict interpretation, the above constraints leave only one degree of freedom for the semantics of our relations, as depicted in Figure 2.2 [Mavro. & Steph., 2].

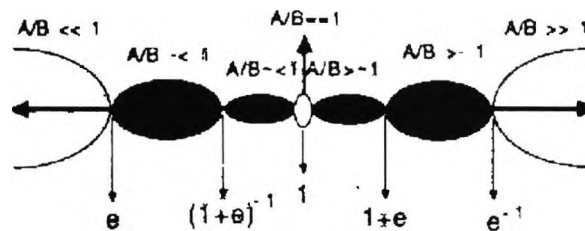


Figure appendix (2.2): Constrained strict interpretation of the relation $A r B$ ([Mavro. & Steph., 2])

The tolerance parameter e is left unspecified because it depends on the particular engineering activity. The value of e denotes the largest amount which we are willing to regard as “much smaller than 1”.

We further adopt a *heuristic interpretation* that replaces the boundary points of the intervals with regions and we construct two sets of primitive intervals: a set of non-exhaustive intervals and a set of overlapping ones (Figure 2.3).

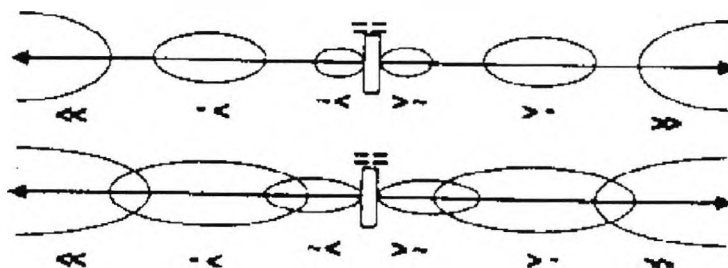


Figure appendix (2.3): Intervals used in the heuristic interpretation: (bottom) overlapping intervals; and (top) non-exhaustive intervals ([Mavro. & Steph., 2])

We then adopt the above inference convention [Mavro. & Steph., 1]: For every inference step, assume the antecedent relations to denote non-exhaustive intervals, but allow the consequent relations to denote overlapping intervals. Thus, when the consequents are used as antecedents at a later step their intervals are “shrunk” and therein lies the power and the risk. For the compound relations this mechanism refers only to the end points of the compound intervals. For more details refer to [Mavro. & Steph., 2].

3. Multilevel Flow Modelling

Basic MFM Function Concepts

We use the methodology of MFM to provide normative models. A *normative* model describes what the system should do in contrast to a descriptive model, which describes the actual behaviour of the systems [Lind, 2]. Both types of models represent abstractions of reality but they represent two different point of view of the system, i.e. different abstractions.

A *normative* model is focusing on the roles or functions of the different parts of the system in fulfilling requirements to the whole system. It is accordingly holistic of nature. A *descriptive* model is focusing on the parts of the system in order to describe the mechanisms or behaviours irrespective of the purpose of the whole. Their fundamental difference is the application of different strategies for abstracting relevant information out of the complexity of the real situation.

The MFM methodology distinguishes between two groups of modelling concepts; one group is related to the representation of goals and functions, the so-called *flow functions*, and the other group is dealing with the representation of how flow functions are realised in terms of physical components. Figure 3.1 represents the functional elements used for building MFM models. The achievement relation is a means-end relation, and the condition relation is used to describe that the existence of a function can be conditional on other plant states [Lind, 5].

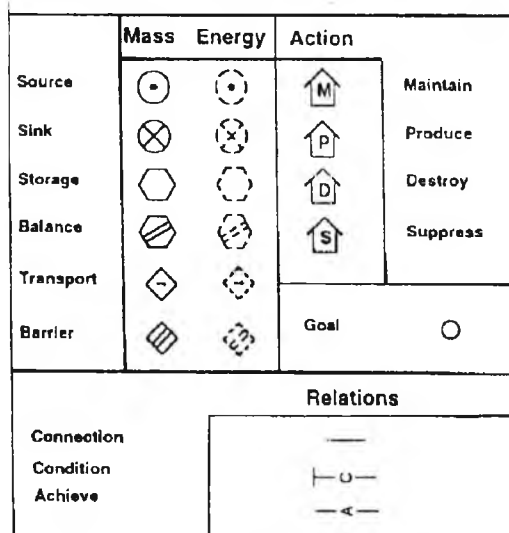


Figure appendix (3.1): MFM concepts and symbology ([Lind, 6])

Functions

Apart from the mass and energy flow functions, which are explicitly described in [Lind, 2], there are also two other types of functions. One type of functions is the *information flow functions* that comprise of the *observation*, *decision*, and *actor* functions. The basis for the modelling of decision functions are decision models such developed by [Rasmus., 1]. The symbol used to represent a decision function is shown below. Connections with observation and actor functions are indicated.

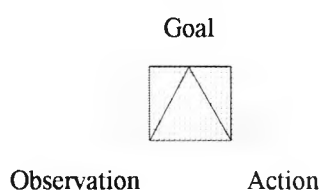


Figure appendix (3.2): The symbol of decision function

Another type of functions is the *organisational functions*, which are used to represent the functions of systems whose purpose is to support the functions of the system or to ensure that goals are met. The first category of functions is called *support* functions and the other category of functions is called *management* functions.

A *support* function represents the property of a system to provide the conditions necessary to allow another system to perform its function.



Figure appendix (3.3): The symbol used for support functions

The role of *management* functions is to provide connecting links between different functional levels in the model. Focussing on relations between means and ends we can characterise a management function as to provide a linkage between a plant goal to be achieved (i.e. to maintain a given temperature within limits) and the plant resources (the energy flow functions) available for control. The role of the management function is in this way to enter into a relation between a plant goal and the associated plant resources. However, the representation of a management function abstractly as a relation between a requirement/goal and a network of flow functions provides only structural information.



Figure appendix (3.4): The symbol used for management functions

Rules

Rule 1:

A flow structure includes only functions of the same type, i.e. mass, energy or information (they must not be mixed).

Rule 2:

Transport and barrier functions have two links.

Rule 3:

Barrier functions should only be linked with either balance or storage functions.

Rule 4:

Transport functions should only be linked with either source, sink, balance or storage functions.

Rule 5:

Storage and balance functions can have any number of links.

Rule 6:

Storage and balance functions should only be linked with transport or barrier functions.

Rule 7:

Source and sink functions have only one link.

Rule 8:

Source functions should only be linked with transport functions and only if the direction of flow is away from the source.

Rule 9:

Sink functions should only be linked with transport functions and only if the direction of flow is towards from the sink.

4. Bayesian Networks

Modelling Methods:

A. Undirected relations

It may happen that the model must contain dependence relations between variables A, B, C , say; but it is neither desirable nor possible to attach directions on them. (In that case the model is called a *chain graph*). A chain graph is an acyclic graph with both directed and non-directed links, where acyclic means that all cycles consist of only non-directed links (chain graphs are treated in depth by [Lauritzen, 1]). This difficulty may be overcome by using conditional dependence. Let $R(A, B, C)$ describe the relation in numbers from $[0, 1]$. Add a new variable D with two states y and n , and let A, B, C be parents of D (see Figure 4.1)

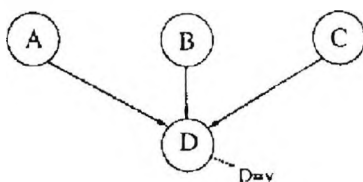


Figure appendix (4.1): A way to introduce undirected relations between A, B , and C ([Jensen, 2])

Let $P(D = y \mid A, B, C) = R(A, B, C)$, let $P(D = n \mid A, B, C) = 1 - R(A, B, C)$ and enter the evidence $D=y$.

B. Noisy or

Let A_1, \dots, A_n be binary variables listing all the causes of the binary variable B . Each event $A_i = y$ causes $B = y$ unless an inhibitor prevents it, and the probability for that is q_i . That is, $P(B = n \mid A_i = y) = q_i$ (see Figure 4.2). We assume that all inhibitors are independent. Then $P(B = n \mid A_1, A_2, \dots, A_n) = \prod_{j \in Y} q_j$ where Y is the set of indices for variables in the state y . For example,

$$P(B = y \mid A_1 = y, A_2 = y, A_3 = \dots A_n = n) = 1 - P(B = n \mid A_1 = y, A_2 = y, A_3 = \dots A_n = n) = 1 - q_1 q_2$$

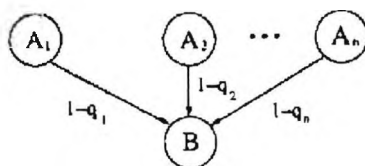


Figure appendix (4.2): The general situation for noisy or. q_i is the probability that the impact of A_i is inhibited ([Jensen, 2])

By assuming “noisy or”, the number of probabilities to estimate grows linearly with the number of parents. Note that the complementary construction to noisy is called *noisy and*. A set of causes shall be on in order to have an effect. However, the causes have random inhibitors which are mutually independent. Noisy or was first described by [Pearl, 3].

C. Causal independence

Let C_1, \dots, C_n be a list of variables all of which are causes of A . If one wishes to specify $P(A \setminus C_1, \dots, C_n)$ one might have a very large knowledge acquisition task ahead of him. Usually, it will only be possible to obtain partial specifications like $P(A \setminus C_i)$.

So now let C_1, \dots, C_n be the parents of A . C_1, \dots, C_n are *causally independent* if the following holds for each parent configuration (c_1, \dots, c_n) and all i : if at some time A is in the state a and the state of C_i is changed to c'_i then the probability distribution of A afterwards is a function of a , c_i and c'_i alone. If we add auxiliary variables C'_i and A' the condition is reflected in the conditional independencies which can be seen in Figure 4.3 [Jensen, 2]. The modelling of causal independence presented here is suggested by [Hecker., 1].

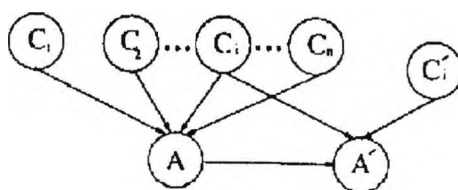


Figure appendix (4.3): A' is independent of $C_1, \dots, C_{i-1}, C_{i+1}, \dots, C_n$ given A and C_i ([Jensen, 2])

D. Divorcing

Noisy or, as well as causal independence represent simplifying assumptions to use when the space of parent configurations is too large. Both methods can be seen as special cases of a more general technique called *divorcing* (this technique was first used in MUNIN [Andreassen *et al.*, 1]).

The set of parents A_1, \dots, A_i for B is divorced from the parents A_{i+1}, \dots, A_n for B by introducing a mediating variable C , making C a child of A_1, \dots, A_i and a parent of B (see Figure 4.4)

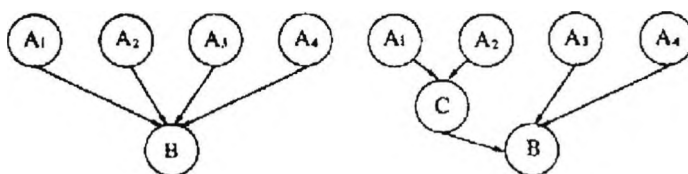


Figure appendix (4.4): A_1 and A_2 are divorced from A_3 and A_4 by introducing variable C ([Jensen, 2])

If all variables in Figure 4.4 are ternary, you will have to specify 81 distributions before divorcing and only 36 distributions after divorcing. Even if C turns out to require five states, the saving is considerable.

5. System Dynamics

Methods for building influence diagrams

1. The list extension method:

An influence diagram is a list of factors in a problem, together with arrows and signs showing the relationships between them. The list extension method is based on the rather obvious idea of starting with a small list and gradually extending it until a diagram emerges.

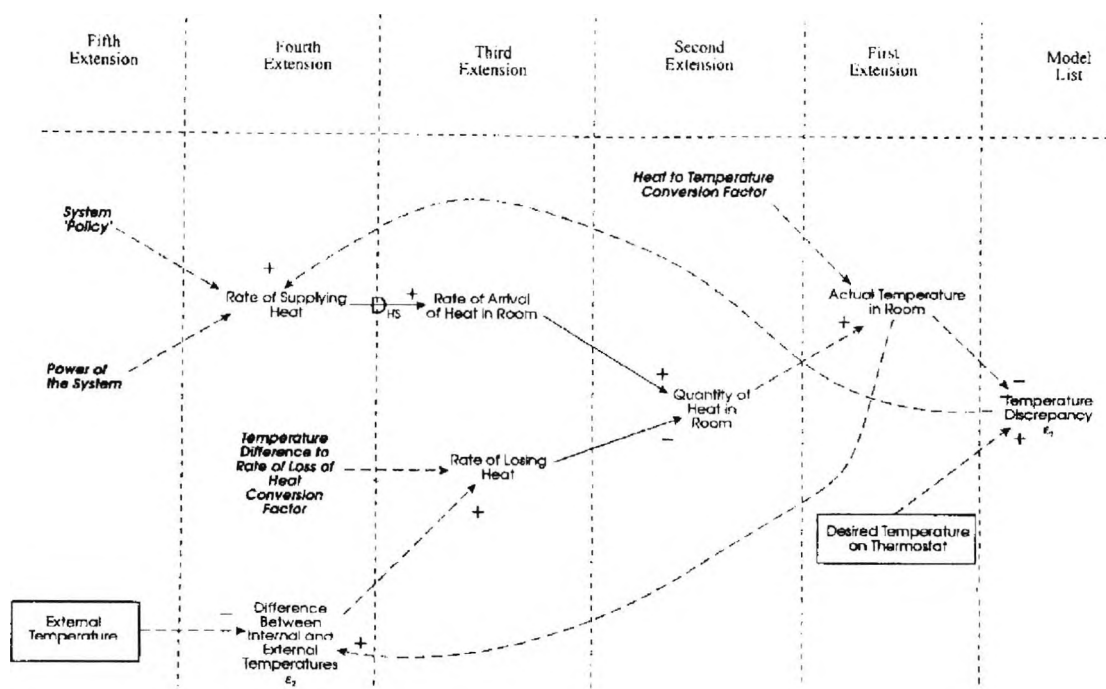


Figure appendix (5.1): The analysis of the problem of the central heating system through the List Extension method ([Coyle, 2])

In the first extension we write down the names of the variables, which directly and immediately influence the variable in the model list, adding the appropriate arrows and signs. The second extension continues the process by writing down the direct and immediate influences on the variables in the first extension. A chain of influence terminates when a parameter or an external force is encountered. Thus in Figure 5.1 we see links running from right to left and these emerge as extensions are built. This method is a good method for starting work on a problem, especially in one's early stages of developing skill in influence diagramming. Once again, the key to successful

modelling is to keep one's understanding of the model and what it says about the problem ahead of its size.

2. The entity/state/transition method:

In this method, we, first of all, identify the separate *entities* in the problem and for each of them, we write down all the possible *states* in which members of that entity can be. Then, we identify the *flows*, which can cause the state to increase or decrease and to check for the connections between flows (we represent any delays in flows). We identify the *controlling* flow rates that drive the system and represent the information and *action influences* on the controlling flow rates.

For example, if we consider the following narrative account, [Coyle, 2]:

...A firm of management consultants recruits new staff to fill any shortfall between the number of qualified consultants available for projects and the number needed to cope with the demand for the firm's services.

Recruits undergo a protracted training period before they become fully qualified consultants. When they complete their training, they join the consultancy team. Qualified consultants tend to stay with the firm for a period of 3 years before they leave to go into industry as managers. Some of the qualified consultants employed by the firm have to be used to supervise the trainees, on the scale of 1 supervisor to 10 trainees, and these supervisors are not available for fee-earning assignments. The trainees are not able to work on assignments. The total number of qualified working consultants the firm needs is driven by the demand for consultancy services...

By using the entity/state/transition method, we get the diagram:

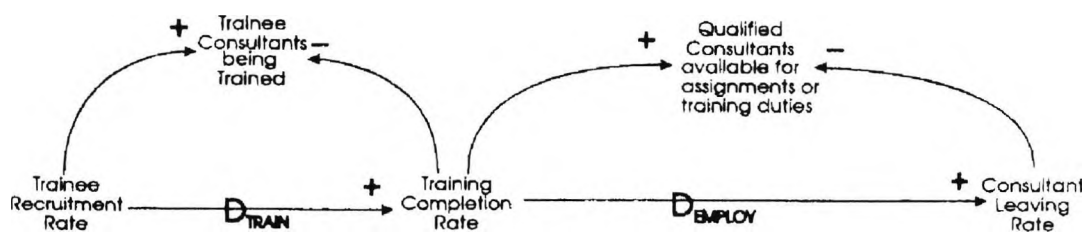


Figure appendix (5.2): The diagram of management consultants recruits ([Coyle, 2])

3. The common modules method:

Managed systems contain common modules of structure which recur in utterly different problems. Once one has learned to recognise these modules, building influence diagrams becomes quite easy.

On the question of which is the right method to use, it should be stressed out that an experienced System Dynamics modeller will often use none of the methods formally, rather a combination of the three methods. Regardless of the method, the fundamental point is that the diagram should portray the interaction of cause and effect within a system, because it is that interplay which will govern its ability to generate its own future behaviour.

4. The consulting firm's problem: Mathematical equations

Microsoft Excel spreadsheet showing the mathematical equations for a consulting firm's model. The spreadsheet is titled "Consulting firm's model.1TM" and is divided into two sections: (INITIALIZATION EQUATIONS) and (RUNTIME EQUATIONS).

(INITIALIZATION EQUATIONS)

- INIT Junior_Con = 25
- promo_time_jun_to_con = 3
- Turnover = GRAPH(TIME)
(1.00, 1.50), (2.00, 1.90), (3.00, 2.40), (4.00, 3.10), (5.00, 3.80), (6.00, 5.10), (7.00, 6.90), (8.00, 6.90)
- average_charging_rate = GRAPH(TIME)
(1.00, 2.30), (2.00, 2.45), (3.00, 2.60), (4.00, 2.72), (5.00, 2.80), (6.00, 2.94), (7.00, 3.15)
- required_manmonths = Turnover*1000/average_charging_rate
- effective_manyear = 11
- required_no_of_consultant_equivalent = required_manmonths/effective_manyear
- INIT External_Consultants = 24.6
- INIT Consultants = 10
- ratio_of_con_supervisors_to_junior = 0.15
- con_supervising_junior = Consultants*ratio_of_con_supervisors_to_junior
- con_available_for_projects = Consultants-con_supervising_junior
- INIT Senior_Con = 5
- ratio_of_sen_supervisors_to_junior = 0.04
- senior_con_supervising_junior = Senior_Con*ratio_of_sen_supervisors_to_junior
- senior_con_available_for_projects = Senior_Con-senior_con_supervising_junior
- productivity_of_jun_&_sup = 0.5
- equivalent_consultant_workforce = External_Consultants+con_available_for_projects+senior_con_available_for_projects+productivity_of_jun_&_sup*(Junior_Con+con_supervising_junior+senior_con_supervising_junior)
- manpower_discrepancy = required_no_of_consultant_equivalent - equivalent_consultant_workforce
- external_hiring_ratio = 0.3
- junior_quit_fraction = 0.3
- junior_quitting = Junior_Con*junior_quit_fraction
- con_quitting_frac = 0.15
- con_quitting = Consultants*con_quitting_frac
- senior_quit_frac = 0.05
- promo_time_con_to_senior = 4
- promoting_to_senior = Consultants/promo_time_con_to_senior
- senior_quitting = Senior_Con*senior_quit_frac
- external_quitting_frac = 0.2
- quitting externals = External_Consultants*external_quitting_frac
- hiring = manpower_discrepancy/(productivity_of_jun_&_sup+external_hiring_ratio)+productivity_of_jun_&_sup*junior_quitting+con_quitting*((1-ratio_of_con_supervisors_to_junior)+ratio_of_con_supervisors_to_junior*productivity_of_jun_&_sup)+senior_quitting*((1-ratio_of_sen_supervisors_to_junior)+ratio_of_sen_supervisors_to_junior*productivity_of_jun_&_sup)+quitting_externals
- promoting_to_con = Junior_Con/promo_time_jun_to_con
- hiring_externals = external_hiring_ratio*hiring

(RUNTIME EQUATIONS)

- Junior_Con(t) = Junior_Con(t-dt) + (hiring - promoting_to_con - junior_quitting) * dt
- External_Consultants(t) = External_Consultants(t-dt) + (hiring_externals - quitting_externals) * dt
- Consultants(t) = Consultants(t-dt) + (promoting_to_con - promoting_to_senior - con_quitting) * dt
- Senior_Con(t) = Senior_Con(t-dt) + (promoting_to_senior - senior_quitting) * dt
- Turnover = GRAPH(TIME)
(1.00, 1.50), (2.00, 1.90), (3.00, 2.40), (4.00, 3.10), (5.00, 3.80), (6.00, 5.10), (7.00, 6.90), (8.00, 6.90)
- average_charging_rate = GRAPH(TIME)
(1.00, 2.30), (2.00, 2.45), (3.00, 2.60), (4.00, 2.72), (5.00, 2.80), (6.00, 2.94), (7.00, 3.15)
- required_manmonths = Turnover*1000/average_charging_rate
- required_no_of_consultant_equivalent = required_manmonths/effective_manyear
- con_supervising_junior = Consultants*ratio_of_con_supervisors_to_junior
- con_available_for_projects = Consultants-con_supervising_junior
- senior_con_supervising_junior = Senior_Con*ratio_of_sen_supervisors_to_junior
- senior_con_available_for_projects = Senior_Con-senior_con_supervising_junior
- equivalent_consultant_workforce = External_Consultants+con_available_for_projects+senior_con_available_for_projects+productivity_of_jun_&_sup*(Junior_Con+con_supervising_junior+senior_con_supervising_junior)
- manpower_discrepancy = required_no_of_consultant_equivalent - equivalent_consultant_workforce
- junior_quitting = Junior_Con*junior_quit_fraction
- con_quitting = Consultants*con_quitting_frac
- promoting_to_senior = Consultants/promo_time_con_to_senior
- senior_quitting = Senior_Con*senior_quit_frac
- quitting_externals = External_Consultants*external_quitting_frac
- hiring = manpower_discrepancy/(productivity_of_jun_&_sup+external_hiring_ratio)+productivity_of_jun_&_sup*junior_quitting+con_quitting*((1-ratio_of_con_supervisors_to_junior)+ratio_of_con_supervisors_to_junior*productivity_of_jun_&_sup)+senior_quitting*((1-ratio_of_sen_supervisors_to_junior)+ratio_of_sen_supervisors_to_junior*productivity_of_jun_&_sup)+quitting_externals
- promoting_to_con = Junior_Con/promo_time_jun_to_con
- hiring_externals = external_hiring_ratio*hiring

6. Matlab programs for the discrete-time model

1. Check state transition

```

function [x_new,w] = state_transition(x_old,u,param)
% function [x_new,w] = state_transition(x_old,u,param)

% INPUT variables:
% x_old          old state vector
% u              control input, number of juniors hired in
this period (integer in range 0 <= u <= u_max)
% param         vector of model parameters (both time varying
and time invariant)
%
% OUTPUT variables:
% x_new         new state vector
% w             output variable: equivalent workforce at
consultant level

%-----
% Decode parameters

alpha=param(1);      % Junior consultants growth parameter
beta=param(2);      % Consultants growth parameter
gamma=param(3);     % Junior -> Consultants promotion parameter
delta=param(4);     % Senior consultants growth parameter
epsilon=param(5);   % Consultants -> Seniors promotion parameter
omega=param(6);     % External consultants growth parameter
mu=param(7);        % Number of promotion time-periods: Juniors ->
Consultants
lambda=param(8);    % Number of promotion time-periods: Consultants
-> Seniors
n=param(9);         % Fraction of Externals/Juniors hired during
this period 0<=n<=1
m=param(10);        % percentage of consultants working only on
projects
l=param(11);        % percentage of seniors working only on
projects
d=param(12);        % relative productivity factor of juniors
p=param(13);        % productivity factor of supervisors
juniors_max=param(14); % maximum number of juniors employed (saturates
juniors)
consult_max=param(15); % maximum number of consultants employed
(saturate consultants)
seniors_max=param(16); % maximum number of seniors employed (saturate
seniors)
extern_max=param(17); % maximum number of externals employed
(saturate externals)
u_max=param(18);    % maximum number of juniors that can be hired
for any period
                    % (integer in range 0 <= u_max <= juniors_max)

%-----
% CHECK compatibility of inputs/parameters

[rx,cx]=size(x_old);
if cx ~= 1 | rx ~= mu+lambda+4 % state-size incompatibility
    disp('error in state_transition ...');

```

```

    x_new=[];w=[];
    return
end

if u_max > juniors_max
    disp('error in state_transition ...');
    x_new=[];w=[];
    return
end

if u < 0 | u > u_max | u ~= round(u) | mu <= 0 | mu ~= round(mu) |
lambda <= 0 | lambda ~= round(lambda)
    disp('error in state_transition ...');
    x_new=[];w=[];
    return
end

if n < 0 | n > 1 | m < 0 | m > 1 | l < 0 | l > 1 | d < 0 | d > 1 | p <
0 | p > 1
    % Check that fractions are between 0 and 1
    disp('error in state_transition ...');
    x_new=[];w=[];
    return
end

for i=1:mu+1
    % check that current and past number of juniors are positive
integers <= juniors_max
    if x_old(i) < 0 | x_old(i) > juniors_max | x_old(i) ~=
round(x_old(i))
        disp('error in state_transition ...');
        x_new=[];w=[];
        return
    end
end

for i=mu+2:mu+lambda+2
    % check that current and past number of consultants are positive
integers <= consult_max
    if x_old(i) < 0 | x_old(i) > consult_max | x_old(i) ~=
round(x_old(i))
        disp('error in state_transition ...');
        x_new=[];w=[];
        return
    end
end

if x_old(mu+lambda+3) < 0 | x_old(mu+lambda+3) > seniors_max |
x_old(mu+lambda+3) ~= round(x_old(mu+lambda+3))
    % check that current number of seniors are positive integers <=
seniors_max
    disp('error in state_transition ...');
    x_new=[];w=[];
    return
end

if x_old(mu+lambda+4) < 0 | x_old(mu+lambda+4) > extern_max |
x_old(mu+lambda+4) ~= round(x_old(mu+lambda+4))
    % Check that current number of externals are positive integers <=
extern_max
    disp('error in state_transition ...');

```

```

    x_new=[];w=[];
    return
end

%-----
% Find current number of Juniors, Consultants, Seniors and Externals

juniors=x_old(1);
consultants=x_old(mu+2);
seniors=x_old(mu+lambda+3);
externals=x_old(mu+lambda+4);

%-----
% Find current output, i.e. equivalent workforce at consultant-level
during current period

e=1-m; % percentage of supervising consultants
f=1-l; % percentage of supervising seniors

super_consultants=round(e*consultants); % supervising consultants
non_super_consultants=consultants-super_consultants; % non-supervising
consultants
super_seniors=round(f*seniors); % supervising seniors
non_super_seniors=seniors-super_seniors; % non-supervising seniors

w = externals + non_super_consultants + non_super_seniors + ...
    p*d*juniors + p*(super_consultants+super_seniors); % current output
(actual workforce) at consultant-period level

%-----
% Find new state-vector

x_new=zeros(mu+lambda+4,1); % initialise new state-vector

new_juniors=round(alpha*juniors)+u; % new juniors
x_new(1) = min(new_juniors,juniors_max); % saturate juniors

if mu > 0
    x_new(2:1+mu)=x_old(1:mu); % delayed state (juniors)
end

new_consultants=round(beta*consultants)+round(gamma*x_old(1+mu));
x_new(2+mu)=min(new_consultants,consult_max); % saturate consultants

if lambda > 0
    x_new(3+mu:2+mu+lambda)=x_old(2+mu:1+mu+lambda); % delayed state
(consultants)
end

new_seniors=round(delta*seniors)+round(epsilon*x_old(mu+lambda+2));
x_new(mu+lambda+3)=min(new_seniors,seniors_max); % saturate seniors

new_externals=round(omega*externals)+round(n*u);
x_new(mu+lambda+4)=min(new_externals,extern_max); % saturate externals

%
%----- end of state_transition.m -----
%-----

```

2. Dynamic programming over finite horizon

```

% Dp1
% Dynamic Programming over finite horizon
%-----
% Encode parameters
param=zeros(1,18);

alpha=0.4;
param(1)=alpha;           % Junior consultants growth parameter

beta=0.4;
param(2)=beta;           % Consultants growth parameter

gamma=0.3;
param(3)=gamma;         % Junior -> Consultants promotion parameter

delta=0.4;
param(4)=delta;         % Senior consultants growth parameter

epsilon=0.35;
param(5)=epsilon;       % Consultants -> Seniors promotion parameter

omega=0.45;
param(6)=omega;         % External consultants growth parameter

mu=1;
param(7)=mu;             % Number of promotion time-periods: Juniors ->
Consultants

lambda=1;
param(8)=lambda;        % Number of promotion time-periods: Consultants
-> Seniors

n=0.4;
param(9)=n;             % Fraction of Externals/Juniors hired during
this period 0<=n<=1

m=0.5;
param(10)=m;           % percentage of consultants working only on
projects

l=0.5;
param(11)=l;           % percentage of seniors working only on
projects

d=0.6;
param(12)=d;           % relative productivity factor of juniors

p=0.5;
param(13)=p;           % productivity factor of supervisors

juniors_max=15;
param(14)=juniors_max; % maximum number of juniors employed (saturates
juniors)

consult_max=10;
param(15)=consult_max; % maximum number of consultants employed
(saturate consultants)

seniors_max=10;

```

```

param(16)=seniors_max; % maximum number of seniors employed (saturate
seniors)

extern_max=10;
param(17)=extern_max; % maximum number of externals employed
(saturate externals)

u_max=4;
param(18)=u_max; % maximum number of juniors that can be hired
for any period
% (integer in range 0 <= u_max <= juniors_max)
%-----
% State-vector (initial conditions)

x0=zeros(mu+lambda+4,1); % initialise state-vector
%
x0(1)=4; % juniors
x0(2)=0; % delayed juniors
x0(3)=4; % consultants
x0(4)=0; % delayed consultants
x0(5)=4; % seniors
x0(6)=4; % externals
%-----
% Generate state-transitions for indicated control sequence:
u_ind=[0 1 2 3 3]; % control sequence

[x1,w0] = state_transition(x0,u_ind(1),param);
[x2,w1] = state_transition(x1,u_ind(2),param);
[x3,w2] = state_transition(x2,u_ind(3),param);
[x4,w3] = state_transition(x3,u_ind(4),param);
[x5,w4] = state_transition(x4,u_ind(5),param);
[x6,w5] = state_transition(x5,0,param); % need only w5
%-----
% Demand vector
%
r=[w0 w1 w2 w3 w4 w5];
%-----

Cont_1=zeros(u_max+1,mu+lambda+8);
for u=0:u_max
    [x_new,w]=state_transition(x0,u,param);
    Cont_1(u+1,1:mu+lambda+4)=x_new'; % state x1
    Cont_1(u+1,mu+lambda+5)=u; % control u0
    Cont_1(u+1,mu+lambda+6)=abs(w-r(1)); % cost |w0-r0|
end

nr1=size(Cont_1,1);
Cont_2=zeros((u_max+1)*nr1,mu+lambda+8);
for i=1:nr1
    x=(Cont_1(i,1:mu+lambda+4))';
    for u=0:u_max
        [x_new,w]=state_transition(x,u,param);
        Cont_2((i-1)*(u_max+1)+u+1,1:mu+lambda+4)=x_new'; % state x2
        Cont_2((i-1)*(u_max+1)+u+1,mu+lambda+5)=u; % control u1
        Cont_2((i-1)*(u_max+1)+u+1,mu+lambda+6)=abs(w-r(2)); % cost
|w1-r1|
    end
end

nr2=size(Cont_2,1);
Cont_3=zeros((u_max+1)*nr2,mu+lambda+8);

```

```

for i=1:nr2
    x=(Cont_2(i,1:mu+lambda+4))';
    for u=0:u_max
        [x_new,w]=state_transition(x,u,param);
        Cont_3((i-1)*(u_max+1)+u+1,1:mu+lambda+4)=x_new'; % state x3
        Cont_3((i-1)*(u_max+1)+u+1,mu+lambda+5)=u; % control u2
        Cont_3((i-1)*(u_max+1)+u+1,mu+lambda+6)=abs(w-r(3)); % cost
    |w2-r2|
    end
end

nr3=size(Cont_3,1);
Cont_4=zeros((u_max+1)*nr3,mu+lambda+8);
for i=1:nr3
    x=(Cont_3(i,1:mu+lambda+4))';
    for u=0:u_max
        [x_new,w]=state_transition(x,u,param);
        Cont_4((i-1)*(u_max+1)+u+1,1:mu+lambda+4)=x_new'; % state x4
        Cont_4((i-1)*(u_max+1)+u+1,mu+lambda+5)=u; % control u3
        Cont_4((i-1)*(u_max+1)+u+1,mu+lambda+6)=abs(w-r(4)); % cost
    |w3-r3|
    end
end

nr4=size(Cont_4,1);
Cont_5=zeros((u_max+1)*nr4,mu+lambda+6);
for i=1:nr4
    x=(Cont_4(i,1:mu+lambda+4))';
    for u=0:u_max
        [x_new,w]=state_transition(x,u,param);
        Cont_5((i-1)*(u_max+1)+u+1,1:mu+lambda+4)=x_new'; % state x5
        Cont_5((i-1)*(u_max+1)+u+1,mu+lambda+5)=u; % control u4
        Cont_5((i-1)*(u_max+1)+u+1,mu+lambda+6)=abs(w-r(5)); % cost
    |w4-r4|
    end
end

% Terminal cost period x5->x6: This depends only on x5 and r5 so use
fictitious input (say 0)
%nr5=nr4*(u_max+1); % state dimension of x5
%terminal_cost=zeros(nr5,1);
%for i=1:nr5
%    x=Cont_5(i,1:mu+lambda+4);
%    [x_new,w]=state_transition(x,0,param); % get w5
%    terminal_cost(i)=abs(w-r(6)); % |w5-r5|
%end
% -----
% DP recursions

for i=1:nr4
    temp=Cont_4(i,mu+lambda+6)*ones(u_max+1,1)+Cont_5((i-
1)*(u_max+1)+1:i*(u_max+1),mu+lambda+6);
    [min_cost,index]=min(temp);
    Cont_4(i,mu+lambda+7)=min_cost;
    Cont_4(i,mu+lambda+8)=index;
end
%
for i=1:nr3
    temp=Cont_3(i,mu+lambda+6)*ones(u_max+1,1)+Cont_4((i-
1)*(u_max+1)+1:i*(u_max+1),mu+lambda+7);

```

```

    [min_cost,index]=min(temp);
    Cont_3(i,mu+lambda+7)=min_cost;
    Cont_3(i,mu+lambda+8)=index;
end
%
for i=1:nr2
    temp=Cont_2(i,mu+lambda+6)*ones(u_max+1,1)+Cont_3((i-
1)*(u_max+1)+1:i*(u_max+1),mu+lambda+7);
    [min_cost,index]=min(temp);
    Cont_2(i,mu+lambda+7)=min_cost;
    Cont_2(i,mu+lambda+8)=index;
end
%
for i=1:nr1
    temp=Cont_1(i,mu+lambda+6)*ones(u_max+1,1)+Cont_2((i-
1)*(u_max+1)+1:i*(u_max+1),mu+lambda+7);
    [min_cost,index]=min(temp);
    Cont_1(i,mu+lambda+7)=min_cost;
    Cont_1(i,mu+lambda+8)=index;
end
%-----
% Reconstruct optimal policy and state-transition
%
[optimal_cost,i0]=min(Cont_1(:,mu+lambda+7));

u_star_0=i0-1; % optimal policy u0
[x_star_1,w_star_0]=state_transition(x0,u_star_0,param);
%
i1=Cont_1(i0,mu+lambda+8);
u_star_1=i1-1; % optimal policy u1
[x_star_2,w_star_1]=state_transition(x_star_1,u_star_1,param);
%
i2=Cont_2((i0-1)*nr1+i1,mu+lambda+8);
u_star_2=i2-1; % optimal policy u2
[x_star_3,w_star_2]=state_transition(x_star_2,u_star_2,param);
%
i3=Cont_3((i0-1)*nr2+(i1-1)*nr1+i2,mu+lambda+8);
u_star_3=i3-1; % optimal policy u3
[x_star_4,w_star_3]=state_transition(x_star_3,u_star_3,param);
%
%i4=Cont_4((i0-1)*nr3+(i1-1)*nr2+(i2-1)*nr1+i3,mu+lambda+8);
%u_star_4=i4-1; % optimal policy u4
%[x_star_5,w_star_4]=state_transition(x_star_4,u_star_4,param);
%
%----- end of dp1.m -----
-----

```

7. Petri Nets

A Petri Net is composed of four parts: a set of *places* P , a set of *transitions* T , an *input* function I , and an *output* function O . The input and output functions relate transitions and places. The input function I is a mapping from a transition t_j to a collection of places $I(t_j)$, known as the *input places* of the transition. The output function O is a mapping from a transition t_j to a collection of places $O(t_j)$, known as the *output places* of the transition. A *Petri net structure* $C = (P, T, I, O)$ is defined by its places, transitions, input function, and output function. An example of a Petri net structure is given below. As we can see, a Petri Net structure is represented as a 4-tuple (see Figure 7.1):

$$\begin{aligned}
 C &= (P, T, I, O) \\
 P &= \{p_1, p_2, p_3, p_4, p_5\} \\
 T &= \{t_1, t_2, t_3, t_4\} \\
 I(t_1) &= \{p_1\} & O(t_1) &= \{p_2, p_3, p_5\} \\
 I(t_2) &= \{p_2, p_3, p_5\} & O(t_2) &= \{p_5\} \\
 I(t_3) &= \{p_3\} & O(t_3) &= \{p_4\} \\
 I(t_4) &= \{p_4\} & O(t_4) &= \{p_2, p_3\}
 \end{aligned}$$

Figure appendix (7.1): A Petri net structure

A place p_i is an input place of a transition t_j , if $p_i \in I(t_j)$, and p_i is an output place if $p_i \in O(t_j)$. The inputs and outputs of the transition are *bags* of places, where bag is a generalisation of sets, which allows multiple occurrences of an element in it. The use of bags, rather than sets, for the inputs and outputs of a transition allows a place to be multiple input or multiple output of a transition. We define a transition t_j to be an input of a place p_i , if p_i is an output of t_j . A transition t_j is an output of place p_i , if p_i is an input of t_j .

A Petri net is a *multigraph*, since it allows multiple arcs from one node of the graph to another. The Petri net graph equivalent to the structure from the above example is shown in Figure 7.2

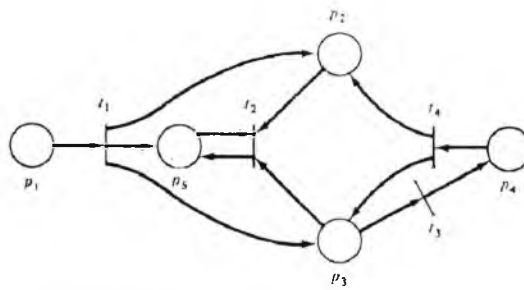


Figure appendix (7.2) : A Petri net graph equivalent to the structure of figure 7.1 ([Peterson, 2])

An example of a marked Petri net and the whole procedure of firing are shown below in Figures 7.3, 7.4, 7.5:

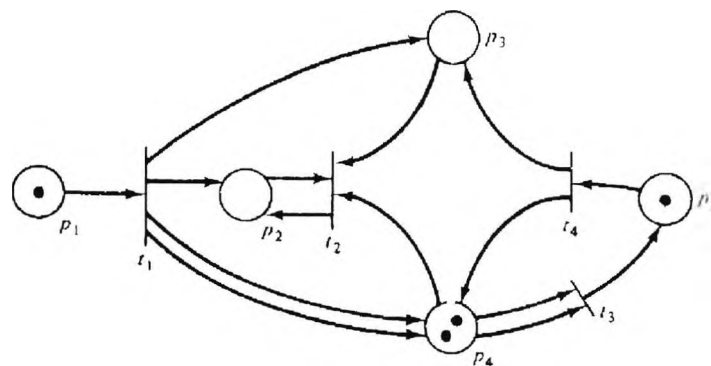


Figure appendix (7.3): A marked Petri net to illustrate the firing rules. Transitions t_1 , t_3 , and t_4 are enabled ([Peterson, 2])

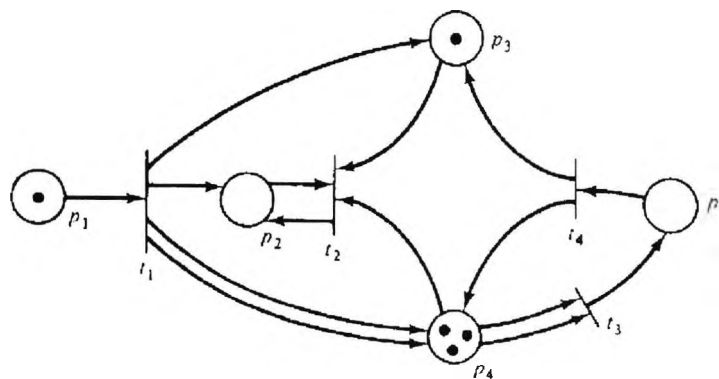


Figure appendix (7.4): The marking resulting from firing transition t_4 ([Peterson, 2])

In Figure 7.3 we can see that only transition t_1 , t_3 , and t_4 are enabled. Transition t_2 is not enabled because there is no token in either place p_2 or p_3 , which are both inputs of transition t_2 . Since t_1 , t_3 , and t_4 are enabled, each of them may fire. Assume that the transition t_4 fires. This means that a token is removed from each input

and is deposited in each output. After firing transition t_4 we have one token in input place p_3 and one more in p_4 , as Figure 7.4 shows.

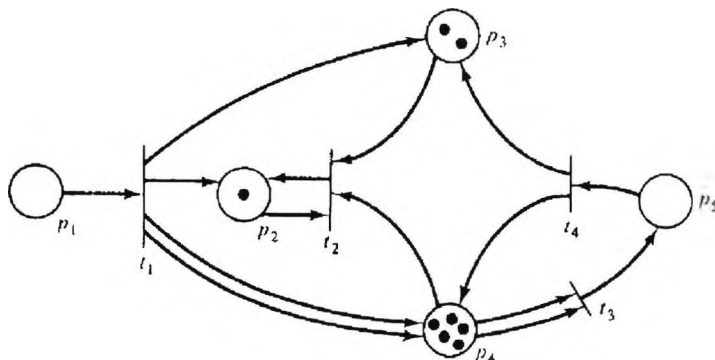


Figure appendix (7.5): The marking resulting from firing transition t_1 ([Peterson, 2])

In this marked Petri net the transition t_1 and t_3 are enabled and may fire. If t_1 fires, we have the marked Petri net shown in Figure 7.5. Firing transition t_1 has removed the token p_1 and has deposited one token in p_2 , one more in p_3 , and two more in p_4 , since it is a multiple output of transition t_1 . Transition firings can continue as long as there exists at least one enabled transition. When there are no enabled transitions, the execution *halts*.

LIST OF PUBLICATIONS

Topintzi, E., N. Karcianas and E. Nistazakis, "System Dynamics: Analysis and Decision-Making Process of a Human Resources Management problem", submitted to appear shortly in *Knowledge based and Intelligent Decision Support Systems*, special issue of the journal *Studies in informatics and Control*.

Topintzi, E. and N. Karcianas, "Systems and Dynamics of Human Resources in a Consulting Company", *ASI'2000 – IIMB'2000*, September 2000, Bordeaux, France.

Manders, C. and E. Topintzi "Icebergs, Pubs and the Wheel World", published in *Knowledge Management* magazine, Vol. 3, Issue 5, February 2000.

Modelling Methodologies for Behavioural Aspects of Business Processes: A Review of Methodologies, PRIMA II Project Report, ESPRIT project (20775), City University, Aug. 1999.

Implementation of Knowledge Diagnosis Method, KNOWNET Project Report ESPRIT project (28928), **D 10**, March 2000.

Knowledge transformation at User Companies, KNOWNET Project Report ESPRIT project (28928), **D 11.1**, March 2000.

Performance Evaluation at NAI Gooch Webster, KNOWNET Project Report ESPRIT project (28928), **D 12.1**, March 2000.