2005-01-01

Development of a Process Modelling System for Simulation

John Ryan
Technological University Dublin, john.ryan@tudublin.ie

Funder: simulation, process modelling, system

Development of a Process Modelling System for Simulation

John Ryan Dr.
DIT, john.ryan@dit.ie

— Use Licence —

Attribution-NonCommercial-ShareAlike 1.0

You are free:

- to copy, distribute, display, and perform the work
- to make derivative works

Under the following conditions:

- Attribution.
  You must give the original author credit.
- Non-Commercial.
  You may not use this work for commercial purposes.
- Share Alike.
  If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

For any reuse or distribution, you must make clear to others the license terms of this work. Any of these conditions can be waived if you get permission from the author.

Your fair use and other rights are in no way affected by the above.

Development of a Process Modelling System for Simulation

By
John A. Ryan (B.Eng.)

This thesis is submitted in accordance with the requirements of the University of Limerick for the degree of Doctor of Philosophy.

Supervisor: Dr. Cathal Heavey

Submitted to the University of Limerick, January 2005

Declaration

Project Title: Development of a Process Modelling System for Simulation.

Supervisor: Dr. Cathal Heavey

This thesis is presented in partial fulfilment of the requirements for the degree of Doctor of Philosophy. It is entirely my own work and has not been submitted to any other university or higher education institution, or for any other academic award in this university. Where use has been made of the work of other people it has been fully acknowledged and fully referenced.

Signature

John A. Ryan.

Date: Jan, 2005

Dedication

To my Parents

I would like to dedicate this thesis to my parents, my Mother and my late Father, whose sacrifices have provided me with this opportunity at the University of Limerick.

Acknowledgements

I would like to thank the following people who contributed each in their own way to the completion of this project:

My supervisor, Dr. Cathal Heavey, for the generous giving of his time when I needed it most and also for his supervision, guidance and patience over the past years.

Dr. Peter Williams For his advice on the completion of this thesis.

Dr. Kevin Griffin Proof-reader and paper folder supreme.

The now defunct Area 51, John, Eilis, Shahab, Shafi, Dino, James and Lenny for ensuring a well rounded postgraduate education.

My colleagues in DIT Cathal Brugha street, for the support and encouragement over these past years.

The two Martins for their advice, support and for lighting for me a path through C++.

Hugh for a friendship that has endured.

Enda for the many runs along the bank, friendship, pints and craic over the years.

David and Darragh a promise kept, thanks for the friendship and support over the years guys.

Mr. and Mrs. Gavin For always welcoming me into your home and the many years of IT support and friendship.

Tom Kinsella for many hours of advice, friendship and craic.

Tony, Mike and Alan for the hurling, footie, craic and friendship.

To all in Munitearas Iosa for the breaks welcome and unwelcome, for constantly challenging me in new ways and most of all for the lifelong friendships made.

The football gang for the Thursdays on the astro turf lads, great distractions and great memories.

The lads, to mention you all by name would result in a volume comparable to this thesis but all of you who helped in any way through the last years a sincere thank you.

My Brother, Martin, and my Sister, Mary, for the support advice and love through the many years of study.

Elaine for patiently suffering through much of this thesis with me, and constantly being a source of support, love and advice, all my love always.

Abstract

By: John A. Ryan

This thesis details the development of a process modelling technique to aid a simulation model developer during the requirements gathering and conceptual modelling phases of a simulation project.

There are a number of process modelling techniques available that are capable of being used during such phases of a simulation project, however there is currently a lack of process modelling techniques developed specifically to aid a simulation model developer in capturing, representing and communicating information and systems issues to persons involved in the operation of discrete systems under investigation.

A detailed review of the literature related to techniques capable of supporting the pre-simulation phases of a simulation project is presented. The main conclusion of this review is that there is a specific lack of support available to aid a simulation model developer in the pre-coding phases of a simulation project. Currently there are no process modelling techniques available that specifically support the pre-simulation phases of a discrete event simulation project.

To attempt to overcome this shortfall the thesis discusses the development of a process modelling technique specifically developed to support the pre-simulation phases of a simulation project. Objectives in the development of this technique were to develop a technique that:

1. Is capable of capturing a detailed description of a Discrete Event System;
2. Has a low modelling burden and therefore is capable of being used by non specialists;
3. Presents modelling information at a high semantic level so that manufacturing personnel can rationalise with it;
4. Has good visualisation capabilities.

The technique developed is called Simulation Activity Diagrams (SADs). To demonstrate the ability of the SAD technique to model discrete event information a prototype process modelling tool, Process Modelling for Simulation (PMS) was developed. An evaluation of the SAD technique is then presented through of a number of real and conceptual discrete event systems used to examine the techniques ability to accurately model information along with its ease of use and modelling accuracy. The thesis concludes that more research is required in validating and developing SADs and in developing other techniques in the pre-simulation area.

Contents

DECLARATION I
DEDICATION II
ACKNOWLEDGEMENTS III
ABSTRACT IV
CONTENTS V
LIST OF TABLES XI

CHAPTER 1: INTRODUCTION 1

1.1 INTRODUCTION 1
1.2 THE PROCESS OF SIMULATION 2
1.2.1 Requirements Gathering 4
1.3 Research Objectives 6

CHAPTER 2: PROCESS MODELLING 10

2.1 INTRODUCTION 10
2.2 PROCESS MODELLING TECHNIQUES 11
2.2.1 Petri- Nets 12
2.2.2 Activity Cycle Diagrams 13
2.2.3 Discrete Event System Specification (DEVS) 15
2.2.4 Unified Modelling Language 17
2.2.5 Role Activity Diagrams 22
2.2.6 The GRAI Method 25
2.2.7 IEM modelling method 31
2.2.8 Event driven process chains 38
2.2.9 IDEF Suite of Modelling Methods 43
2.3 PROCESS MODELLING TOOLS 52
2.3.1 MOOGO 52
2.3.2 ARIS toolset 55
2.3.3 ProSim 59
2.4 CONCLUSIONS 62

CHAPTER 3 SAD DEVELOPMENT PROCESS 66

CHAPTER 4: SIMULATION ACTIVITY DIAGRAMS 76

4.1 INTRODUCTION 76
4.2 DESIGN OBJECTIVES 78
4.3 SIMULATION ACTIVITY DIAGRAM MODELLING PRIMITIVES 80
4.3.1 Timing of the events in a discrete simulation model 80
4.3.2 SAD Model structure 81
4.3.3 Primary resource element 84
4.3.4 Queue resource element 84
4.3.5 SAD State Elements 85
4.3.6 Auxiliary resource element 88
4.3.7 Branching Elements 90
4.3.8 Link Types 98
4.3.9 SAD Frame Element 100
4.4 DEVELOPING A SIMULATION ACTIVITY DIAGRAM 102
4.4.1 An Activity and an Action list 103
4.4.2 Extending SADs to include systems information data 106
4.4.3 Elaboration of SAD models 111
4.4.4 Hierarchical structuring of SADs 113
4.5 DIFFERENTIATION OF THE SAD TECHNIQUE FROM CURRENTLY AVAILABLE TECHNIQUES 120

4.6 INITIAL VALIDATION OF THE SAD TECHNIQUE 121
4.7 CONCLUSIONS 122

CHAPTER 5: PROCESS MODELLING FOR SIMULATION (PMS) DEVELOPMENT 124

5.1 INTRODUCTION 124
5.2 SOFTWARE DEVELOPMENT PLATFORM 125
5.3 PMS SOFTWARE DESIGN 127
5.3.1 PMS Architecture 127
5.3.2 PMS Specific Code 128
5.3.3 PMS Software Design Overview 131
5.4 PMS SOFTWARE OVERVIEW/SAD MODEL DEVELOPMENT PROCESS 132
5.5 PMS HIERARCHICAL MODELLING 145
5.6 PROPOSED USAGE OF THE SAD TECHNIQUE/PMS TOOL 147
5.7 DISCUSSION 149
5.8 CONCLUSIONS 150

CHAPTER 6: VALIDATION OF THE SAD TECHNIQUE 152

6.1 INTRODUCTION 152
6.2 OVERVIEW OF A PRECISION COMPONENT MANUFACTURING SYSTEM 153
6.2.1 Inspection 157
6.3 MODELLING PRODUCTION CONTROL SYSTEMS 161
6.3.1 SAD Model of a Kanban production control system 162
6.4 MODELLING A SECTION OF A BATCH FLOW-SHOP 168
6.4.1 Work Region two, carburising 169
6.4.2 Modelling the carburising area 169
6.4 MODELLING A PRODUCTION LINE 174
6.4.1 74mm Syndite Line Product Description 175
6.4.2 Surface grinding 180
6.5 CONCLUSIONS 183

CHAPTER 7: CONCLUSIONS 185

7.1 THESIS SUMMARY 185
7.2 REFLECTION 186
7.3 CONCLUSIONS 190
7.4 FUTURE WORK 191

REFERENCES: 193

APPENDIX A: EXAMPLE OF SADS REPRESENTING A PRECISION COMPONENT MANUFACTURING SYSTEM A1

A.1 INTRODUCTION A2
A.2 DELIVERY A4
A.3 DRILLING STATION A8
A.4 MILLING MACHINES A11
A.5 INSPECTION A14
A.6 PACKAGING AREA A18
A.7 WAREHOUSING A21

APPENDIX B: BOART SAD MODEL B1

B 1 CARBURISING AREA B2

APPENDIX C: SAD MODEL OF A PRODUCTION LINE C1

C.1 PRODUCT DESCRIPTION C2

C.2 MATERIALS CONTROL C6
C.3 CENTRELESS GRINDING C8
C.4 FACE GRINDING C11
C.5 SURFACE GRINDING C14
C.6 EDM PLANING C16
C.7 FINISH LAPPING C20
C.8 SANDBLASTING C23
C.9 ASSESSMENT C26
C.10 MATERIALS CONTROL C29

List of Figures

FIGURE 1.1 THE LIFE CYCLE OF A SIMULATION STUDY [8] 3
FIGURE 2.1 PETRI NET [27] 13
FIGURE 2.2 ACTIVITY CYCLE DIAGRAM 15
FIGURE 2.3 DEVS FORMALISM 16
FIGURE 2.4 UML ACTIVITY DIAGRAM [42] 19
FIGURE 2.5 ROLE ACTIVITY DIAGRAM [50] 23
FIGURE 2.6 GRAI MODEL SUB-SYSTEMS 26
FIGURE 2.7 GRAI MODEL 27
FIGURE 2.8 GRAI CONCEPTUAL REFERENCE MODEL 28
FIGURE 2.9 GRAI CONTROL MODEL 29
FIGURE 2.10 GRAI GRID AND GRAI NET 31
FIGURE 2.11 IEM GENERIC CLASS STRUCTURE 32
FIGURE 2.12 IEM MAIN VIEWS 35
FIGURE 2.13 AN IEM ACTION, FUNCTION AND ACTIVITY 36
FIGURE 2.14 GENERIC ACTIVITY 36
FIGURE 2.15 IEM PROCEDURE/PROCESS 37
FIGURE 2.16 IEM MODELLING CONSTRUCTS 38
FIGURE 2.17. EVENT DRIVEN PROCESS CHAIN [64] 39
FIGURE 2.18. EXTENDED EVENT DRIVEN PROCESS CHAIN [66] 41
FIGURE 2.19 OBJECT-ORIENTED EVENT DRIVEN PROCESS CHAINS [64] 42
FIGURE 2.20 IDEF0 MODEL [68] 44
FIGURE 2.21 IDEF0 DECOMPOSITION [68] 46
FIGURE 2.22 IDEF0 EXAMPLE [68] 47
FIGURE 2.23 IDEF3 UNIT OF BEHAVIOUR (UOB) DESCRIPTION 49
FIGURE 2.24 EXAMPLE IDEF3 OBJECT STATE TRANSITION NETWORK DIAGRAM [71] 50
FIGURE 2.25 THE MOOGO USER INTERFACE 52
FIGURE 2.26 STRUCTURE OF THE MOOGO TOOL 53
FIGURE 2.27 MOOGO USER INTERFACE SHOWING GENERATED REPORTS. 54
FIGURE 2.28 ARIS VIEWS 56
FIGURE 2.29 ARIS TOOLSET USER INTERFACE 57
FIGURE 2.30 HOW THE CONTROL VIEW LINKS THE VARIOUS VIEWS. 58
FIGURE 2.31 ROUTING AND MATERIAL FLOW AS EVENT-DRIVEN PROCESS CHAIN. 59
FIGURE 2.32 PROSIM USER INTERFACE SHOWING UOBS 59
FIGURE 2.33 PROSIM SHOWING AN EXPANDED UOB. 60
FIGURE 2.34 UOB SUB-MODEL. 61
FIGURE 2.35 PROSIM USER INTERFACE SHOWING THE OSTN VIEW 61
FIGURE 2.36 OSTN SUB-MODEL 62
FIGURE 3.1 REQUIREMENTS SATISFACTION ATTRIBUTED TO REVIEWED TECHNIQUES 67
FIGURE 3.2 HIGH LEVEL INITIAL SAD DIAGRAM 72
FIGURE 3.3 INITIAL SAD DRAFT OF DELIVERY AREA 73
FIGURE 3.4 HIGH LEVEL SAD DRAFT MODEL SHOWING INFORMATION FLOWS 75
FIGURE 4.1 DIFFICULTIES WITH SIMULATION MODELS AS A COMMUNICATIVE TOOL 77
FIGURE 4.2 PROPOSED USE OF THE SAD TECHNIQUE 78
FIGURE 4.3 THE DIRECTION OF EXECUTION OF EVENTS WITHIN A SAD DIAGRAM 81
FIGURE 4.4 A CHANGE OF STATE OF A SIMPLE DISCRETE EVENT SYSTEM 82
FIGURE 4.5 SAD ACTIONS 83
FIGURE 4.6 QUEUE AND PRIMARY RESOURCE ELEMENTS. 85
FIGURE 4.7 ENTITY STATES 86
FIGURE 4.8 AUXILIARY RESOURCES 89
FIGURE 4.9 SAD BRANCHING ELEMENTS. 92
FIGURE 4.10 AND BRANCHES 93
FIGURE 4.11 OR BRANCHES 93
FIGURE 4.12 USE OF BRANCHING ELEMENTS 95
FIGURE 4.13 ASYNCHRONOUS "AND" BRANCHES 96
FIGURE 4.14 SYNCHRONOUS "AND" BRANCHES 96
FIGURE 4.15 ASYNCHRONOUS "OR" BRANCHES 97
FIGURE 4.16 SYNCHRONOUS "OR" BRANCHES 97
FIGURE 4.17 USE OF DIFFERENT BRANCH TYPES TOGETHER IN THE SAME MODEL 98

FIGURE 4.18 SAD LINK TYPES 98
FIGURE 4.19 ENTITY LINK 99
FIGURE 4.20 INFORMATION LINK 99
FIGURE 4.21 ACTIVITY LINK 100
FIGURE 4.22 FRAME ELEMENTS 102
FIGURE 4.23 A SIMPLE SYSTEM 102
FIGURE 4.24 AN ACTIVITY IN A DISCRETE EVENT SYSTEM. 102
FIGURE 4.25 A SYSTEM TRANSITIONING AT A DECISION POINT, D, AS A RESULT OF AN ACTIVITY, A. 103
FIGURE 4.26 AN ACTIVITY WITH A NUMBER OF ACTIONS. 104
FIGURE 4.27 AN ACTIVITY INCORPORATING RESOURCES. 105
FIGURE 4.28 A SIMPLE SIMULATION ACTIVITY DIAGRAM (SAD). 105
FIGURE 4.29 INFORMATION FLOWS WITHIN A MANUFACTURING SYSTEM. 107
FIGURE 4.30 EXTENDED SAD STRUCTURE 108
FIGURE 4.31 AN EXTENDED SAD. 110
FIGURE 4.32 HIERARCHICAL STRUCTURE OF A MANUFACTURING SYSTEM. 114
FIGURE 4.33 AN EXTENDED SAD INCLUDING FRAMES. 115
FIGURE 4.34 EXTENDED SAD FOR DEPARTMENT A 117
FIGURE 4.35 EXTENDED SAD FOR DEPARTMENT B 119
FIGURE 5.1 DOCUMENTS AND VIEWS IN THE MFC APPLICATION FRAMEWORK 126
FIGURE 5.2 PMS HIGH LEVEL ARCHITECTURE 127
FIGURE 5.3 PARTIAL MFC HIERARCHY CHART – INHERITANCE 128
FIGURE 5.4. PARTIAL MFC HIERARCHY CHART - INSTANTIATION 129
FIGURE 5.5 PMS CLASSES WITH INHERITED MFC CLASSES 130
FIGURE 5.6 PMS MODELLING ENVIRONMENT START SCREEN 133
FIGURE 5.7 PMS OPTION TO CREATE ENTITIES 133
FIGURE 5.8 ENTITIES OR INFORMATION CREATION OPTIONS 134
FIGURE 5.9 ENTITIES WITH VARIOUS STATES CREATED 134
FIGURE 5.10 MODELLING ELEMENTS THAT CAN BE ADDED TO BUILD THE MODEL. 135
FIGURE 5.11 A STANDARD DETAILS DIALOG BOX FOR THE ADDITION OF A MODELLING ELEMENT. 135
FIGURE 5.12 A STANDARD DETAILS DIALOG BOX FOR THE ADDITION OF A BRANCH MODELLING ELEMENT 135
FIGURE 5.13 A PRIMARY RESOURCE ELEMENT FOR A MILLING M/C 136
FIGURE 5.14 A STANDARD DIALOG BOX FOR THE ADDITION OF AN ENTITY OR INFORMATION STATE 136
FIGURE 5.15 ELEMENTS FOR A SIMPLE SAD DIAGRAM 137
FIGURE 5.16 SAD MODEL POPUP MENU 137
FIGURE 5.17 ADDING LINKS AND THE ADD LINK POPUP MENU 138
FIGURE 5.18 EDIT PROPERTIES DIALOG BOX 139
FIGURE 5.19 EDIT DESCRIPTION DIALOG BOX 139
FIGURE 5.20 ATTACH DOCUMENT DIALOG 140
FIGURE 5.21 OPEN DIALOG 140
FIGURE 5.22 ATTACHED DOCUMENT ADDED TO THE ATTACH DIALOG 140
FIGURE 5.23 CREATE/EDIT ATTRIBUTE DIALOG 141
FIGURE 5.24 ATTRIBUTE SELECTION DROPDOWN LIST 141
FIGURE 5.25 ELABORATE FUNCTION 142
FIGURE 5.26 PMS ELABORATION 142
FIGURE 5.27 A FRAME ELEMENT 145
FIGURE 5.28 MIGRATE UPWARDS BUTTON 145
FIGURE 5.29 A HIGH LEVEL INITIAL SAD DIAGRAM CONTAINING FRAME ELEMENTS 146
FIGURE 5.30 SUB-MODEL OF A DISCRETE EVENT SYSTEM CONTAINED WITHIN A FRAME ELEMENT 147
FIGURE 5.31 SAD AND PMS CURRENT SPHERE OF USAGE 148
FIGURE 6.1 SAD MODELLING ELEMENTS 154
FIGURE 6.2 HIGHEST LEVEL OF PRECISION COMPONENT MANUFACTURING SYSTEM. 155
FIGURE 6.3 INSPECTION AREA 158
FIGURE 6.4 TYPES OF KANBAN CARD 161
FIGURE 6.5 KANBAN CONTROL EXAMPLE HIGH LEVEL VIEW 163
FIGURE 6.6 KANBAN CONTROL OF MACHINING AREA 165

FIGURE 6.7 KANBAN CONTROL EXAMPLE ASSEMBLY AREA 167
FIGURE 6.8 FURNACE AREA SAD 172
FIGURE 6.9 74MM HIGH LEVEL SAD DIAGRAM 178
FIGURE 6.10 SURFACE GRINDING 181
FIGURE 7.1 (A) TECHNIQUES REQUIREMENTS SATISFACTION (B) REQUIREMENTS CLAIMS FOR SAD 167
FIGURE A.1 SHOP FLOOR LAYOUT A2
FIGURE A.2 HIGHEST LEVEL OF THE SYSTEM. A3
FIGURE A.3 DELIVERY AREA A6
FIGURE A.4 DRILLING AREA A9
FIGURE A.5 MILLING AREA A12
FIGURE A.6 INSPECTION AREA A15
FIGURE A.7 PACKAGING AREA A19
FIGURE A.8 WAREHOUSING AREA A22
FIGURE B.1 FURNACE AREA SAD B5
FIGURE C.1 74MM SYNDITE TOP LEVEL SAD C4
FIGURE C.2 MATERIALS CONTROL SAD C7
FIGURE C.3 CENTRELESS GRINDING C9
FIGURE C.4 FACE GRINDING C12
FIGURE C.5 SURFACE GRINDING C15
FIGURE C.6 EDM PLANING C19
FIGURE C.7 FINISH LAPPING C21
FIGURE C.8 SANDBLASTING C24
FIGURE C.9 ASSESSMENT C27
FIGURE C.10 MATERIALS 2 ..... C30

List of Tables

TABLE 6.1 ELABORATION DESCRIPTION THE HIGHEST LEVEL OF THE PRECISION COMPONENT MANUFACTURING SAD DIAGRAM 156
TABLE 6.2 ELABORATION LANGUAGE DESCRIPTION FOR THE INSPECTION AREA 160
TABLE 6.3 KANBAN HIGH LEVEL SAD ELABORATION 164
TABLE 6.4 KANBAN MACHINING AREA ELABORATION 166
TABLE 6.5 KANBAN ASSEMBLY AREA ELABORATION 168
TABLE 6.6 FURNACE OPERATION PRIORITIES 171
TABLE 6.7 FURNACE AREA ELABORATION 174
TABLE 6.8 CARBURISING FURNACE CYCLE TIMES 174
TABLE 6.9 74MM SYNDITE LINE PRODUCTS AND ITEM NUMBERS. 176
TABLE 6.10 74MM SYNDITE PROCESS 177
TABLE 6.11 74MM HIGH LEVEL SAD ELABORATION 179
TABLE 6.12 SURFACE GRINDING ELABORATION. 182
TABLE A.1 ELABORATION DESCRIPTION THE HIGHEST LEVEL SAD DIAGRAM A4
TABLE A.2 ELABORATION DESCRIPTION FOR THE DELIVERY AREA A8
TABLE A.3. ELABORATION DESCRIPTION OF THE DRILL AREA A11
TABLE A.4. ELABORATION LANGUAGE DESCRIPTION OF THE MILLING AREA A14
TABLE A.5. ELABORATION LANGUAGE DESCRIPTION FOR THE INSPECTION AREA A17
TABLE A.6 ELABORATION LANGUAGE FOR THE PACKAGING AREA A20
TABLE A.7 ELABORATION LANGUAGE DESCRIPTION FOR THE WAREHOUSING AREA A23
TABLE B.1 FURNACE OPERATION PRIORITIES B4
TABLE B.2 FURNACE AREA ELABORATION B7
TABLE B.3 CARBURISING FURNACE CYCLE TIMES. B7
TABLE B.4 MAXIMUM CARBURISING JIG UTILISATION CHART. B8
TABLE B.5 HEXAGONAL ROD WEIGHTS IN LBS (ROUNDED UP). B9
TABLE B.6 ROUND ROD WEIGHTS IN LBS (ROUNDED UP). B9
TABLE B.7 MANNING REQUIREMENTS FOR LOADING THE CARBURISING FURNACE JIG WITH HEXAGONAL RODS. B9
TABLE B.8 MANNING REQUIREMENTS FOR LOADING THE CARBURISING FURNACE JIG WITH ROUND RODS. B9
TABLE B.9 MANNING REQUIREMENTS FOR UNLOADING THE CARBURISING FURNACE JIG WITH HEXAGONAL RODS. B10
TABLE B.10 MANNING REQUIREMENTS FOR UNLOADING THE CARBURISING FURNACE JIG WITH ROUND RODS. B10
TABLE B.11 TRAY TYPES AND MAXIMUM CAPACITY. B10
TABLE C.1 74MM SYNDITE LINE PRODUCTS AND ITEM NUMBERS. C2
TABLE C.2 74MM SYNDITE PROCESS C3
TABLE C.3 74MM SYNDITE TOP LEVEL ELABORATION C5
TABLE C.4 MATERIALS CONTROL ELABORATION C8
TABLE C.5 CENTRELESS GRINDING ELABORATION. C10
TABLE C.6 FACE GRINDING ELABORATION. C13
TABLE C.7 SURFACE GRINDING ELABORATION. C16
TABLE C.8 EDM PLANING ELABORATION C19
TABLE C.9 FINISH LAPPING ELABORATION. C23
TABLE C.10 SANDBLASTING ELABORATION. C25
TABLE C.11 ASSESSMENT ELABORATION. C29
TABLE C.12 USABLE DISC AREA FOR CUTTING DISCS C29
TABLE C.13 MATERIALS 2 ELABORATION. C31

Chapter 1: Introduction

1.1 Introduction

Most Discrete Event Systems (DES) such as manufacturing systems or business processes are complex and difficult to understand and operate efficiently. One of the most commonly used tools for the analysis of such systems is simulation, [1], [2]. Simulation in theory has great potential to assist in the understanding and efficient operation of these systems, however it has not achieved the penetration that was predicted in the 1980's. Many reasons have been put forward for this such as, poor salesmanship, poor education and time commitments within an organisation [3]. However another reason may be the heavy burden placed on the model developer.

Prior to coding a simulation model a model developer has to gather detailed information on a system under investigation. This information then has to be communicated to system personnel to ensure correct assumptions are being made regarding the system and that the information being used is accurate. Often simulation modelling can then become a very heavy programming-oriented task with the gathered information regarding the inner workings of a system being lost in the detailed programming code and only visible to those who are intimately involved in the programming task. While it is important to have this information in a format that can be communicated and reasoned over during the initial phases of a simulation project, this information may also contain valuable insights into the operation of a system that may otherwise be lost in simulation code. For instance a simulation model will contain detailed information as regards part routings, operations, resource configurations, processing times and

so on. A lot of this information concerning the operation of a system is lost in the detailed simulation code.

1.2 The Process of Simulation

In conducting a simulation project it is recommended that a structured systematic approach be carefully planned and rigidly adhered to. The “40-20-40” rule is widely quoted in simulation texts. The rule states that, in developing a model, an analyst’s time should be divided as follows [4]:

- 40% to requirements gathering such as problem definition, project planning, system definition, conceptual model formulation, preliminary experiment design and input data preparation;
- 20% to model translation;
- 40% to experimentation such as model validation and verification, final experimental design, experimentation, analysis, interpretation, implementation and documentation.

It is rare for these phases to be totally independent. For example, in the requirements gathering phase one would consider programming implications. The model developer would also make an effort to program the simulation model in such a way as to allow for easy and accurate experimentation. Figure 1.1 shows in more detail the tasks involved in simulation modelling [8]. Many of these tasks take place prior to the coding phase of a project and may be repeated at different stages of the project depending on model revisions. Many developments have taken place around supporting the “model coding or translation task” of a simulation model with highly developed modelling tools such as EM Plant [5], Arena [6] and Taylor ED [7]. But there have been very few techniques or tools developed to explicitly support the tasks prior to coding a simulation model.

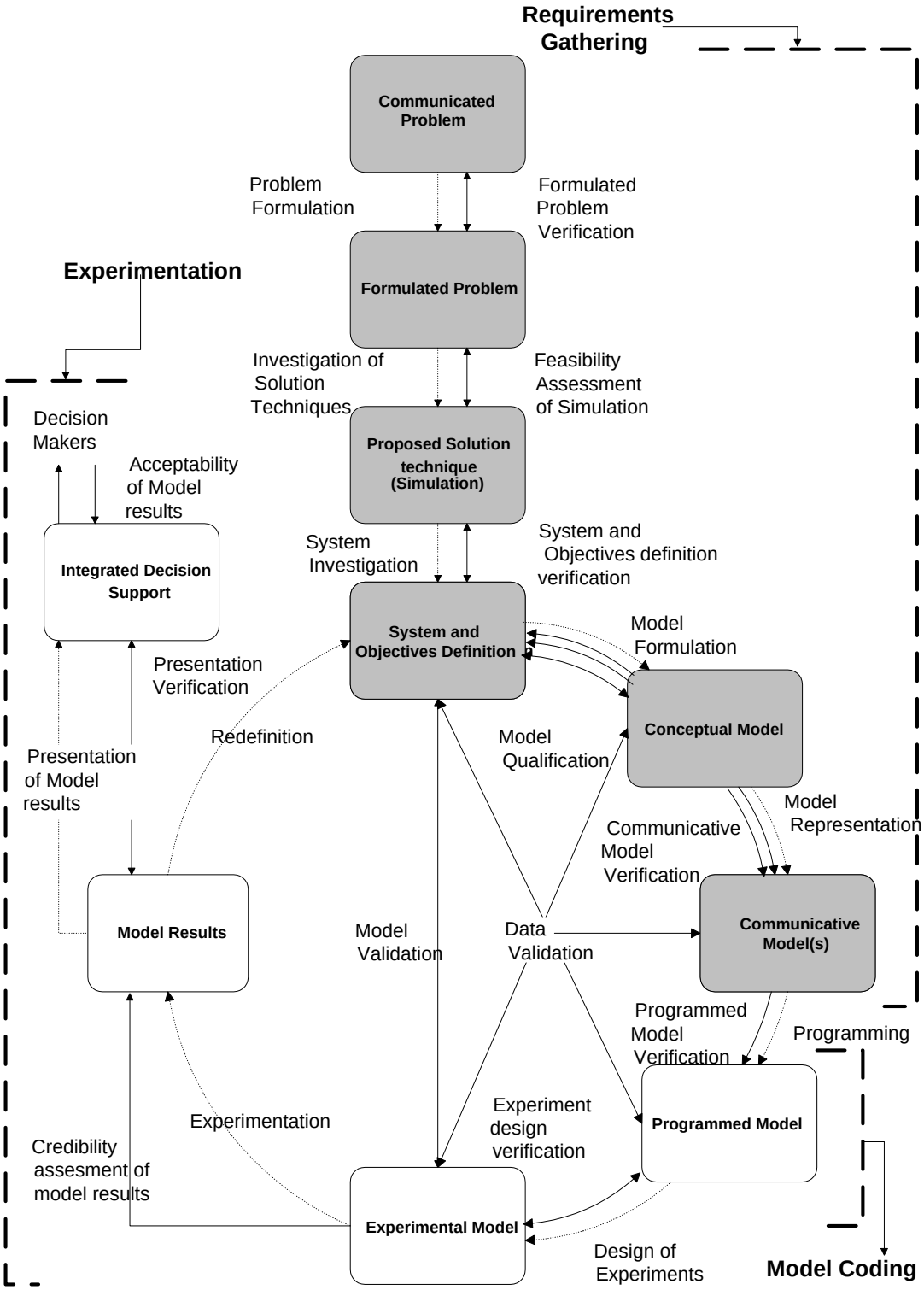

Figure 1.1 The Life cycle of a simulation Study [8]

1.2.1 Requirements Gathering

Organisations are complex systems and collecting data relating to operations and processes from such systems can be very difficult and time-consuming. On the other hand this difficulty may be overcome by the use of systematic methods and tools as suggested by The National Research Council (NRC) [9]:

> ***"… tools for describing process are critical for the design of individual products, the design and operation of factories, and the development of modelling and simulation technology. Formal descriptions are necessary if processes are to be presented in sufficient detail and with enough specificity to be adequately complete and unambiguous. Such formalisms would allow designers to describe factory processes (involving both machines and people), design activities, decision processes, among others."***

Process Modelling methodologies have been developed to collect and evaluate the knowledge on processes in production, material flow, business, production development, logistics and production procedures. They are used to gain an understanding of the static and dynamic behaviours of systems. The main objectives of Process modelling are to [10]:

- Facilitate human understanding and communication. This requires a group to be able to share a common representational format;
- Support process improvement. This requires a base for defining and analysing processes;
- Automate process guidance. This requires automated tools for manipulating process descriptions;
- Automated execution support. This requires a computational base for controlling behaviour within an automated environment.

During the initial stages of developing a simulation model it is necessary to have a means of modelling a process, which allows personnel involved in the operation of the system to have access to an effective method of communicating

system knowledge and information to the model developer [8]. For a modelling technique to be accepted by a project team the notation used should be easily understandable and promote communication [11]. Such a method of communication should increase the understanding of a wide variety of people with varying levels of skills and expertise, from the managing director to the shop-floor worker in order to aid in the process of developing a valid conceptual model. For a technique to be capable of achieving such a goal the modelling elements used should be capable of capturing and representing detailed information while also not being abstract to the point of being obscure to system personnel.

The problem definition and conceptual model formulation process is often a time-consuming one, as is the process of collecting detailed information on the operation of a system [8]. The development of a technique, which allows for the effective communication of system operational issues at an early stage of development would be of benefit, as it would give an early indication of information that is relevant to the model being developed.

Hollocks [12] recognised that such pre modelling and post experimentation phases of a simulation project together represent as much or more effort than the modelling section of such projects and that software support for these phases of the wider simulation process would be valuable. Some of the particular areas of potential support highlighted by Hollocks included documentation, communication, and administration. Such areas are also discussed by Sargent [13] in terms of model documentation, and model validity. This lack of support for documentation in preference for rapid model production was further highlighted by Cornwell et al. [14], who claimed that only 2% of software systems such as modelling and simulation are usable upon delivery. This they ascribe point to the lack of development, documentation, maintenance and management practices for software development, which if in place can result in systems that can provide greater returns on investment and that can be used and evaluated for suitability without the need for costly rework. The difficulties of establishing model credibility due to the lack of good development practices and documentation are also

discussed. Nethe and Stahlmann [15] discuss the practice of developing high level process models prior to the development of a simulation model. Such a method they feel would greatly aid in the collection of relevant information on system operations (i.e. data collection) and therefore reduce the effort and time consumed to develop a simulation model. Such a process modelling method for simulation could be used as a knowledge acquisition method for simulation studies.

The above work reinforces the perception that this area of pre-coding support is ill-addressed and worthy of attention. Given the above evidence of the need for the support of the wider simulation project phases and the current lack of techniques or tools with capabilities to fulfil this gap this present research was undertaken.

In relation to the wider software process modelling Acuna et al [16] in a review of software process modelling listed the basic process modelling elements which included elements such as user viewpoint, versioning, transactions, agents, actors or roles, activities, products. Elements such as these may be used to form the basis of the process modelling technique developed. However the technique should be developed with a specific emphasis on usefulness in the pre-coding phases of a simulation project.

1.3 Research Objectives

The objective of the work reported in this thesis was to develop a process modelling technique to aid a simulation model developer during the requirements gathering phase of a discrete-event simulation project.

The more detailed goals emanating from the primary objective above are the development of a technique that:

- Could capture a detailed description of the various aspects of a DES for the purposes of a simulation project, those being;

- The flow of work, or change of state of a discrete event system;
- The flow of information associated with the control of a discrete event system;
- The activities that are associated with the execution of the flow of work and information within a discrete event system;
- The resources necessary and their usage in the execution of the activities associated with both work and information within a discrete event system;

- Has a low modelling burden and therefore can be used by non-specialists; aspects that may facilitate this include:
  - The modelling of a discrete event system from the perspective of the user and their interactions with the system in the execution of activities within the system.
  - The separation between the process modelling tool and the simulation engine to allow for the capture, representation and communication of detailed interactions at a high level during the requirements gathering phase, as opposed to purely at the low level code stage of a project.
- Presents modelling information in terms of concepts that are meaningful to system personnel such as resources and activities, as opposed to abstract terms, to facilitate understanding and communication.
- Has a good visualisation capability to facilitate communication between a model developer and system personnel. The following initiatives may be of benefit.
  - The access to a means of elaborating graphical models to facilitate the communication of detailed information associated with such graphical representations

- To be capable of hierarchically structuring a model to facilitate the decomposition of complex situations into related sub models;
- To graphically represent the various tasks within a system and present these tasks in a time phased sequence of execution within a system.

The points listed above will hereafter be referred to as “the requirements”.

In summary, the above requirements were developed to allow for the development of a process modelling technique that was capable of facilitating communication and understanding between a simulation model developer and system personnel, while simultaneously being capable of aiding in the requirements gathering and conceptual modelling phases of such a project.

The shaded tasks shown in Figure 1.1, page 3 depict the application area of the proposed modelling technique within the overall development of a simulation model. Within these tasks the pre-coding tasks, including the development of the conceptual process model, are developed by both a simulation model developer and system personnel. This information is then communicated to personnel drawn from various aspects of the system being modelled. For example in a logistics problem, one would have representatives of suppliers, buyers, traffic planning, warehousing, assembly, transport agencies, and so on. In the modelling of a manufacturing plant, one might have representatives of planning, scheduling, maintenance, production, product engineering, finance, marketing, and so on - people with a wide degree of differences in perception and goals. This communication process will be aided by the use of high level modelling semantics that are comprehendible by both a simulation model developer and system personnel not skilled in simulation modelling.

Such an approach as proposed would facilitate and encourage a systematic teamwork approach to the development of conceptual process models and in turn simulation models. This would be achieved by aiding understanding and allowing for consensus agreement over important data and system logic at an early stage of the modelling process. Also, by reducing the initial development

iterations, changes to the actual simulation model and also reducing the validation time, it was hoped to reduce simulation modelling lead times, and achieve better-balanced models in terms of functionality.

The graphical nature of the technique therefore gives the model developer and system personnel representations, with which to reason over the assumptions, which are undertaken in the development of the simulation model prior to coding. A technique satisfying the above requirements would promote knowledge reuse, in as much as the detailed information regarding the inner workings of a simulation model module would be available in a format that could easily be accessed and understood, and thereby more effectively documented for future use.

The document is divided into the following remaining chapters:

- **Chapter 2**. Process Modelling: Various established methods that can be employed for the purposes of modelling and representing processes are reviewed.
- **Chapter 3.** SAD Development Process: The evolution of the SAD concept is outlined.
- **Chapter 4.** Simulation Activity Diagrams (SADs): the elements of the proposed modelling technique are presented.
- **Chapter 5:** Process Modelling for Simulation (PMS) Development: The software tool that was developed to implement the SADs, entitled “the Process Modelling for Simulation” (PMS) tool is introduced.
- **Chapter 6**: Validation of the SAD Technique: The PMS software tool is used to develop and implement a number of SAD model instances, based on both conceptual and real discrete-event system cases, and the experiences reflected upon.
- **Chapter 7**: Conclusions: The overall processes in the development of the SAD technique and PMS tool are discussed and reflected upon; summary conclusions and possibilities for further work are outlined.

Chapter 2: Process Modelling

2.1 Introduction

This thesis proposes the development of a dedicated process modelling technique to aid in the requirements gathering phase of a discrete event simulation project. Discrete Event Systems (DES) encompass a wide variety of physical systems, including manufacturing systems, service systems, traffic systems and communication systems. A DES may be thought of as a dynamic system that is equipped with a state space and a state-transition structure. In particular, a DES is discrete in time and in state space, and is event-driven, i.e., a state change is precipitated by the occurrence of an event. For example at a certain level of abstraction in a manufacturing system, a machine may be described as a DES with states “idle”, “working”, and “breakdown”, and the associated transitions “start”, “finish”, “machine-failed”, and “stop”.

This chapter reviews a number of different modelling techniques and tools that are used to model discrete event systems. It would not be practical to undertake a review of all such techniques and tools, Kettinger et al [17] listed over one hundred different methods available for the purposes of process modelling in a survey that was not exhaustive. The techniques and tools reviewed in this chapter are those that it is felt are relevant to the modelling of complex discrete event systems. The rationale behind this review is to ascertain the ability of these techniques and tools to capture and communicate detailed information and knowledge such as that contained in the detailed code of a simulation model. This capture and communication of such detailed information should be visual in nature, utilising high level semantics, to aid both the model developer and associated system personnel in gaining a common understanding of a system

under investigation. As outlined in chapter one such information is often lost in the detail of simulation code and could be of great benefit to others interacting with the system apart from the simulation model developer.

The chapter is divided into two main sections:

1. ***process modelling techniques***

   This section reviews a number of process modelling techniques that, while not specifically designed for the purposes of supporting simulation, can be used in this way.

2. ***process modelling tools***

   Many of the techniques that are used to model discrete processes are implemented or supported by means of software tools. This section provides a review of a number of these process modelling software tools.

The chapter then concludes by providing the reader with the overall conclusions derived from this review.

2.2 Process Modelling Techniques

Discrete event systems are complex and the process of requirements gathering or conceptual modelling for such systems can be very difficult and time-consuming. This difficulty arises from the necessity for a process model developer to gain a thorough understanding of the detailed operations of a discrete system to allow for the formulation of an accurate process model. This process of conceptual modelling is not unimportant within the overall structure of a simulation project [18]. It has been argued that such conceptual process models may even lead to the discovery of a solution to a problem without the necessity of simulating the process [18]. Therefore, the process of developing an accurate process model of a discrete system prior to the development of a simulation model is an extremely important one. However there is a severe lack of publications on the overall subject of conceptual modelling [18]. The following section introduces a number of existing Process Modelling techniques that have been developed to support the modelling of various types and aspects of

systems and can be of use in the development of conceptual process models of a discrete event system. These process modelling techniques are reviewed with a view to assessing their ability to aid the capture, visualisation and communication of detailed discrete event system logic or simulation logic in a high level and user friendly manner.

2.2.1 Petri- Nets

A Petri net is a mathematical formalism that finds its basis in a few simple objects, relations, and rules, but is capable of representing very complex systems [19]. Standard Petri-nets contain the following components: transitions (represented by bars), places (circles), directed arcs and tokens. Arcs join transitions and places, while tokens are dynamic elements moving from place to place. Places and transitions alternate in the net, so that each transition has its preceding (input) and succeeding (output) places [20].

There are various kinds of Petri nets used in simulation and modelling [21], these Petri nets have been used by many researchers in the development of simulation models for a variety of manufacturing systems [22, 23]. Petri nets have also been used to model both knowledge based [24] and management systems [25]. Petri-nets have also been used to simulate discrete event systems, [26]. In these Petri-nets, time delays are added using temporal events shown with a star symbol, which represents the condition for a transition firing. These time delays give individuality to tokens, which can therefore be created or destroyed, split or merged. Transitions correspond to events in simulation, places to activities or states, and tokens to dynamic entities [27]. The example given in Figure 2.1 is of a customer process and server cycle. The start of service transition (event) can be fired only when there is a customer in a “wait” place (state) and a server in the “idle” place. Customer arrival and departure transitions (events) require the passage of time, shown by the temporal events (star symbols).

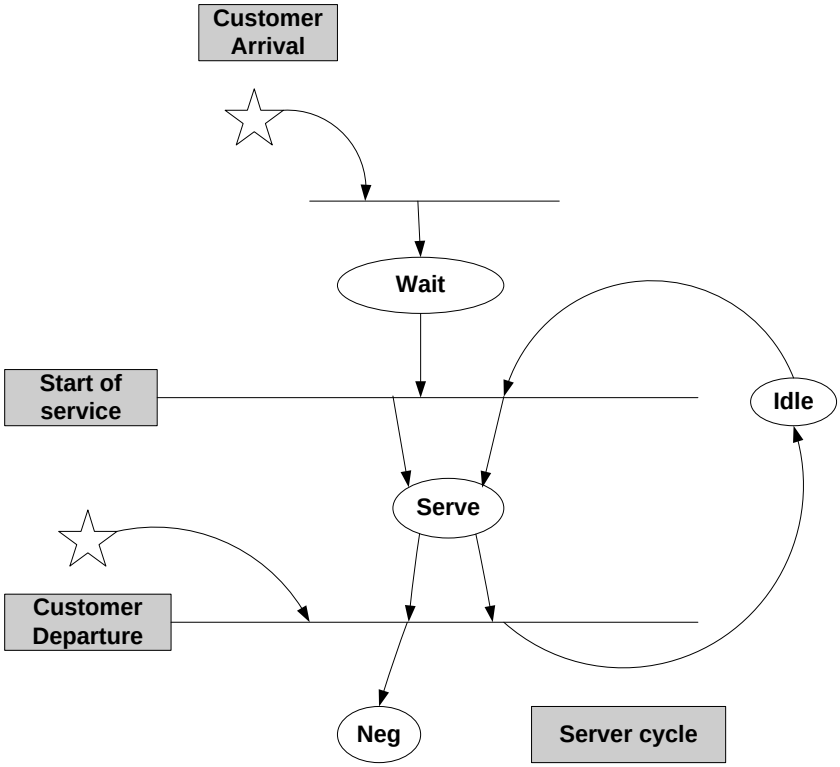

Figure 2.1 Petri Net [27]

There are various different kinds of Petri nets such as stochastic, coloured, hierarchical coloured and object oriented Petri nets. These different types of Petri net vary a lot in their expressive power, legibility of models and analytical capabilities [21]. Such Petri nets are capable of very accurately modelling and representing a real system. The drawback to such a modelling technique is that it tends to be highly abstract and difficult for a non-expert to reason with the logic contained within a model. Therefore, while Petri nets can accurately represent a complex discrete event system the technique has difficulty communicating the detailed information in a manner that could allow both a model developer and system users to use it as a communication medium to reach a common understanding in regard to system operational issues.

2.2.2 Activity Cycle Diagrams

The Activity Cycle Diagram (ACD) is a technique for representing the interaction of entities within a system and is based on stochastic gearwheels as presented

by Tocher [28]. In an ACD entities cycle through alternating states of activity and waiting [29].

ACDs only use two node symbols corresponding to an entity's active and idle states. Lines of different colours, to represent different entities, are then drawn between the nodes and map out the life cycle of the entities [30]. In this way activity cycle diagrams are nothing more than an alternating sequence of queues and activities, starting and ending with a queue. If the same queue is used for the start and end of the cycle, a closed cycle results. ACDs therefore consist of a number of entity life cycles. In each life cycle, the entity cycles through alternating active and passive states, *i.e.* activities and queues. Activities are interaction points between different life cycles in an activity cycle diagram, where different types of entities co-operate, while queues represent states of entities waiting for some conditions to be fulfilled in order to move to an active state. An ACD therefore, graphically shows both the potential life cycle of each class of entity within a system and the entities interaction with the system. There are some basic rules for the constructing of ACDs [29]:

1. A queue must contain only one type of entity;
2. An activity always follows a queue and vice-versa, when there are no reasons for queuing before an activity, dummy queues may be incorporated into the model;
3. All life cycles of each entity type should be closed.

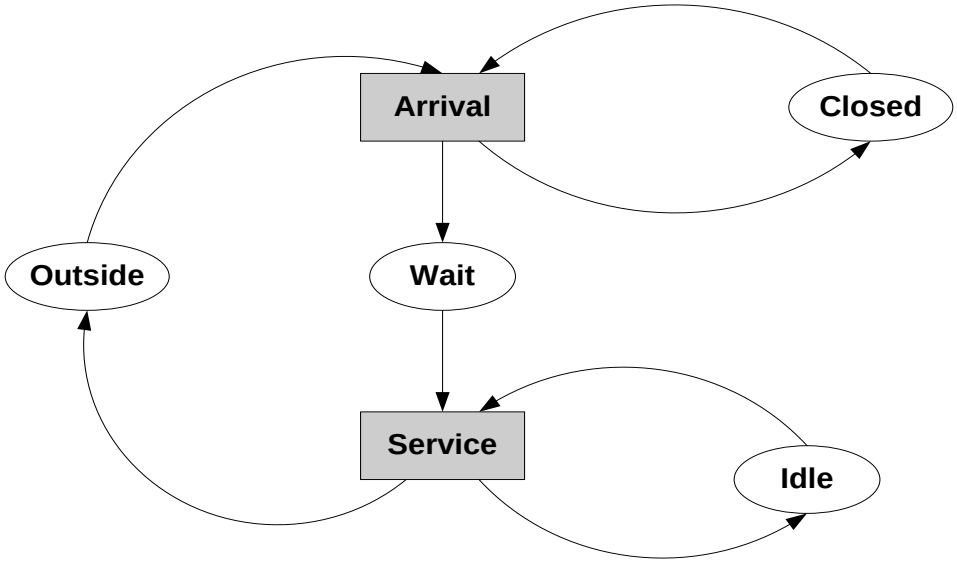

Figure 2.2 Activity Cycle Diagram

Figure 2.2 shows an activity cycle diagram for a customer arrival system. The example shows three life cycles, “Arrival” and “Service” activities are the interaction points between different entities. For example, the “Service” activity starts whenever both the customer queue “wait” and the server queue “idle” are not empty. ACDs have been used in the development of STROBOSCOPE, a simulation language that can be used to express the logic of complex simulation models for construction [31]. ACDs have also been proposed as a method for simplifying the modelling process of construction simulation [32]. While ACDs are capable of being used to model information, a number of weaknesses have been noted, including difficulties in capturing complex logic along with models becoming cumbersome when modelling a complex system [29].

2.2.3 Discrete Event System Specification (DEVS)

Zeigler described the DEVS formalism [33] as a means of specifying a mathematical object called a system, which has a time base, inputs, states, and outputs, and functions for determining next states and outputs from current states and inputs. He proposes that discrete event systems represent certain collections of such parameters just as continuous systems do, along with proposing that

there should be a separation between a model that describes a system and the mechanism used to simulate that system Figure 2.3

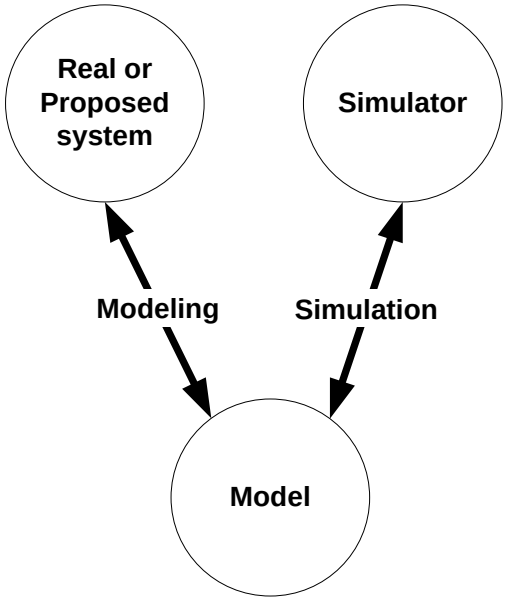

Figure 2.3 DEVS Formalism

Zeigler, [33], [34] proposed the discrete event system specification (DEVS) to also overcome the problem of separation between a model of a system and the means of simulating that system. The system uses a mathematical formalism to represent discrete event systems. A model M is represented by (X, S, Y, δ_{int}, δ_{ext}, λ, t_u). A basic model contains the following information:

- The set of input ports through which external events are received;
- The set of output ports through which external events are sent;
- The set of state variables and parameters: two state variables are usually present- phase and sigma (in the absence of external events the system stays in the current phase for the time given by sigma);
- The time advance function which controls the timing of internal transitions- when the sigma state variable is present, this function just returns the value of sigma;

- The internal transition function which specifies to which next state the system will transit after the time given by the time advance function elapses;
- The external transition function which specifies how the system changes state when an input is received - the effect is to place the system in a new phase and sigma thus scheduling it for a next internal transition; the next state is computed on the basis of the present state, the input port and value of the external event, and the time that has elapsed in the current state;
- The output function which generates an external output just before an internal transition takes place.

According to Zeigler the DEVS formalism provides not only a method for constructing simulation models but also a formal representation of discrete event systems, facilitating mathematical manipulation just as differential equations serve this role for continuous systems. Within this formalism, complex systems may be modelled, designed, analysed and simulated. This formalism has been used to support the design and simulation of computer architectures, communications networks, and manufacturing systems [35], [36]. It provides a formal representation of discrete event systems capable of mathematical manipulation just as differential equations serve this role for continuous systems, however the proposed mathematical representation is difficult to reason over without a detailed knowledge of the formalism. So, while the DEVS formalism is capable of accurately modelling a complex discrete event system, the technique does not lend itself to communicating such complex information in a manner that facilitates its use as a means for model developers and non-specialists to reason over system issues.

2.2.4 Unified Modelling Language

The Unified Modelling Language (UML) is designed to aid software developers in specifying, visualising, constructing and documenting a software system,

business system or other non-software system and represents a collection of the best engineering practices that have proven successful in modelling large and complex systems [37]. UML has also been proposed as a means to specify simulation models [38]. UML provides the model developer with a collection of different graphical diagrams, these being:

- Use class diagrams;
- Class diagrams;
- Behaviour diagrams;
- State chart diagrams;
- Activity diagrams;
- Interaction diagrams;
- Sequence diagrams;
- Collaboration diagrams;
- Implementation diagrams;
- Component diagrams;
- Deployment diagrams;

These diagrams provide the model developer with multiple views of the system being developed. The underlying model of the UML then integrates these views into one consistent model that can be documented, built and analysed [37]. Of these diagrams, the UML activity diagram is the only notation proposed for modelling Business Processes and workflows [37], [39]. However, statecharts [40] were originally developed for the purposes of aiding in the building of airplane simulators, and have been adopted by UML, therefore statecharts will also be introduced here.

2.2.4.1 UML Activity Diagrams

A UML activity diagram represents the execution of a process as a sequence of steps grouped sequentially as parallel control flow branches. An activity diagram consists of a series of activities represented by rounded rectangles, decision points represented by a diamond, synchronisation bars represented by bars, and transitions represented by lines. These diagrams may also be split into swim lanes to show the various responsibilities within an organisation [41].

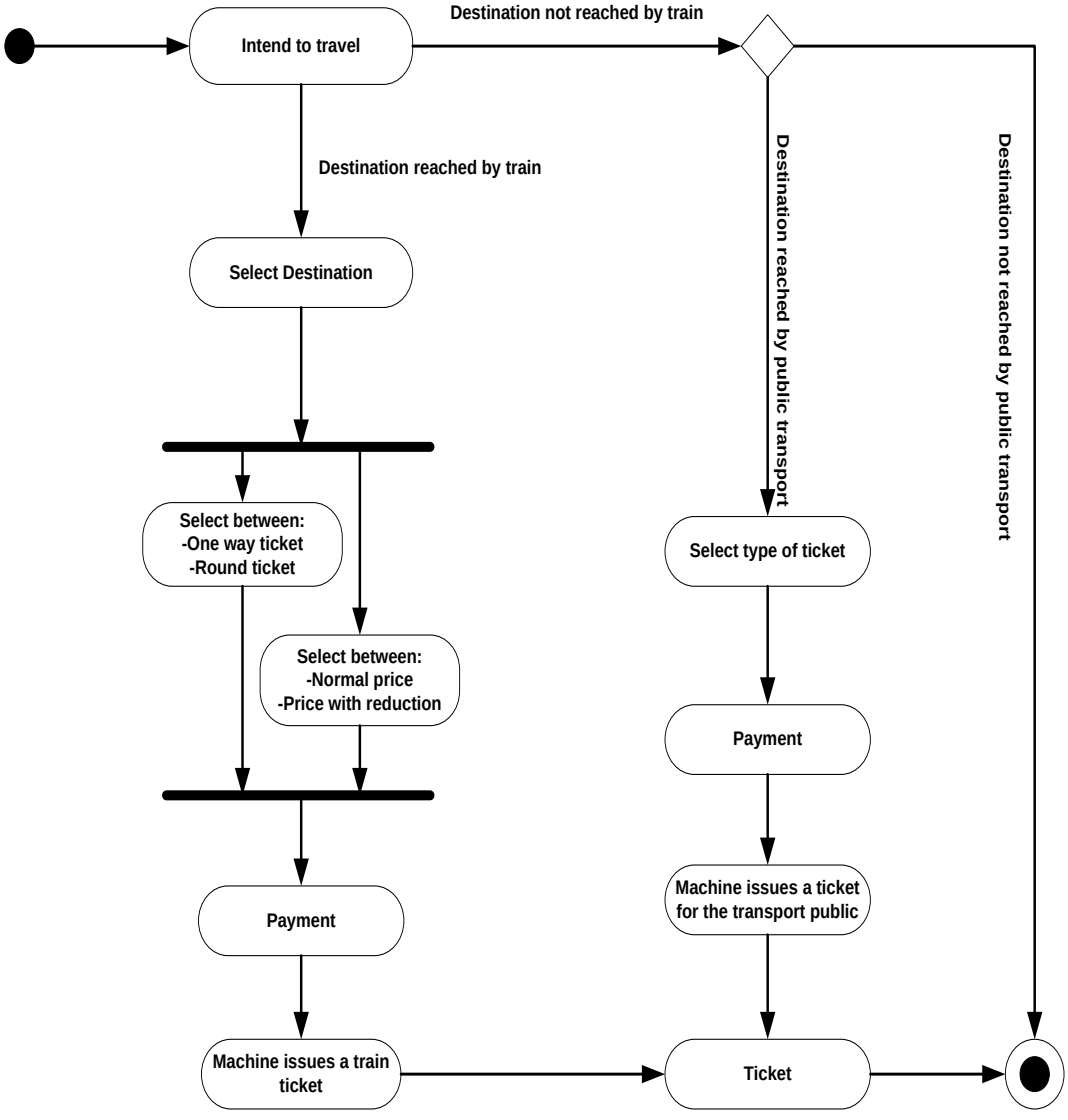

Figure 2.4 UML Activity Diagram [42]

Some of the advantages of activity diagrams are the following [42]:

- Activity diagrams are process-oriented and therefore allow a model developer to view in a comprehensive way all the sub-processes of a process under examination;
- Activity diagrams are capable of handling parallel processes, which is an advantage over those techniques that are limited to sequential processes;
- Activity diagrams can also model dynamic aspects of a system.

These elements give the user a notation, which can be used to model both data and workflow [37]. Activity diagrams have been used in this regard to model or assist in the modelling and simulation of business systems [42]. Figure 2.4 shows the activity diagram for a ticket-selling machine as presented by Barjis and Shishkov [42]. UML activity diagrams have been proposed as a pre-simulation technique [42] and have also been used as part of the FUJABA environment which has been used to test and simulate production control systems [43]. While UML activity diagrams are capable of representing workflow and dataflow within a discrete process they do not visually account for detailed interactions or complex usage of resources such as can take place within a detailed simulation model. Therefore UML activity diagrams may be used to support the requirements gathering or conceptual modelling phase of a simulation project. However, a technique that can visually represent the interactions between resources, system activities and the flow of work would, it is felt, be more capable of communicating detailed simulation logic to a non-simulation expert.

2.2.4.2 UML Statecharts

UML statecharts are based on the notation introduced by Harel [40]. A statechart diagram is made up of a number of basic elements, states and transitions. These statechart diagrams are used to show the flow of control or sequences of states that a system can proceed through as a result of discrete events [44]. A UML statechart is shown in Figure 2.5, in this Figure a number of the statechart elements are shown.

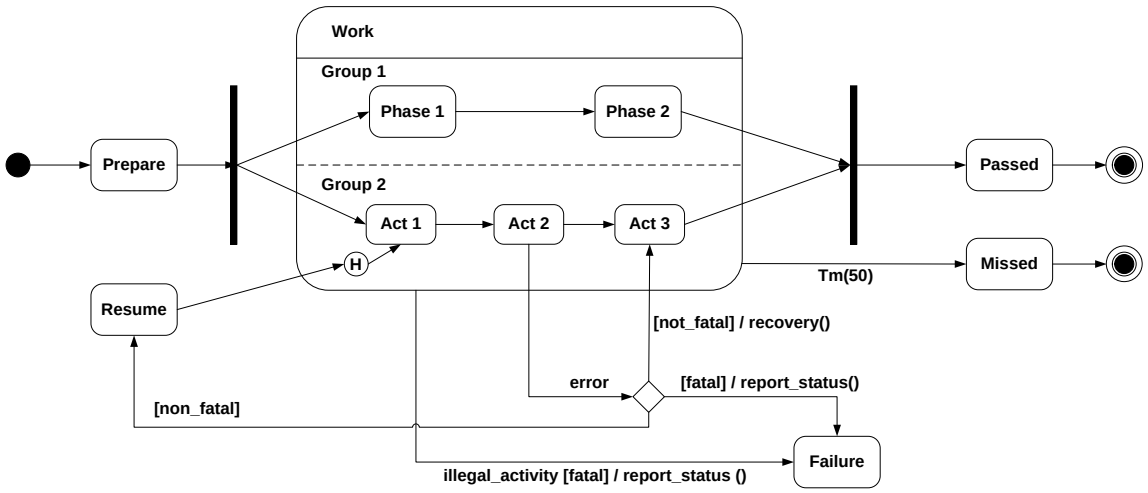

Figure 2.5 UML Statechart [45]

A start state, represented by a black circle shown at the extreme left hand side of the Figure 2.5 is the beginning point of a statechart. Basic states such as the “prepare” state in Figure 2.5 are represented by a rectangle with rounded corners. A state can also be hierarchically decomposed into one or more substates as shown in the “work” state in Figure 2.5. Such decompositions can be either concurrent as with “Group 1” and “Group 2” in Figure 2.5, known as regions (each of which contain their own substates), or mutually exclusive.

Transitions can consist of simple transitions which indicate that the system changes state when a specified event occurs. Compound transitions are also used, and symbolise the splitting or joining of singular threads into multiple threads. Branch segments can also be used to show a point of divergence or convergence of multiple threads as the result certain conditions. Such instances are shown in Figure 2.5 with a simple transition joining the start state with the simple “prepare” state. Also shown are a split and join represented by black bars. Finally a branch segment is shown as a diamond.

History states, similar to that represented by a circled “H” in Figure 2.5 allow a transition to link to the last active substate in that composite state in which it resides. Finally an end state is shown on the far right hand side of Figure 2.5 with a black circle encircled by a circle.

Such statecharts are used in the specification of dynamic systems and provide a means of mapping the various states through which a discrete system can transition and have been used in system simulation [38], [46]. However the statechart diagram does not allow for the capture or modelling of either resource interactions or the activities that cause the change of states within a discrete system. Therefore statecharts do not fully lend themselves to the visual representation of all detailed interactions that may occur within a complex discrete event system and as a result do not have the ability to communicate all such interactions in a visual manner.

2.2.5 Role Activity Diagrams

The technique of Role Activity Diagrams (RADs) as introduced by Ould [47], attempts to model a process in terms of the roles present within the process, their component activities and their interactions, together with external events and the logic that determines what activities are carried out when and by whom. Such an approach of graphically modelling the human interaction with a system benefits the promotion of communication and understanding by means of explicitly representing a person's role within a system. Although RADs have been used in software engineering they are not primarily directed at modelling the information flows within an organisation, a feature that distinguishes them from many other notations in the field [48]. As a result, RADs can and have been used to express the organisation of design activities, communication between various groups involved, and the links between these and the evolving project [48]. RADs have also been proposed as an aid to modelling of a safety process for the purposes of building a safety case for new systems [49]. The notation presented here is based on Ould's notation [47]. A RAD, Figure 2.5, comprises of one or more symbols. The following subsections briefly discuss these symbols as used in this RAD notation.

2.2.5.1 Roles and Activities

A role is depicted by a rounded rectangle surrounding activities, such a role groups together a series of activities that are carried out by an actor or agent, i.e. the unit of responsibility for that actor or agent. The aforementioned activities are used to represent the items of work that people carry out within a role and are represented by boxes within a role. While roles are independent of each other they can communicate through interaction, which acts as a synchronisation mechanism between roles that are acting in parallel [50].

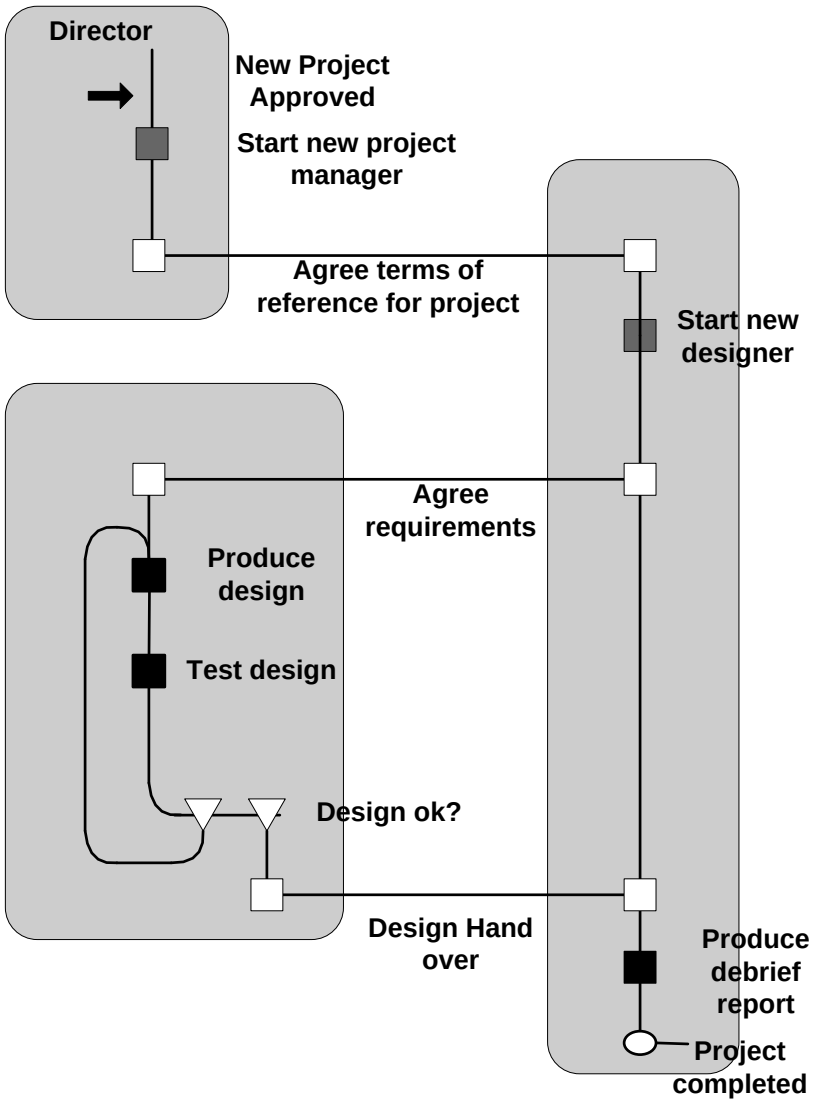

Figure 2.5 Role Activity Diagram [50]

2.2.5.2 Control

Within every role there is a thread of control depicted by a vertical line, which runs from top to bottom within the role with activities being ordered in this way. The thread of control within a role also allows for the following [50]:

- **Interactions:** The point at which a role interacts with another role, represented by a horizontal line or thread linking two activities in corresponding roles;
- **Choices:** Conditional elements within a role, which allow for a choice of activities to be made, conditional elements are represented by inverted triangles;
- **Path refinement:** Used to represent divergent paths or sub-threads within a main thread of control. Path refinements are represented by linked triangles within a RAD diagram;
- **State:** Used to identify particular states within a diagram, denoted by a freeform loop;
- **Iteration:** Used to represent the return to a previous state within a role, such iterations can be represented state markers or an arrow linking two states;
- **Wait:** A wait is used to represent the need for an external input prior to the continuity of the RAD, such waits are graphically represented by an arrow entering the thread of control from the left;
- **Start another role:** One role can start from another role, represented by a crossed box within a RAD.

While RADs lack the ability to model the change of state of a discrete event system they do attempt to model a process in terms of roles that have to be carried out within that process. This modelling approach while not explicit in terms of the logical execution of tasks, as required for simulation, does place the interactions or roles of a person with a process more to the fore than a sequential

task based model [51]. Such an approach would be expected to lessen the cognitive jump that a user has to make to visualise their interactions within the model and in turn the real process, and should improve their ability to reason over information contained therein. Therefore, by developing a technique that centres around the interactions of a user and their role within a system, while also visually modelling the logical sequences of execution of tasks as would be contained within a simulation model, the model should allow a user to directly reason over their interactions with the complex information of the simulation model.

2.2.6 The GRAI Method

The GRAI (Graphe a Resultats et Activites Interlies) model was originally developed from the theory of complex systems by Doumeingts [52] and was originally designed to aid in the design of production management systems but has been used in various areas where there is a need for co-ordination between different groups [53]. This GRAI model along with the following five elements forms part of the GRAI Integrated Methodology (GIM) [54]:

- GIM modelling framework;
- GIM reference architecture;
- GIM modelling formalisms;
- GIM structured approach;
- GIM CASE tool.

2.2.6.1 GRAI Model

Of the various elements of the GIM it is the GRAI model that will be discussed here. The GRAI model is made up of three sub-systems, those being the physical, decision and information systems. Figure 2.6 shows the three sub-systems of the GRAI model and their interactions. This model which is made up of inputs from both control and systems theory allows for the description of the

structure of both a manufacturing system and it's control system in a generic way [55].

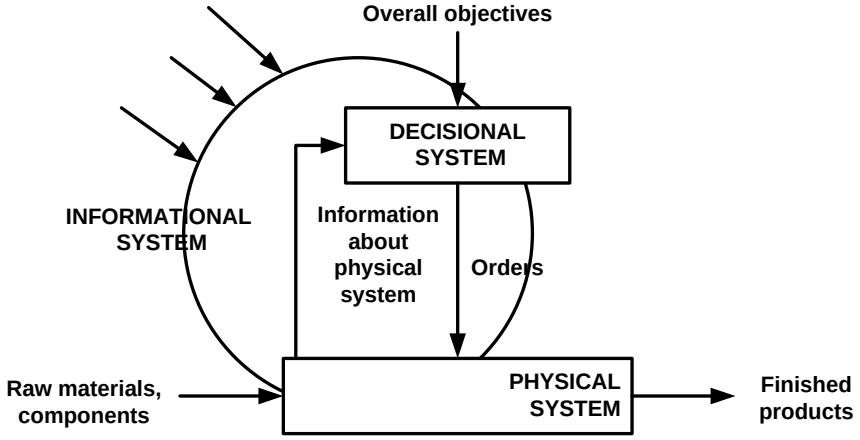

Figure 2.6 GRAI Model sub-systems

The overall GRAI model is shown in Figure 2.7 and is divided into the following systems:

- Physical system;
- Decision System;
- Operational system;
- Information System.

The physical system is used to model the process of transforming input objects into output or finished objects by means of a flow of these objects or materials through a model of the physical layout of equipment [56]. Such a physical model can contain resources such as personnel types, workplaces and products and forms the basis of the GRAI model [57].

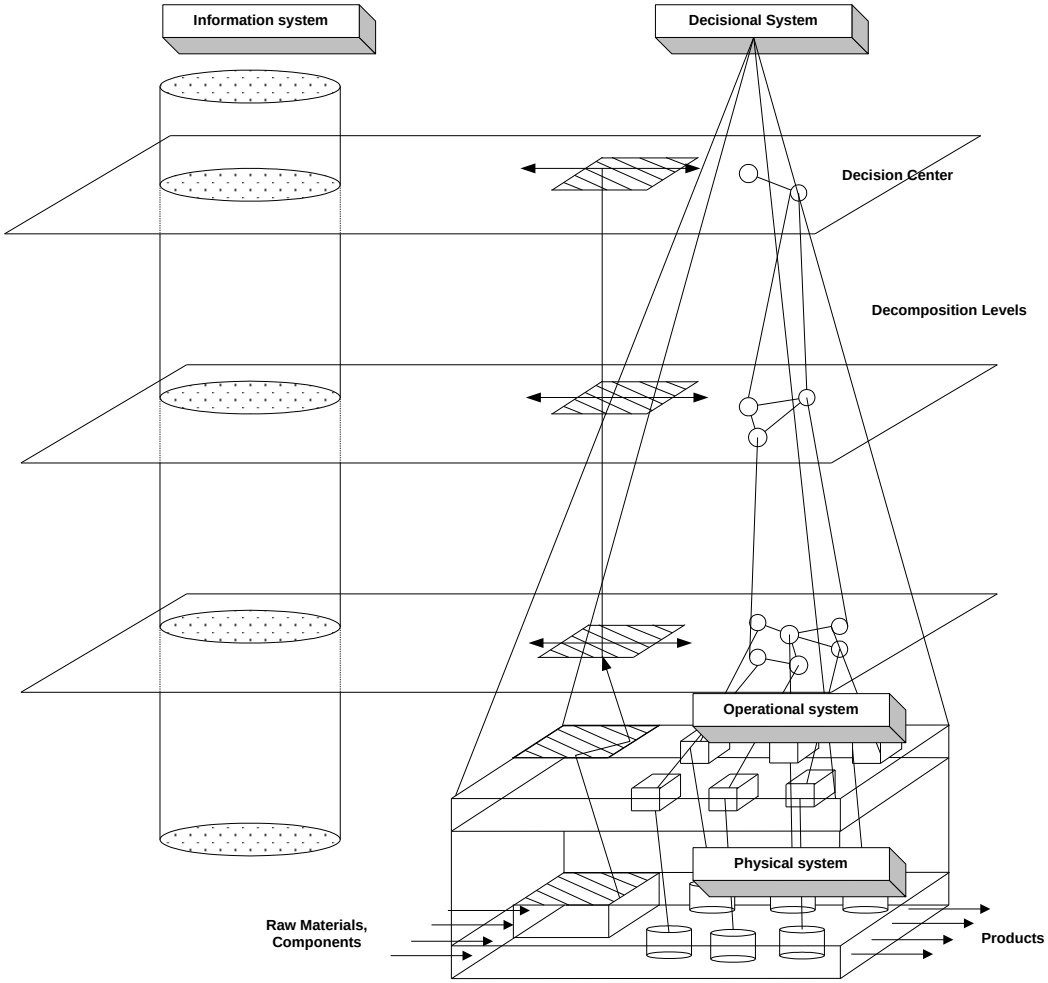

Figure 2.7 GRAI Model

The decision system, based on the GRAI conceptual reference model, Figure 2.8, is made up of a series of decision centres, which are locations of decision making for those managing the physical system. The GRAI-Grid is a decisional matrix, which represents this general decision making structure of the physical system [58].

This GRAI grid is divided along a vertical and horizontal axis [59]:

- The vertical axis divides the decisions into three types strategic, tactical and operational;
- The horizontal axis deals with the functional decomposition and gives a business process view of the system.

The GRAI method also accounts for a decision system that runs both periodically and in real time. At a higher-level, decisions would be made periodically, while at a lower or operational level a system will be event driven and run in real time. This operational level is shown in Figure 2.7 as the operational system, which the GRAI-grid does not model [55].

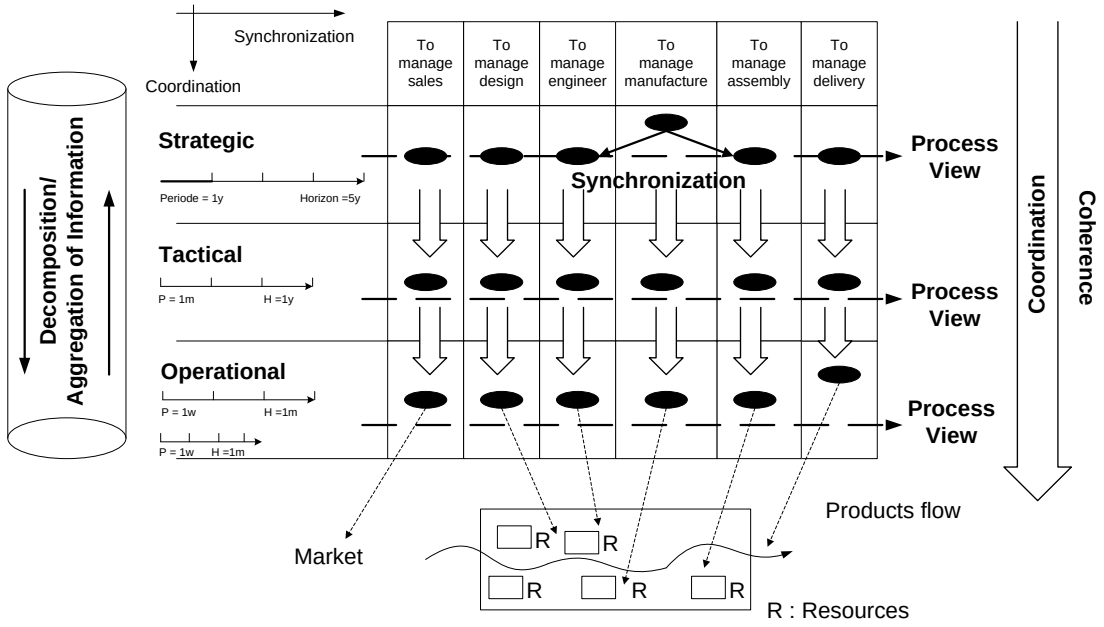

Figure 2.8 GRAI Conceptual reference model

While this GRAI model gives the user a generic structure it has to be used in conjunction with a control model to allow the control of the physical system, (see Figure 2.9) [57]. As stated earlier the physical system is concerned with the flow of objects or parts through resources and the execution of activities on these resources leading to the transformation of the objects.

The GRAI model tries to control this physical system in an optimal way by controlling and synchronising the flow of products in relation to the availability of resources. This synchronisation between the two functions is carried out by a third element known as “To Plan”. Therefore the three most basic elements involved in the control of the physical system are “To Plan”, “To manage the resources” and “To manage the products” as shown in Figure 2.9. Therefore to

get a complete GRAI grid in theory it is necessary to combine both the GRAI reference and control models [56].

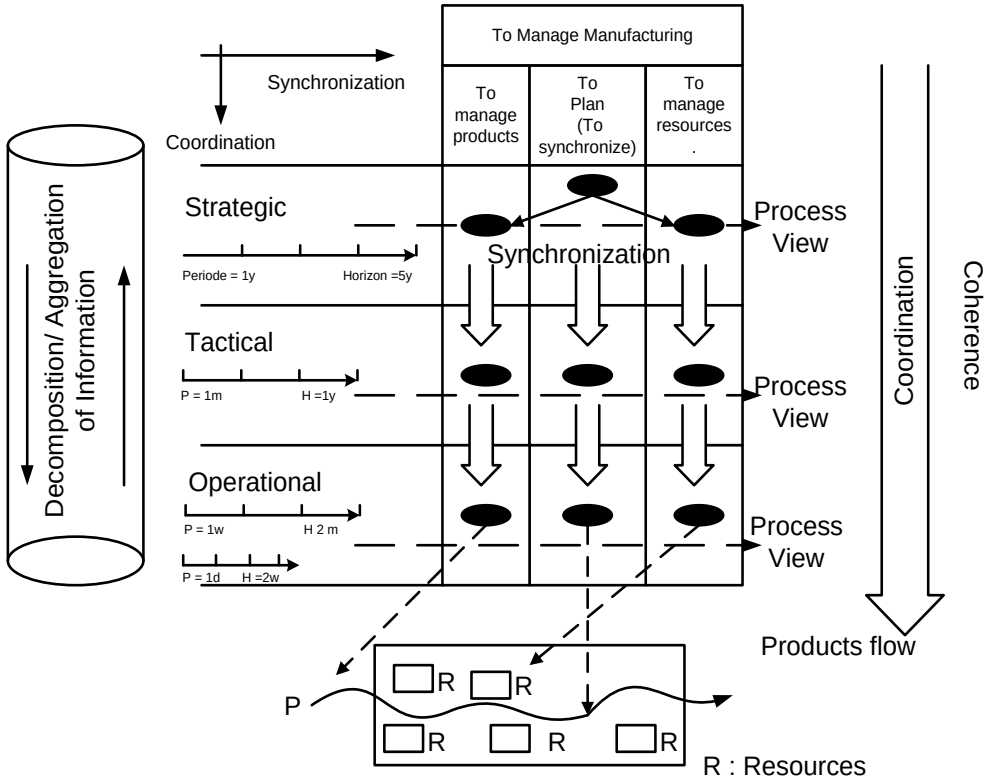

Figure 2.9 GRAI Control Model

The information system contains all the information for the running of a system and therefore is structured in an identical hierarchical manner to the decision system [56].

Within a GRAI grid a decisional centre is taken as the intersection of a function and a level. At any one of these levels the decisional centre can be decomposed into a micro model, using the decomposition criteria of the micro model having a physical, decisional and information component. In this instance the physical part is composed of the view of the system seen by the decision maker, this would differ between an operator and a manager for instance. The decisional part is then made up of the decision maker be it a person or a machine and the various interacting elements that help in the decision making process. The information

system then shows the flows of information before, during and after the decision at that particular level. This description of a decision centre is done by means of the GRAI net as shown in Figure 2.10. The GRAI model has been used to analyse and improve various aspects of business and production processes, [53], [57], [59].

The GRAI model gives a global description of the enterprise. In terms of BPR, if a comparison is made with business process modelling techniques such techniques model a process in one dimension while modelling a system with the GRAI modelling method is modelling in n dimensions [58]. However, the GRAI method is primarily focused on the decisional structure of a manufacturing system. Therefore, it does not adequately model the physical system or flow of work to allow the development of a communicative model that would accurately and intuitively model a discrete event system for the purposes of capturing and aiding in the communication of system issues in the pre-coding stages of a simulation project. The modelling of the decisional structure of a discrete event system is however important as modern systems rely heavily on such decisional systems for control and regulation. As a result, it is felt that the graphical representation of such a decisional system and its interactions with the flow of work through a discrete event system would be vital to aid in the communication of system issues between a model developer and system personnel.

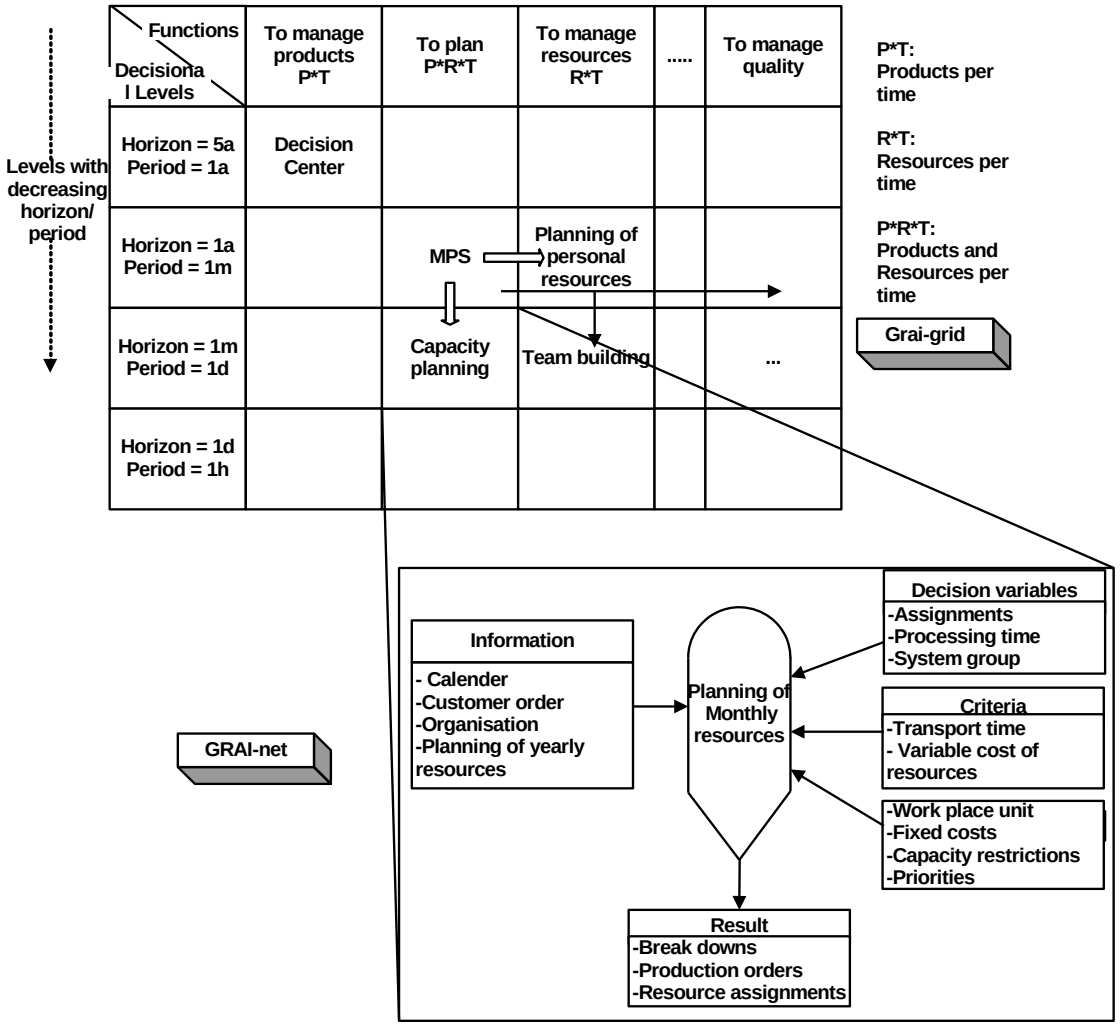

Figure 2.10 GRAI grid and GRAI net

2.2.7 IEM modelling method

The IEM method uses the integrative potentials of information processing technology to integrate a variety of organisational goals such as organisational development, quality management, information systems planning and cost control within the one modelling method. The business process and relevant information, which is represented in one integral model, forms the core element to this method. To this core model the organisational structure, quality management system, cost structures, control system and information system are represented by means of user views, which are directly related to the core element of the model. The IEM method achieves this by allowing the user to adopt different views of a company while also allowing the analysis and

optimisation of the various interactions and interdependencies between them [60]. The next section describes the modelling language and rules that enable a user to create such an integrated model.

2.2.7.1 IEM Generic Classes

The IEM modelling method is based around three generic object classes, which are Product, Resource and Order, Figure 2.11. These represent the following [61]:

- Product classes represent the main output from this enterprise process or products of the enterprise;
- Resource classes represent the means including organisational which are needed to carry out any activity in an enterprise;
- Order classes represent planning and control information.

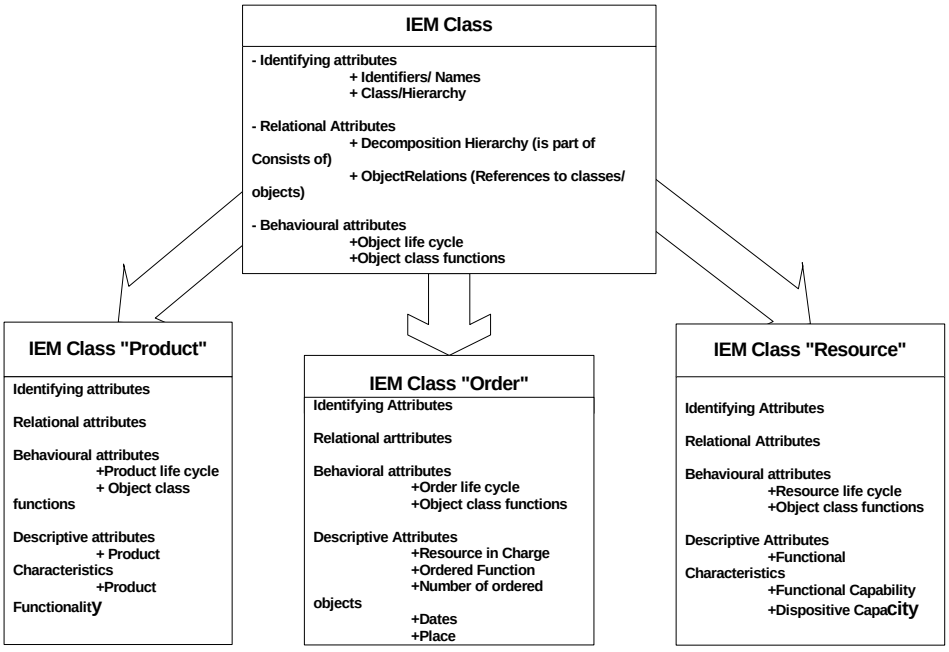

Figure 2.11 IEM Generic Class structure

2.2.7.2 Product Class

The product class can be specialised into subclasses of “product”, this allows the user to define both customary and specific subclasses. Complicated product structures are also described by means of “is part of” and “consists of” relations between the different “product” subclasses [61].

“Product” class objects represent the products of a company, thus all necessary information required to manufacture the product including the information on product characteristics as well as quality information is represented. Depending on the level of detail required, a product class can contain a representation of all relevant product states, the necessary functions to process the products, the logical sequence of the functions and of the relations with other objects, and object classes.

2.2.7.3 Order Class

The order class represents all the information required to plan and control enterprise functions [61]. The “order” class can be specialised to provide a specific hierarchical model of orders within a company by defining sub classes.

Within such a model the user can describe the planning of control functions for each order class, the processing of the orders (information) and the generation, of new orders (planning and control information). In this way the “order” class represents the information relevant to planning, controlling and supervising a process within a company.

This information concerns the planning, authorisation and control of:

- Functions to manufacture products;
- Resources required to execute the functions, including the necessary functions to prepare and supply the resources;
- Functions to process objects of the class “order” themselves.

2.2.7.4 Resource

The resource class represents all things, facilities, persons and information that are able or necessary to execute functions. This class can also be specialised according to company specific requirements by using sub-classes. Each description of a resource class should include the relevant states of resources, the functions that are required to achieve or maintain the states, and the logical sequence of the functions. The description should also include all relevant relations to objects and object classes that are used in the maintenance or execution of services.

2.2.7.5 IEM Main Views

The core of the IEM model comprises of two views those being the business process model view and the information model view, Figure 2.12 [60].

The business process model view presents a functional model of a process. This view focuses on the tasks that are to be executed on both business processes and objects. This view describes all possible states of objects, their related functions, activities and their various logical connections.

The information model view concentrates on objects describing data. In this way the structure of the objects and their attributes are described. This view also represents the descriptions of the various states that are used in process representations in the “business process model”. The core views are linked by referring to the same objects and activities in both views, however, each view represents them in different ways, level of detail and context.

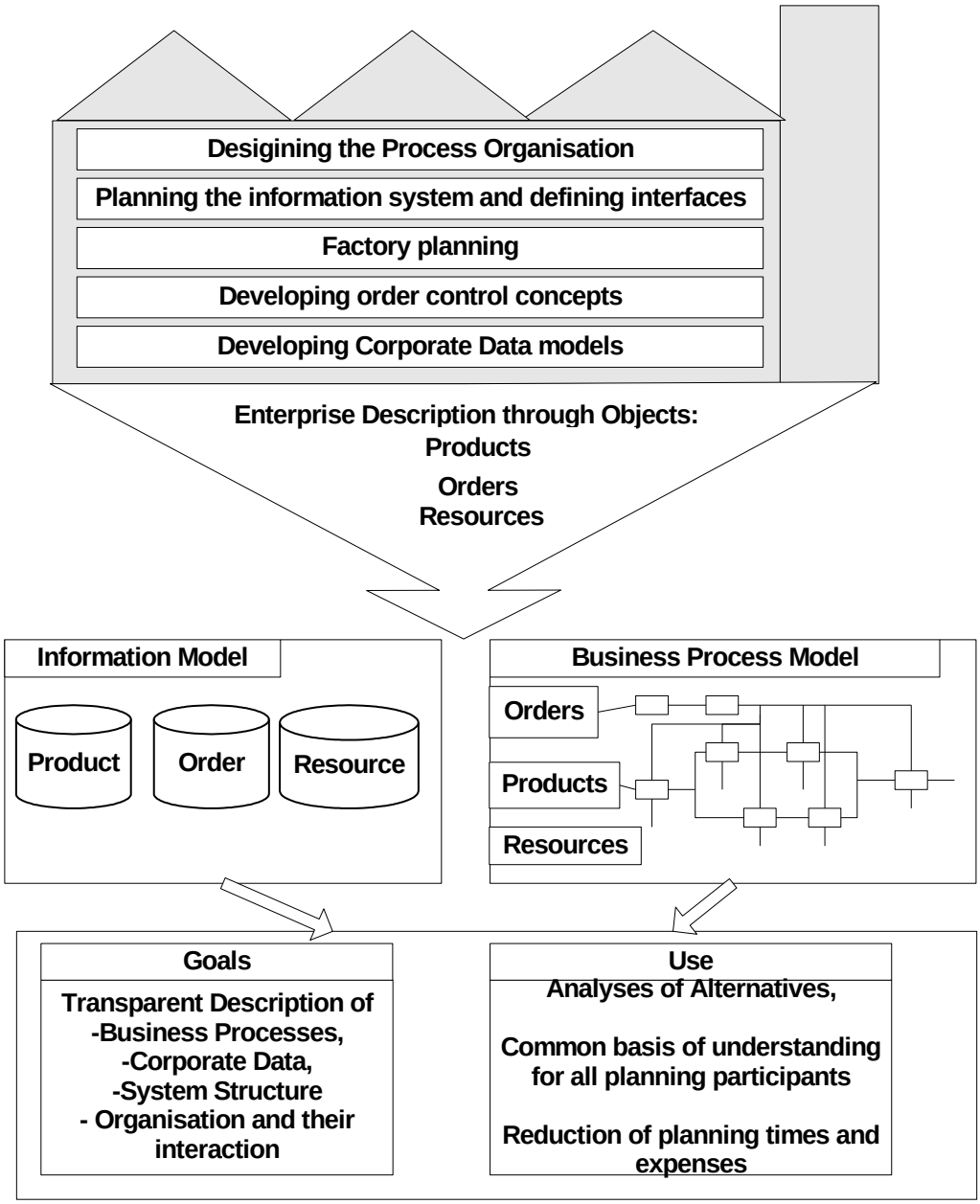

Figure 2.12 IEM Main Views

2.2.7.6 IEM Object Interactions

The IEM method describes an activity as being anything that happens in a manufacturing enterprise. It allows for the description of such an activity on three levels as shown in Figure 2.13 those being [61]:

- An action, which is a description of any task, process step or procedure;

- A function, which describes the processing of objects as a transformation from one determined state to another determined state;
- An activity, which specifies the order that controls the execution of the function and the resources that are in charge of the execution of the function.

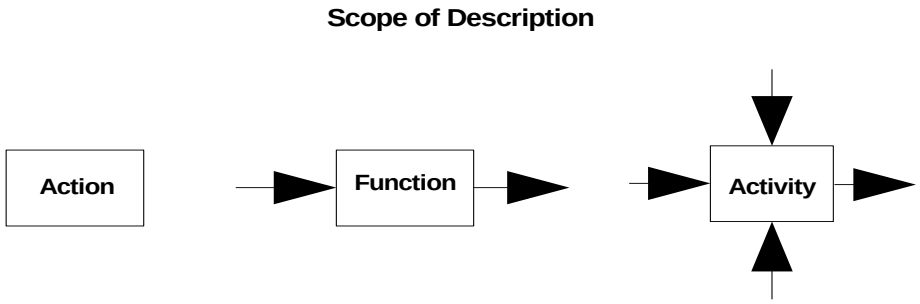

Figure 2.13 An IEM Action, function and activity

The generic activity shown from Figure 2.13 is further expanded in Figure 2.14. In this Figure the beginning and ending states are connected with the action rectangle by arrows from left to right. An order state description and a dashed vertical arrow represent the control of the activity from the top. A resource state description and a dashed vertical line represent the resource assigned for the execution of the action from the bottom.

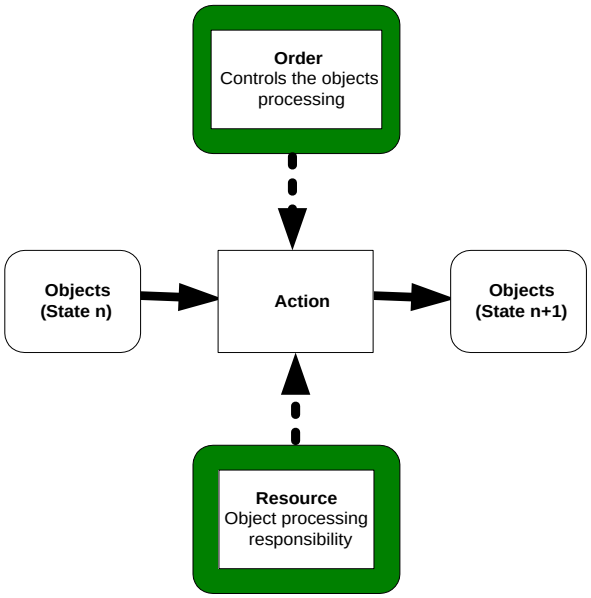

Figure 2.14 Generic Activity

This generic activity model represents the processing of objects of the product, order or resource class and indicates the interaction of the various objects while processing [61]. The generic activity model can be expanded as in Figure 2.15 to model a procedure or process, which is a series of linked generic activities as shown previously in Figure 2.14.

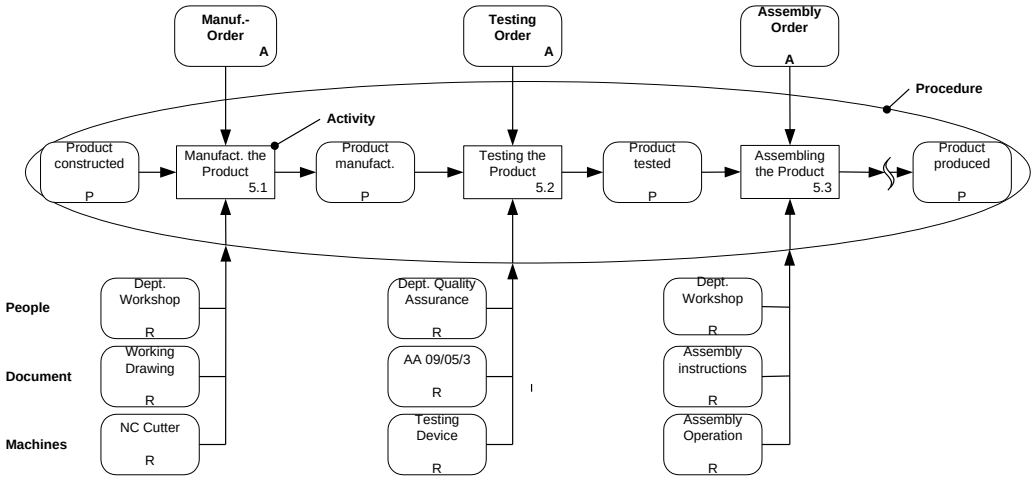

Figure 2.15 IEM Procedure/process

The IEM modelling constructs for the process model view are shown in Figure 2.16. This shows how the special linking constructs, actions, functions and activities are combined to represent business processes. Aggregation and decomposition are also supported in this view.

The IEM technique is capable of modelling discrete processes. The technique also accounts for the interaction of both control and resource elements in the execution of activities. However, the technique is limited in its three modelling constructs and lacks the inclusion of an element such as a queue which would be vital to the modelling of a discrete event system for the purpose of a gathering requirements or building a conceptual model for the purposes of a simulation project. As a result, the technique, while being capable of modelling discrete systems, is not capable of capturing and representing such detailed interactions as those inherent in complex discrete event systems. Therefore, it is not ideally suited to the purpose of communicating system issues between model developers and system personnel involved in a simulation project. The IEM

modelling technique is implemented in the MOOGO process modelling tool, which will be presented in section 2.3.1

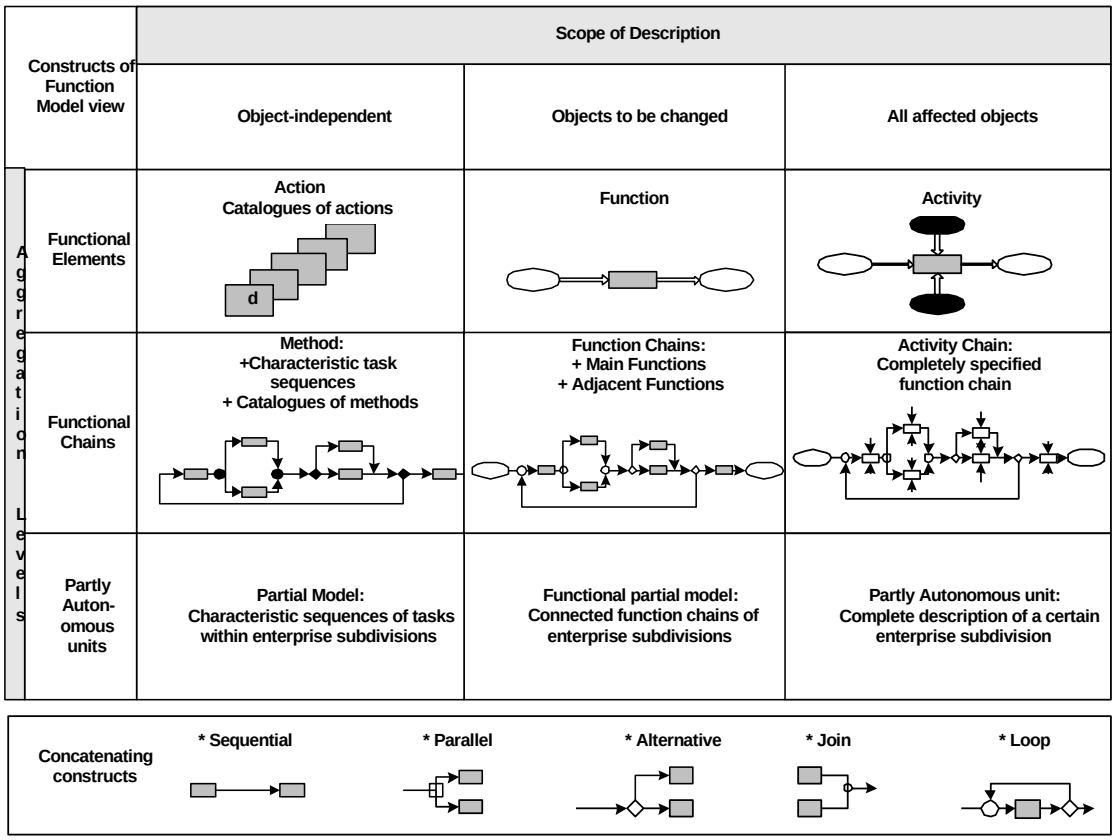

Figure 2.16 IEM Modelling constructs

2.2.8 Event driven process chains

Event driven process chains (EPCs) [62], [63], are a graphical business process description language. EPCs take their name from the diagram shown in Figure 2.17, which shows the structure and flow of a business process. In this modelling technique a process consists of sequences of functions and events. A “function”, the basic building block of an event driven process chain, corresponds to an activity that needs to be executed, while an “event” describes the situations both before and after a function is executed. In this way an EPC consists of the capturing, representation and sequencing of activities that are to be executed in the progressing of a process.

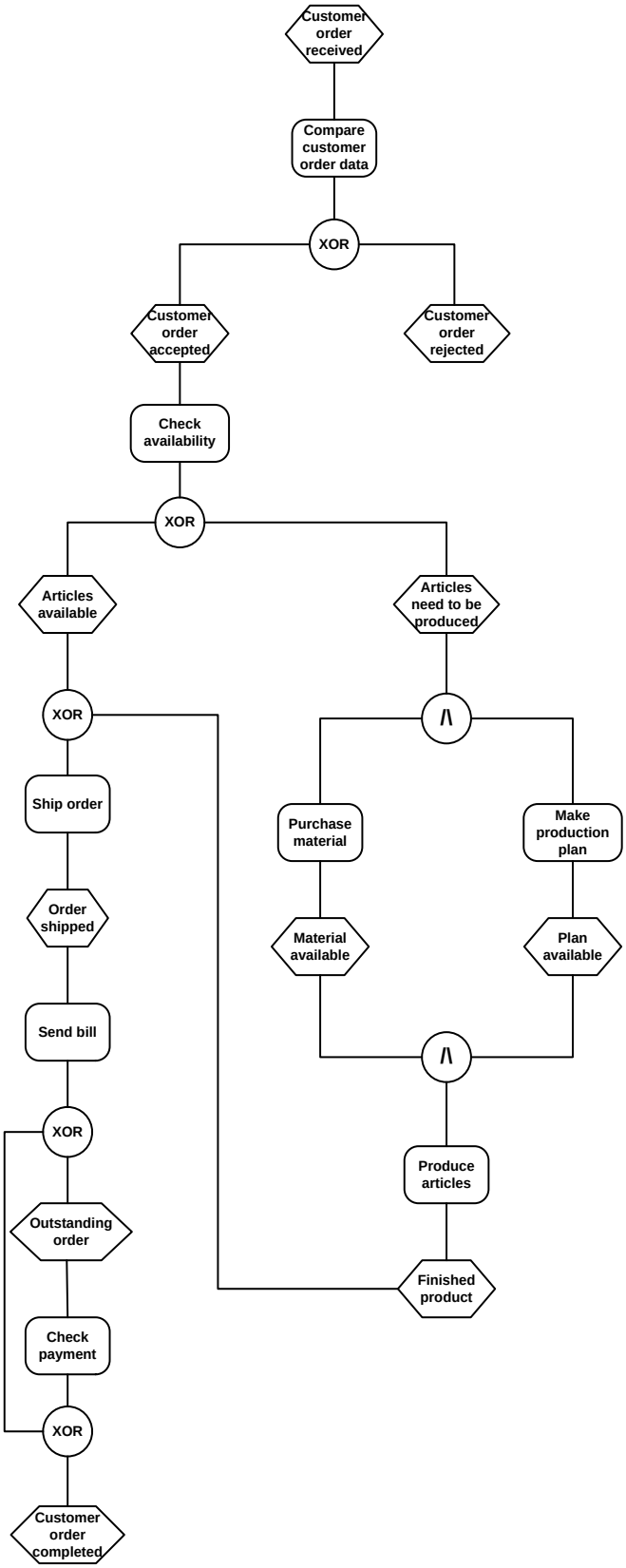

Figure 2.17. Event Driven Process Chain [64]

Therefore functions are linked by events. These events act as triggers for following functions, which themselves are the results of other functions. The only exceptions to this rule of an event being a product of a function are the initial event or events that trigger the process. In addition to these two constructs logical connectors of types “AND”, “OR” and “EXCLUSIVE OR” can be used within an EPC to connect functions and events. In this way an EPC can be used to show complex flows of control within a particular process. This technique aims not to formally describe business processes but to describe the processes in terms of their business logic in a way that is easy to understand and use [65].

The process modelled in Figure 2.17 is a customer order transaction example from Nuttgens et al [64]. The process begins with an event called “Customer order received”. From this the first function, “Compare customer order data”, is executed and as a result of this the order is either accepted or rejected. This is modelled by the XOR connector, which shows the user that after executing the “Compare customer order data” function either one of the two events, “Customer order accepted” or “Customer order rejected” will be executed. If the “Customer order rejected” event is executed the process stops. However, if the “Customer order accepted” event is executed the process continues with the availability of the parts being checked, represented by the “Check availability” function. If it is found here that the required parts are not available then the “Articles need to be produced” event is generated. This leads to the two functions, “Purchase material” and “Make production plan”, which are executed in parallel. On the execution of these two functions both the “Plan available” and “Material available” events are generated. These two events cause the “Produce articles” function to be executed which leads to the generation of the “Finished product” event. In other words the parts are now available.

From here another XOR connector shows that the system will progress to the “Ship order” function if either the “Articles available” or the “Finished product” events are generated. On executing the “Ship order” function the “Order shipped” event is generated.

After this event the “Send bill” function is executed, the bill being sent to the customer. This results in the “Outstanding order” event being generated. However at this stage in the model an XOR connector is used to set up a loop. This loop is a method whereby a check is made to see if the bill has been paid. This is modelled using the “Check payment” function, if the result from this function is positive then the “Customer order completed” event is the result and the process stops, otherwise the loop continues, denoted by the XOR connector, until the result is positive. Figure 2.17 illustrates how easy event driven process chains are to read and shows why many have accepted them as a modelling technique for Business Process Reengineering (BPR) projects.

Figure 2.17 shows a basic event driven process chain. This type of EPC can be extended by the inclusion of further elements of description. Examples of these extensions are data flows, organisation units and systems. Figure 2.18 shows an EPC with these extensions. Such EPCs are known as extended event-driven process chains (eEPCs) [66].

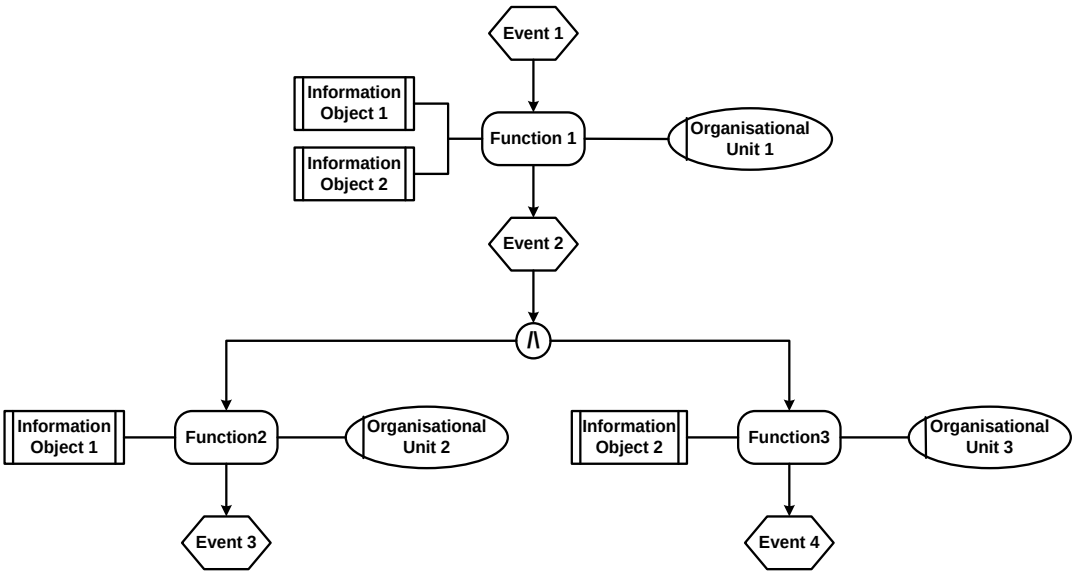

Figure 2.18. Extended event driven process chain [66]

Another extension of EPCs is that of an object-oriented event driven process chain (oEPC) [67], which aim to preserve the capabilities of standard EPCs while integrating object-oriented elements. In the oEPC method, business objects replace the functions of standard EPCs. This is shown in Figure 2.19. Within this

method business objects and events/rules are defined as object classes and therefore can be described in greater detail by the addition of attributes and operations to the respective classes [64].

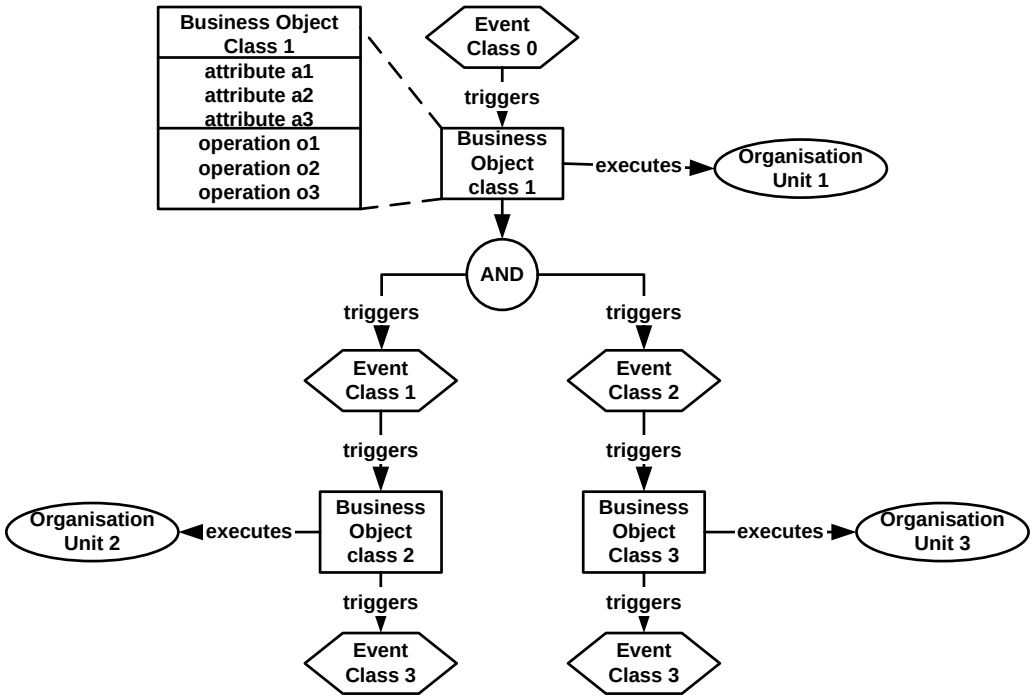

Figure 2.19 Object-Oriented Event Driven Process Chains [64]

Organisation units, resources, business objects and Boolean operators can be used in the oEPC method as in the standard EPCs. The interaction between business objects is based on event driven message exchange. These messages reflect the decision and control mechanism of a business process. The business process model shown in Figure 2.19 uses oEPC symbols and graphically illustrate the control flow defined by event driven messages [64].

EPCs are capable of accurately representing the flow of activities that are associated with the execution of tasks within a discrete event system. However, the technique does not allow for the modelling of the change of state of a discrete event system or the modelling of the control of discrete systems. Therefore, while the EPC technique is capable of accurately representing certain areas of a complex discrete event system, it lacks the ability to capture and represent all of the aspects that would allow it to function as a communicative and representative technique for use by a model developer and system personnel during the initial

requirements gathering or conceptual modelling phases of a simulation project. EPCs form the basis of the ARIS process modelling tool, one of the most popular process modelling tools, this tool is presented in section 2.3.2.

2.2.9 IDEF Suite of Modelling Methods

The Integration Definition (IDEF) modelling method was developed by the U.S. Air Force as part of an Integrated Computer Aided Manufacturing (ICAM) program during the 1970"s. This research identified the need for better analysis and communication techniques for people involved in improving manufacturing productivity. The aim was to provide an integrated suite of tools for the purpose of modelling activities within an organisation.

The IDEF method was developed to support the better communication, understanding and analysing of systems. This method involves functional, informational and dynamic modelling methods. This modelling approach helps people involved in improving manufacturing productivity to understand different aspects of a system such as:

- The activities and their relationships within a system;
- The informational requirements of a system;
- The behaviour of functions and information interacting over time.

The IDEF methods have been further developed by Knowledge Based Systems Incorporated (KBSI) and provide an integrated suite of tools for the purposes of modelling activities within an enterprise. The suite of methods consists of the following:

- IDEF0 (function modelling method): Used for the structured representation of the activities within a system [68];
- IDEF1 (information modelling method): Used for the generation of an information model, which represents the structure and semantics of information within a system [69];
- IDEF1X (data modelling method): Semantic data modelling technique [70];

- IDEF3 (process flow and object state description capture method): Used for the documentation of how systems work [72];
- IDEF4 (object-oriented design method) Used as a software design method [73];
- IDEF5 (ontology description capture method). Used to capture information to support enterprise ontology's [74].

A number of researchers have shown that methods from the IDEF approach could be used to support simulation. For instance Jeong [75] used both IDEF0 and IDEF3 to develop an Optimised Simulation-Based Scheduling System (OSBSS), while Perera and Liyanage [76] used IDEF0 and IDEF1X to address the rapid collection of input information for the simulation of manufacturing systems. Also, other researchers such as van Rensburg and Zwemstra [77] and Al-Ahmari and Ridgway [78] have demonstrated the use of IDEF0, IDEF1X and IDEF3 to support simulation for manufacturing and system design. Furthermore, it has been suggested [77] that the use of IDEF techniques in simulation modelling enhanced the quality of simulation models and helped to reduce the time needed to generate simulation models. The following section outlines the IDEF0 and IDEF3 methods as these are most directly applicable to simulation modelling.

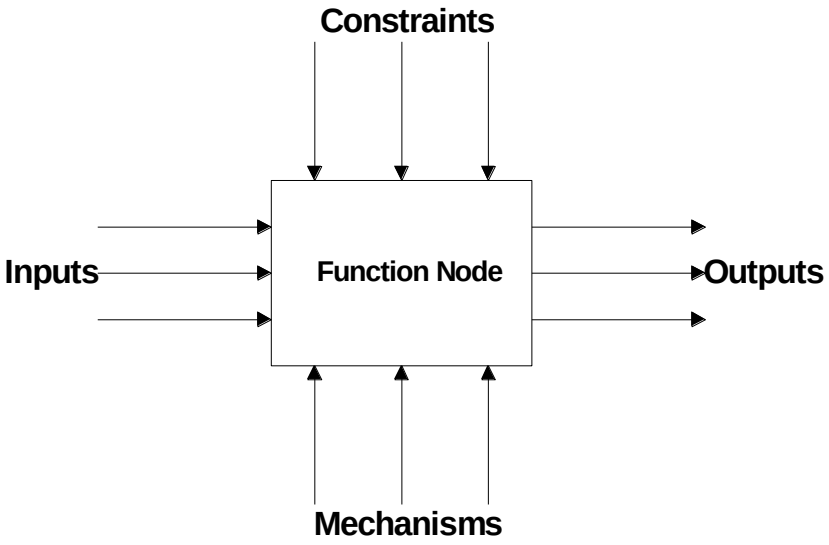

Figure 2.20 IDEF0 Model [68]

2.2.9.1 IDEF0

The IDEF0 functional modelling method was developed from the SADT (Structured Analysis and Design Technique) method, to allow the analysis and communication of the functional aspect of a system [68].

IDEF0 can be used as both a communication and an analysis tool. As a communication tool it allows decision making through simplified graphical devices. As an analysis tool it identifies the functions performed, what is needed to perform those functions and what the current system does [68]. IDEF0 is a modelling technique used for [68]:

- Performing system analysis and design at all levels;
- Producing reference documentation concurrent with developments to serve as a basis for integrating new systems or improving existing systems;
- Communicating between analyst, designer, user and manager;
- Allowing coalition team agreement to be achieved by shared understanding.

Furthermore, the IDEF0 modelling method establishes the scope of analysis either for a particular functional analysis or for future analyses from another system perspective. As a communication tool, IDEF0 enhances domain expert involvement and consensus decision making through simplified graphical devices. As an analysis tool, IDEF0 assists the model builder in identifying the functions performed and highlights what is needed to perform them. Thus, IDEF0 models are often created as one of the first tasks of a system development effort.

The approach adopted in IDEF0 is to describe each process (or activity) as a combination of processes, inputs, controls and mechanisms, as in Figure 2.20. At the highest level, the representation may be of an entire process. This representation may then be subdivided into several more activity boxes or sub-processes. In such a fashion, the breakdown continues as shown in Figure 2.21.

Until the point is reached where sufficient detail is at hand to make the changes that might be needed.

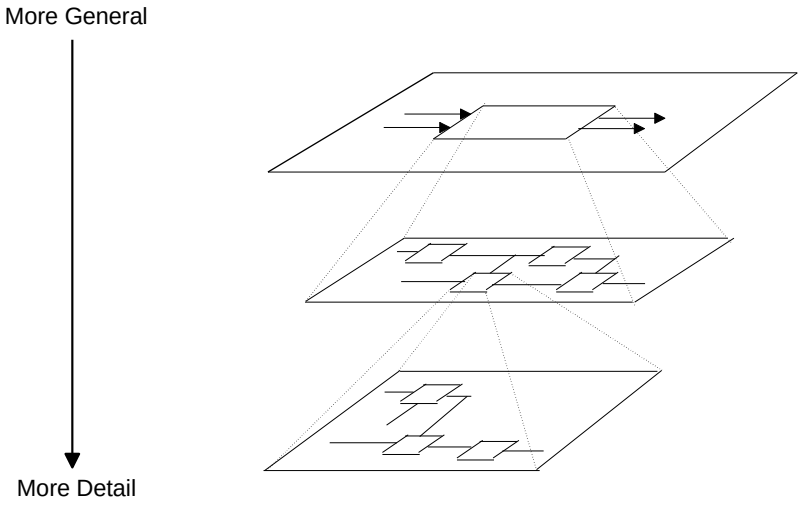

Figure 2.21 IDEF0 Decomposition [68]

The IDEF0 model shown in Figure 2.22 is based on a simple syntax. Each activity or function is represented by a box, these boxes are interconnected by arrows. An arrow may be an input, control, output, or mechanism, depending on where it enters the box, see Figure 2.20. Inputs are defined as items which are consumed by a function. They are therefore more often material than informational. Information appears as controls, which help to constrain functions or influence how they are performed. Every box has at least one control arrow. Outputs may be informational or material. Mechanisms typically include the non-consumable resource inputs, such as tools or human resources, and indicate how a function is performed. In IDEF0, arrows represent data constraints, rather than flow or sequence [68]. However, feedback, iteration, continuous processes, and overlapping functions are easily portrayed, see Figure 2.22. For example, the input and control arrows on function, “A”, represent data or objects needed to perform some part of the function. The control being an output from function, “C”.

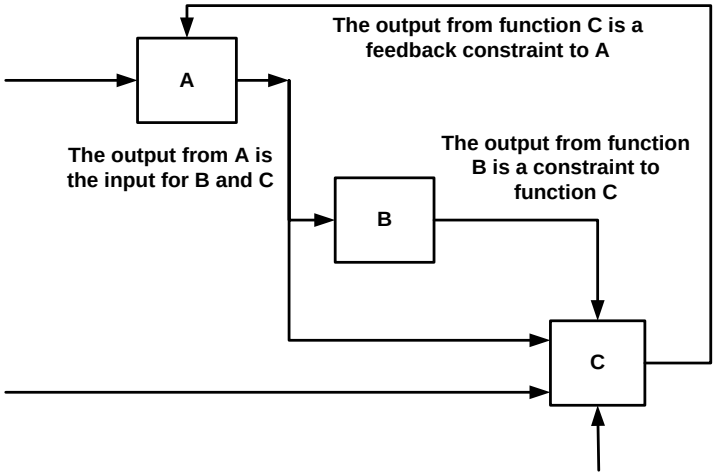

Figure 2.22 IDEF0 Example [68]

IDEF0 allows for the visual modelling of the decisions and activities in a system. However the technique again lacks the ability to model the various other aspects of a complex discrete event system, such as the workflow and control flow, that are necessary to capture and communicate during the conceptual modelling or requirements gathering phase of a simulation project. The technique also lacks the capability to graphically represent the division of a system into multiple processes.

2.2.9.2 IDEF3 Process method

The IDEF3 Process Description Method provides a mechanism for collecting and documenting processes. IDEF3 captures precedence and causality relations between situations and events in a form natural to domain experts, by providing a structured method for expressing knowledge about how a system, process, or organisation works [71].

This process knowledge is structured within the context of a scenario, making IDEF3 an intuitive knowledge acquisition device for describing a system. IDEF3 captures all temporal information, including precedence and causality relationships associated with enterprise processes. The resulting IDEF3 descriptions provide a structured knowledge base for constructing analytical and

design models. These descriptions capture information about what a system actually does or will do, and also provide for the organisation and expression of different user views of the system [71].

There are two IDEF3 description modes, process flow and object state transition network. A process flow description captures knowledge of “how things work” in an organisation, *e.g.*, the description of what happens to a part as it flows through a sequence of manufacturing processes. The object state transition network description summarises the allowable transitions an object may undergo throughout a particular process. Both the process flow description and object state transition description contain units of information that make up the system description. These model entities, as they are called, form the basic units of an IDEF3 description [71].

The Process Flow Description

An IDEF3 Process Flow Description captures a description of both a process and the network of relations that exists between processes, within the context of the overall scenario in which they occur. The development of an IDEF3 Process Flow Description consists of expressing facts, collected from domain experts [71].

The following example illustrates how the building blocks of the IDEF3 method can describe a scenario typically found in a manufacturing environment. The situation to be described is a painting and inspection process associated with applying primer paint to a part that will become an element of a subassembly for a piece of heavy construction equipment. Figure 2.23 is the graphical representation of the scenario (story) told by a paint shop supervisor when asked to describe: “What goes on in the primer shop?” [71].

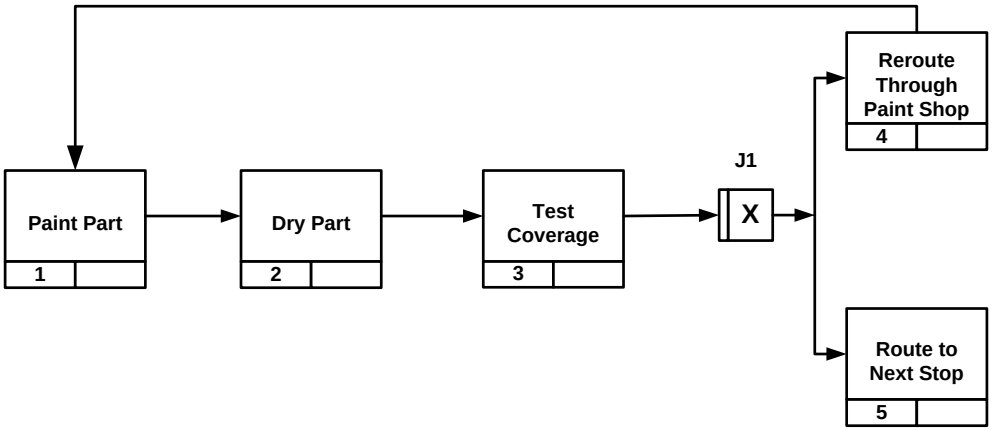

Figure 2.23 IDEF3 Unit Of Behaviour (UOB) Description

The story the example describes follows:

"Parts enter the shop ready for the primer coat to be applied. We apply one very heavy coat of primer paint at a very high temperature. The paint is allowed to dry in a bake oven after which a paint coverage test is performed on the part. If the test reveals that not enough primer paint has been sprayed on the surface of the part, the part is re-routed through the paint shop again. If the part passes the inspection, it is routed to the next stop in the process [71]."

The following elements are shown in Figure 2.23, the Unit Of Behaviour (UOB), represented by boxes such as "Paint Part". These are the basic building blocks used to model the various activities in an IDEF3 process flow. The arrows that join these UOBs are used to show the logical flows, while smaller boxes as shown in Figure 2.23 represent the junctions used to bring logic into the process flows.

There are components that are not directly represented in Figure 2.23 namely the decomposition and elaboration components of IDEF3. Each UOB can have associated with it both descriptions in terms of other UOBs and a description in terms of a set of participating objects and their relations. The former are known as decompositions of a UOB and the latter as elaborations of a UOB. Intuitively, a decomposition is a closer look at some given UOB within a larger diagram. This decomposition may be of some UOB in the scenario (top level) diagram or it may be of a UOB in a decomposition. More precisely, a decomposition of a given

UOB is a more fine-grained IDEF3 representation of that UOB. Multiple views are allowed in IDEF3 as it is meant to be used as a description capture method [71].

An elaboration is an element of the IDEF3 description that captures the objects that participate in a particular activity and the facts and constraints that are defined on these objects and on instances of that activity. Each element of an IDEF3 description can have an elaboration, which can simply be text based or use the IDEF3 elaboration language. Resource requirements of systems are also captured in the elaboration [71]. The lack of a graphical representation of resources within the IDEF3 makes it difficult to graphically represent the detailed interactions that may be present in a complex discrete event system. Therefore such interactions have to be represented within an elaboration. This is achieved as aforementioned by either using a textual description or the IDEF3 elaboration language, which is not easy to reason with and does not lend itself to being readily understood by persons who are unfamiliar with it.

The Object State Transition

Object state transition network (OSTN) diagrams capture object-centred views of processes that cut across the process diagrams and summarise the allowable transitions. Figure 2.24 shows a sample OSTN diagram [71].

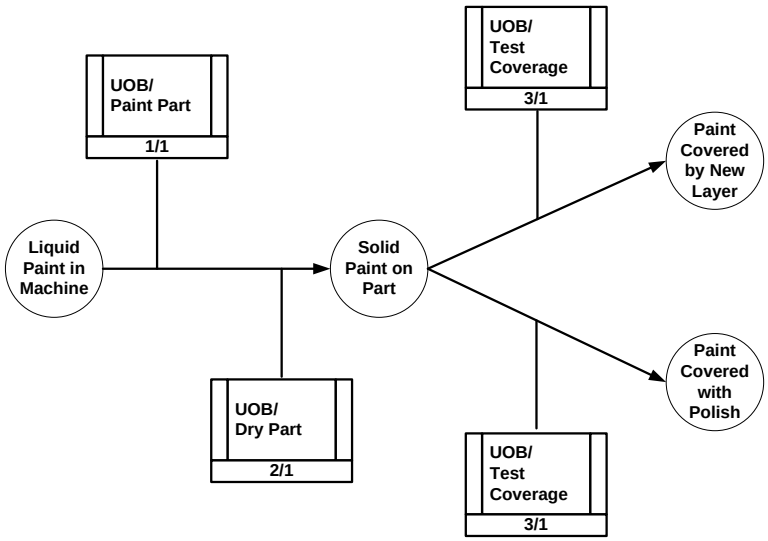

Figure 2.24 Example IDEF3 Object State Transition Network Diagram [71]

Object states and state transition links are the key elements of an OSTN diagram. Object states are represented by circles and state transitions are represented by the lines connecting the circles. An object state is defined in terms of the facts and constraints that need to be true for the continued existence of the object in that state and is characterised by entry and exit conditions. The entry conditions specify the requirements that need to be met before an object can pass into a state. The exit conditions characterise the conditions under which an object can pass out of a state. The constraints are specified by a simple list of property/value pairs or by a constraint statement. The values of the attributes must match the specified values for the requirements to be met [71].

State transitions represent the allowable transitions between the focus object states. It is often convenient to highlight the participation of a process in a state transition. The importance of such a process constraint between two object states can be represented in IDEF3 by attaching a UOB referent to the transition between them [71].

The IDEF3 process modelling technique allows for the capture and graphical representation of both the transition of states through a discrete event system and the activities associated with such state transitions. However, the modelling of the control of a discrete system is also not graphically represented. Also, the IDEF3 modelling technique does not graphically allow for the representation of resources within either the process flow description or the OSTN views. Such resources are often very important in the modelling and simulation of a discrete event system as are queues, which again are not graphically represented. The IDEF3 elaboration language does allow for the capture and representation of resource interactions and queuing situations. However, the language is abstract in nature and does not lend itself to the communication of information to untrained users. As a result the IDEF3 technique is capable of capturing certain aspects of a complex discrete event system however it lacks the ability to graphically represent a number of important issues such as resource interactions and queuing. Therefore, the technique is not fully suited to the capture, representation and communication of all discrete system issues between both a

process model developer and system personnel in the early phases of a simulation project.

Both IDEF0 and IDEF3 have been used to develop simulation models directly from their process models using process modelling tools, WorkFlow Modeller for IDEF0 and ProSim for IDEF3 [79]. The ProSim process modelling tool is presented in section 2.3.3.

2.3 Process Modelling Tools

There are many commercial tools available to facilitate process modelling and reengineering. What follows is a review of a number of packages based on some of the techniques discussed in the previous section.

2.3.1 MOOGO

The MOOGO tool, representing the Method for the Object-Oriented Business Process Optimisation has been developed to support object oriented modelling with the method of Integrated Enterprise Modelling (IEM) introduced previously.

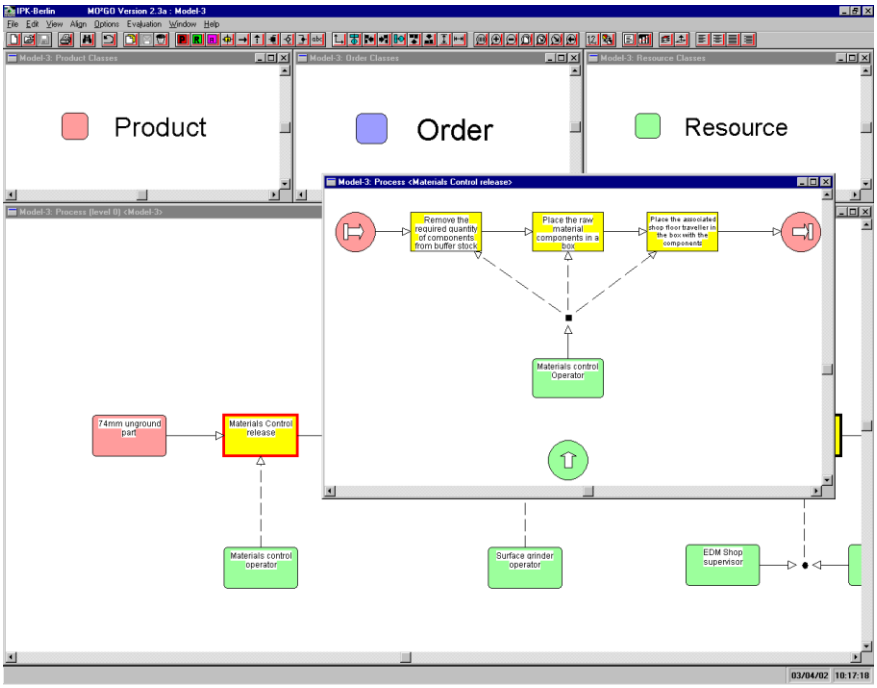

Figure 2.25 The MOOGO User interface

This tool enables the description, analysis and optimisation of operational structures and business processes by enabling the description and analysis of products, orders and resources, along with their related business processes. This is achieved by allowing the user to build a hierarchical model, using the IEM constructs, of the enterprise being examined. The MOOGO user interface is shown in Figure 2.25, which also shows a model and a sub-model, or more detailed model, of the “Materials Control release” function. This sub-model is shown in the smaller window within the screen shot.

MOOGO allows for the integration of both the planning and optimisation processes and the reusability of a model for any projects that concern corporate planning, such as information systems, controlling, quality management and organisational development. As the tool is used to develop an integrated model of a company, large amounts of data will be contained within the model. To allow users access to this information, different views of the integrated model may be selected, Figure 2.26 shows the structure of the MOOGO tool.

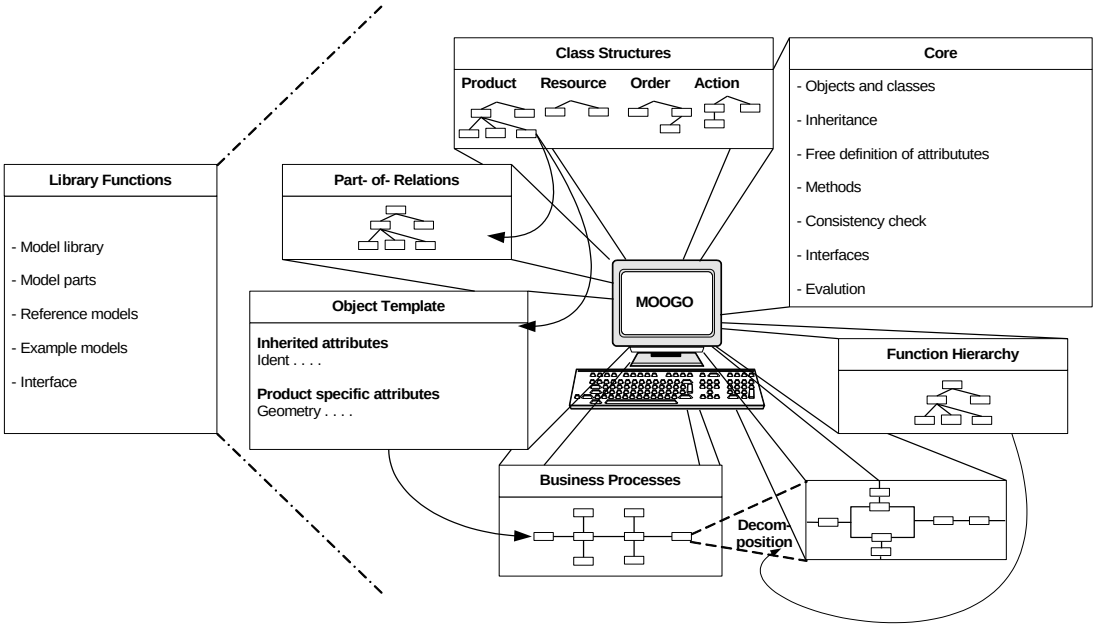

Figure 2.26 Structure of the MOOGO tool

These views relate to the information at the model core and include information systems, the process organisation, quality requirements and qualification requirements.

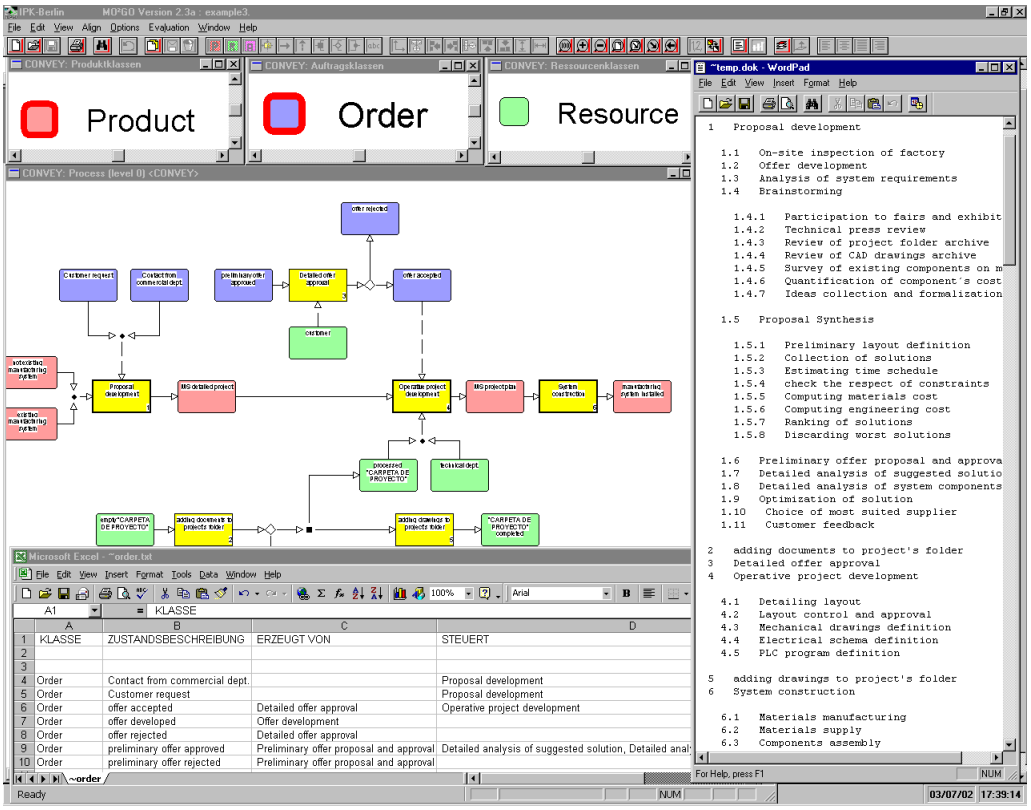

Figure 2.27 MOOGO user interface showing generated reports.

These views take the form of libraries of class structures and models and are supported by the evaluation functions of MOOGO.

These evaluation functions allow for the measuring of an attribute within a model, such as “process time” within a process sequence. They also allow for the generation of specific Tables such as order, which can describe the modelled orders, their associated processes for the production of each order along with the processes that are controlled by each order. Similar Tables can also be generated for resources. These Tables are all exported to EXCEL for viewing. Figure 2.27 shows the MOOGO user interface with both a report and a Table describing the orders in the model in question open for viewing. MOOGO also allows for the generation of quality manuals.

The IEM method, and therefore the MOOGO process modelling tool, develops models of business processes using the generic activity based on the order, resource and product elements. These elements allow for the development of a

sequential model of a discrete system but again lack the detailed modelling of resources and their interactions within a system. Another draw back of this technique for the purposes of communicative modelling is the perspective of the model. The IEM technique does not take account of the user and their interaction in the system, therefore making it more difficult for a user to intuitively reason over such a model.

2.3.2 ARIS toolset

Architecture of Integrated Information Systems (ARIS) facilitates the description of an enterprise's underlying business processes. The components that make up an ARIS model include processes, activities, events, conditions and organisational units. To lessen the complexity of having to consider all the effects of every element on a process the ARIS model is divided into a number of individual views that represent different aspects of a process that can be for the most part modelled separately thus reducing the complexity.

These views are as follows:

- Events and data references or conditions make up the Data View;
- The functions to be executed and their relationships form the function view;
- The organisational view consists of the structure and the relationships between organisational units;
- The services or products that form inputs or outputs of functions are contained in the product/service view;
- To integrate these individual models into a model that allows for the description of the relationships between the various views the control view is used.

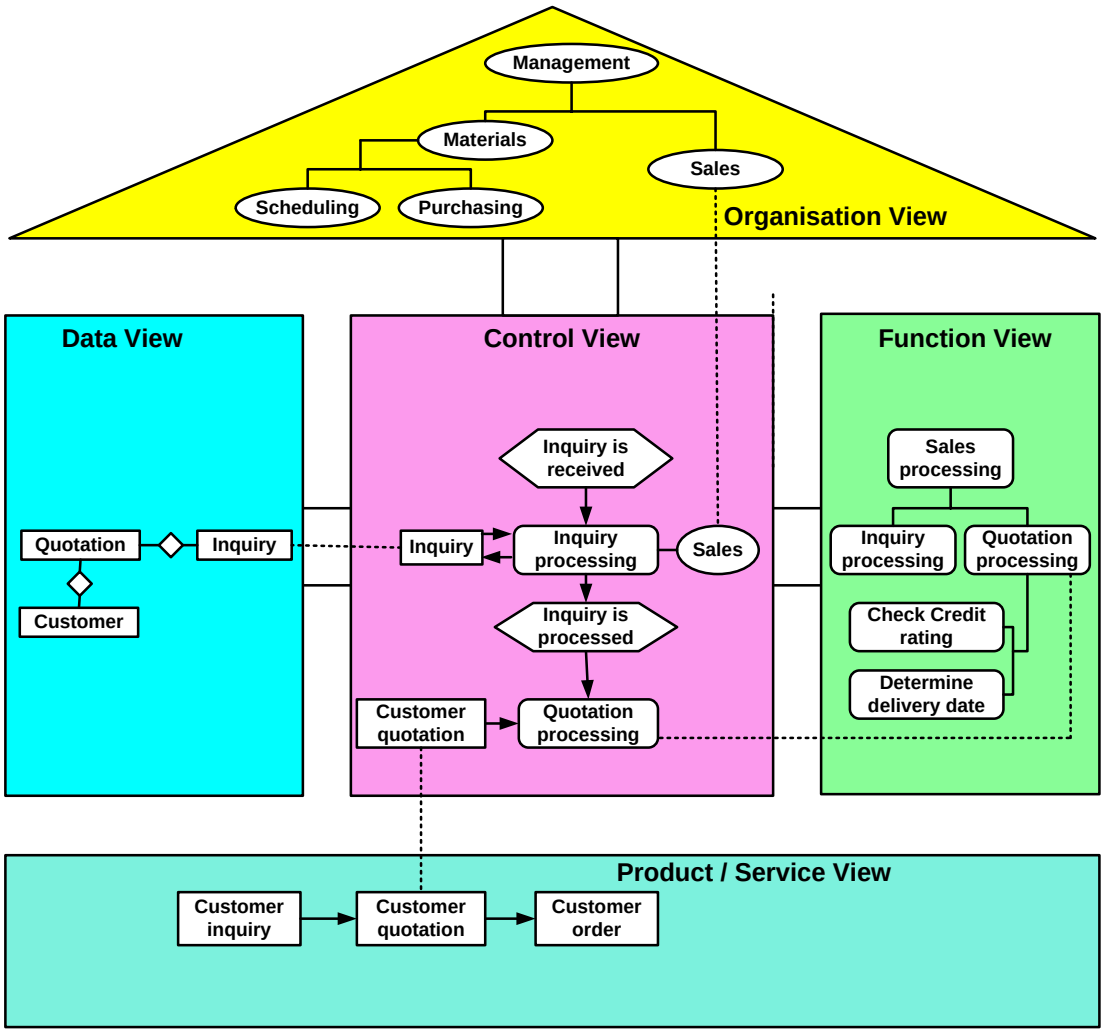

Figure 2.28 ARIS views

It is this control view that is the most essential ARIS component. This results in the ARIS views as shown in Figure 2.28.

ARIS Toolset is a product of the IDS Prof. Scheer GMBH [80]. This product is based on the ARIS concept and supports the user in modelling, analysing and navigating through business processes. Figure 2.29 shows the ARIS toolset user interface. This tool consolidates the various views into one business process as presented previously. It is the control view that records the relationships between these various views and thus consolidates the model. This is achieved by using the event-driven process chains (EPC) method that was introduced previously, an example of how the EPCs link the various views together is shown in Figure 2.30.

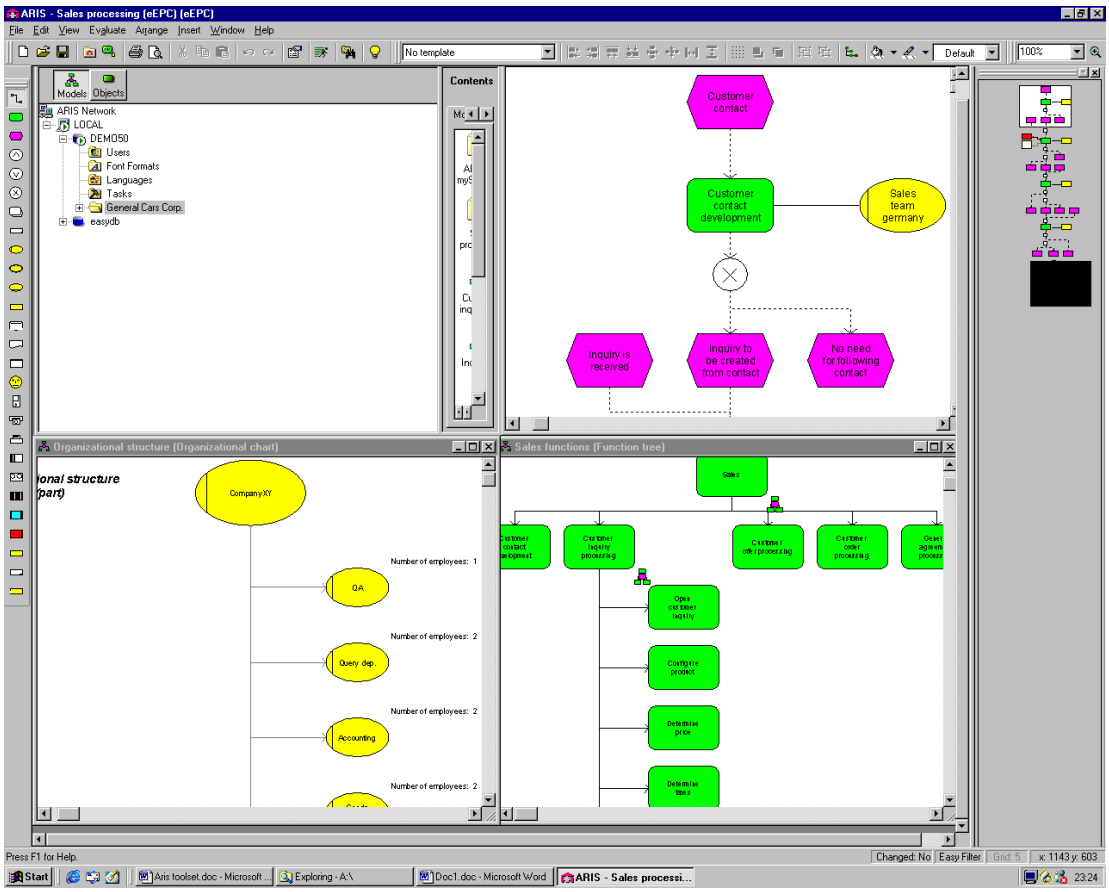

Figure 2.29 ARIS toolset user interface

This shows how the various views, namely data, function, organisation and product/service are brought together by the EPC method in the control view. This EPC method could be used to model routings and materials flows as in a discrete manufacturing process. Such a scenario is shown in Figure 2.31 as presented in [80].

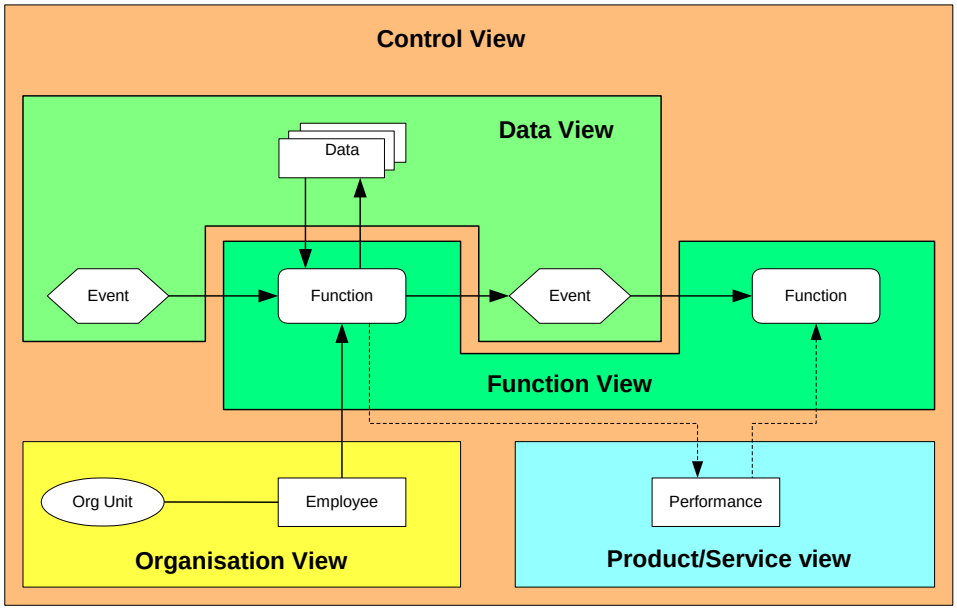

Figure 2.30 How the control view links the various views.

Having modelled a process in ARIS toolset a user can then evaluate and compare the process to others using ARIS analysis. Another tool offered to the user is ARIS simulation. With this a user can analyse bottlenecks and test alternative configurations within the modelled business process. The Event Driven Process Chains (EPC) technique of modelling on which ARIS toolset is based divides a process into a series of events and functions. Such a technique allows for the inclusion of the user within the model and their interaction therein. This technique can also be used to model the routings within a production facility as shown in Figure 2.31. However, this technique does not allow for the visual modelling of the detailed interactions of resources in the execution of each function within a process.

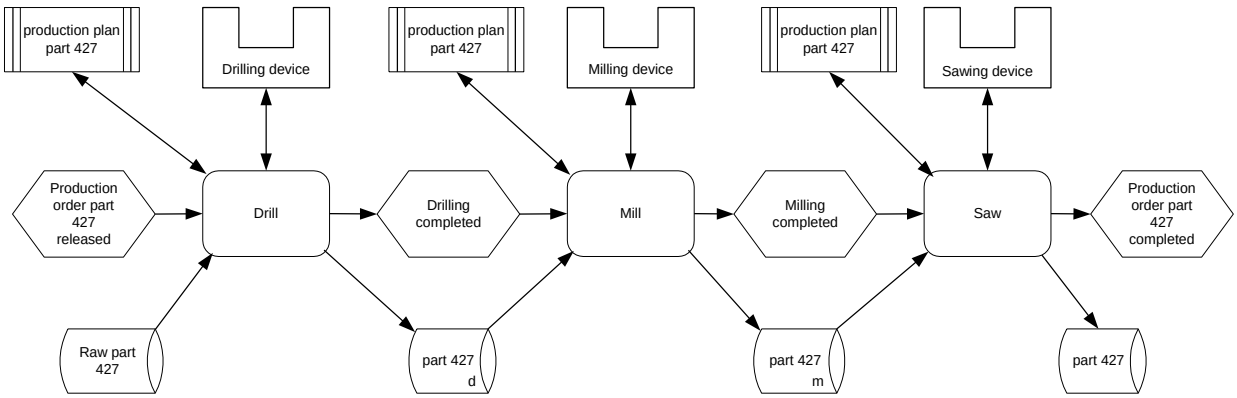

Figure 2.31 Routing and material flow as event-driven process chain.

2.3.3 ProSim

ProSim is KBSI's simulation design tool based on the IDEF3 process description capture method, introduced previously. In this method the focus is on the abstraction and capture of knowledge about a given real-world system. The tool, in a similar fashion to the capture method, focuses primarily on what fundamentally occurs in a system, the dynamic patterns among elements that repeatedly occur, as opposed to what happens at particular time instances in a system [81]. The tool divides a real world system into two types of scenarios or views, these being process flow diagrams and Object State Transition Networks (OSTN's).

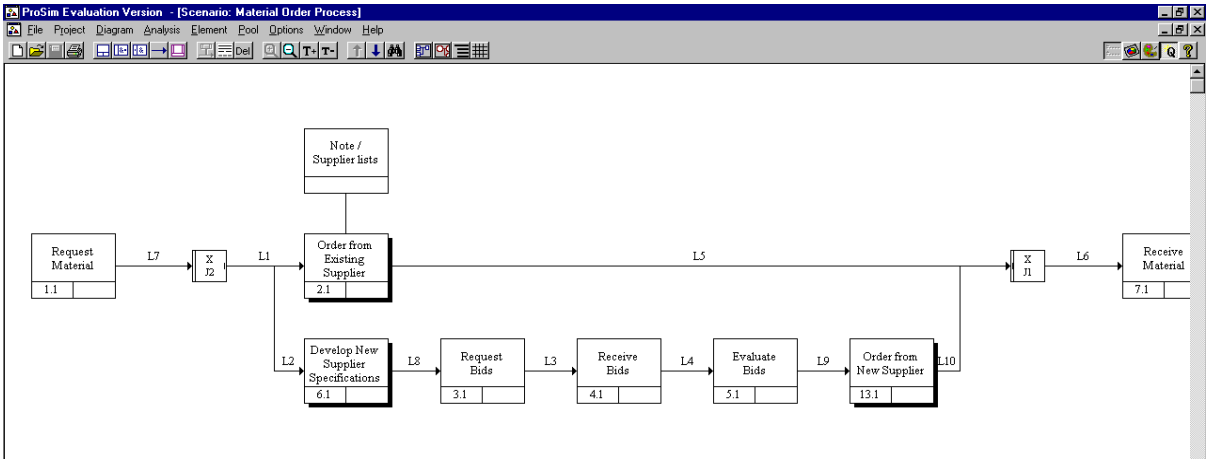

Figure 2.32 ProSim user interface showing UOBs

2.3.3.1 Process Flow Diagrams

In the process flow diagram scenarios of the ProSim tool, a user can develop a model or models of different perspectives of how a system operates. The main modelling element in this view is the IDEF3 Unit of Behaviour (UOB). This UOB element can be easily decomposed into a sub model, in other words a user can develop a more detailed model of any UOB if required.

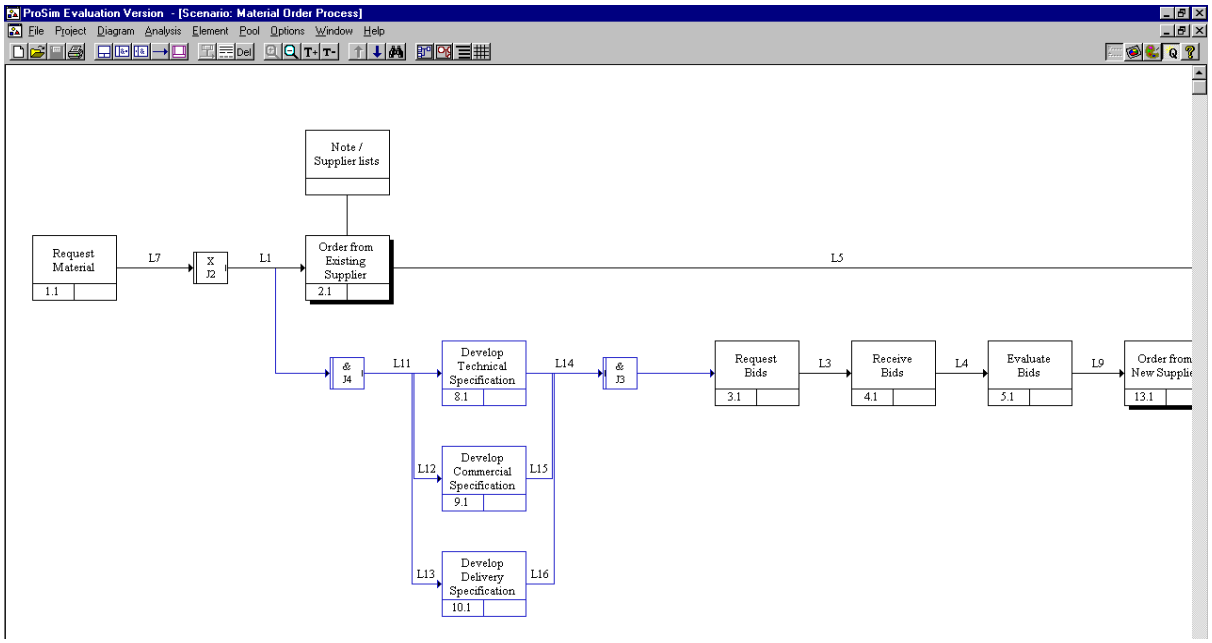

Figure 2.33 ProSim showing an expanded UOB.

Figure 2.32 shows the ProSim user interface with a Process flow diagram having a number of UOBs having sub models or decompositions. For example the UOB “Develop New Supplier Specifications” has a decomposition attached to it, this being denoted by a black shadow behind the UOBs containing decompositions.

Figure 2.33 shows the same process flow diagram but with the same UOB expanded and included in the main model, this sub model can also be viewed separately as in Figure 2.34.

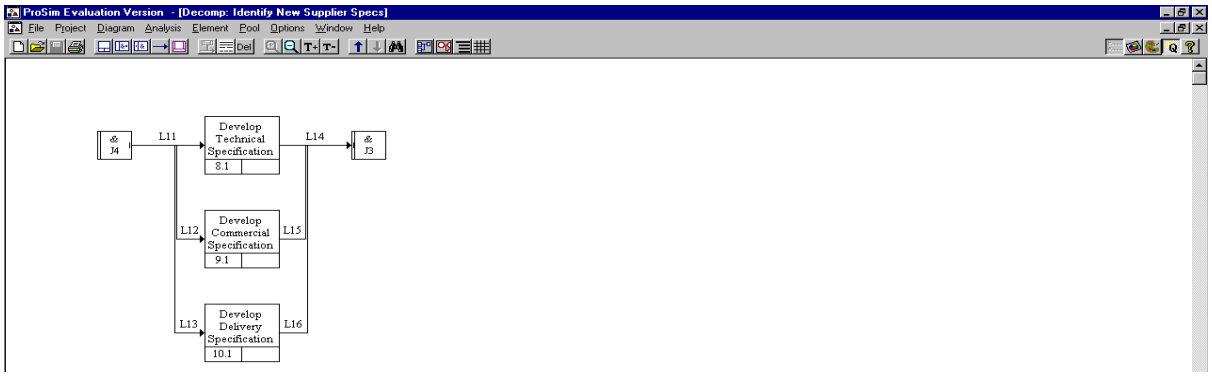

Figure 2.34 UOB sub-model.

Along with the UOB modelling element, the user also has access to the various junctions, links, elaboration and referents as in the IDEF3 capture method.

2.3.3.2 Object State Transition Networks

The second modelling view available to the user in ProSim it that of the Object State Transition Network (OSTN). In this object centred view, ProSim summarises the allowable transitions of an object through a system. The modelling elements used in this view are object states and transition arcs. Figure 2.35 shows this OSTN view as in the ProSim tool.

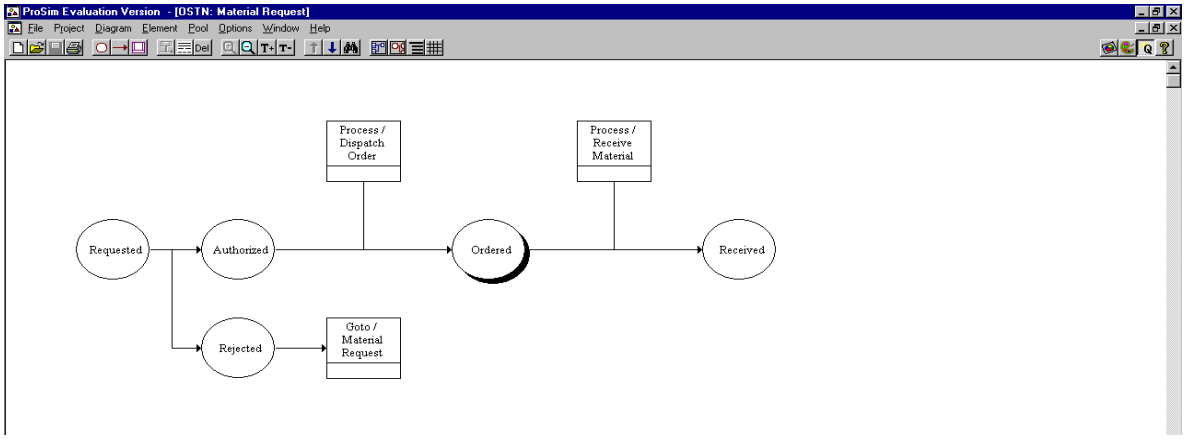

Figure 2.35 ProSim user interface showing the OSTN view

As in the process flow diagram scenario view any object state can be decomposed into a more detailed sub model. In this case the “Ordered” object state has a more detailed sub model, which is shown in Figure 2.36.

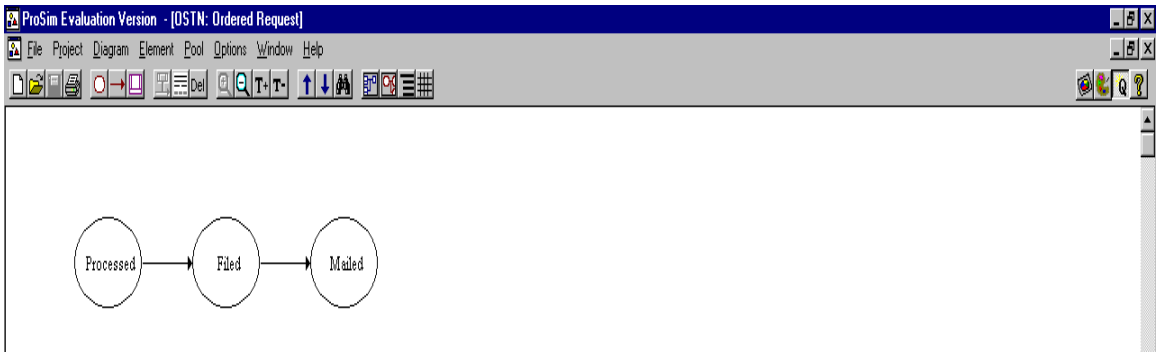

Figure 2.36 OSTN sub-model

Having developed an IDEF3 model in ProSim the user can automatically create a Witness simulation model. To achieve this the user has to add information to the model such as process times, resources and their usage and the objects that are to be processed in the model in order to generate the Witness simulation model. Therefore ProSim and the IDEF3 modelling technique can be used to model discrete event systems, these process models can also be used to for the automatic generation of a Witness simulation model. However Robinson 2004 [18] defines a conceptual model as being a non proprietary simulation software specific description of a simulation model. This it is argued should lead to the selection of the simulation software based on the understanding of the conceptual model rather than the case of tying a model developer to a particular simulation tool during the early phases of a simulation project. This is not the case with ProSim and Witness being explicitly linked.

2.4 Conclusions

This chapter presented a review of a number of different process modelling techniques and tools for the purposes of aiding the requirements gathering, conceptual modelling phases of a simulation project. This review was carried out with a view to ascertain each techniques ability to capture, represent and communicate the various aspects of a complex discrete event system. While certain techniques such as Petri nets and the DEVS formalism were capable of capturing and accurately representing complex discrete event system information, the techniques did not lend themselves to acting as a means of

communicating such complex information. ACDs while capable of modelling the workflow aspect of a discrete event system in terms of cycles of activity and waiting, does not allow for the capture of detailed system logic. The technique also does not allow for the hierarchical decomposition of a model and can become cumbersome when modelling a complex system. A UML activity diagram technique is capable of representing the execution of a discrete event system as a sequence of activities. The UML statechart diagram technique allows for the modelling of the flow of control or sequences of states that a system can proceed through as a result of discrete events. However, the techniques lack the ability to graphically capture and communicate the interactions between resources and the flow of work or information, which are felt necessary to facilitate the communication of all detailed simulation logic to a non-simulation expert. RADs do not model the change of state of a discrete event system. Instead they model a process in terms of roles that have to be carried out within that process. This approach of placing the role of an individual to the fore in terms of their interaction with a discrete event system aids reasoning and communication of system issues. However, the lack of modelling of logical sequences of discrete system states means that the technique does not lend itself readily to the capture and communication of detailed discrete event system information. The GRAI method focuses primarily on modelling the decisional structure of a system. The approach does not model the physical aspects of a discrete event system adequately to facilitate the capture and communication of complex discrete event system issues. However, modelling the decisional structure of a discrete event system is important as modern discrete event systems depend on such decisional systems in their operation. Therefore, the capture, representation and communication of such decisional aspects of a discrete event system are important to aid a model developer in communicating issues associated with decisional aspects of a discrete event system to system personnel. The IEM technique allows for the graphical representation of both control and resource elements in the execution of activities associated with the execution of a discrete event system. However the technique lacks the modelling constructs which

would allow for the capture, graphical representation and communication of all aspects of a complex discrete event system. This technique is implemented in the MOOGO process modelling tool. While the EPC technique, which is used as part of the ARIS toolset software, is capable of accurately representing certain areas of complex discrete event systems, such as the flow of activities associated with the execution of tasks, it lacks the ability to capture and represent all of the aspects that would allow it to function as a communicative and representative technique during the requirements gathering phases of a simulation project. In a similar way the IDEF0 technique allows for the visual modelling of the decisions and activities in a discrete event system, but, lacks the constructs to model the various other aspects of a complex discrete event system, such as the flow of work and control. Therefore, while the IDEF0 technique is capable of capturing, representing and communicating certain aspects of a discrete event system, it lacks the ability to capture, represent and communicate all such aspects of such systems. The final technique reviewed was the IDEF3 process modelling technique. This technique allows for the capture, representation and communication of the various states through which a discrete event system can transition and the activities associated with them. However, the information associated with the control of such systems and the use of resources are not graphically represented within the technique. Therefore, while the IDEF3 technique is again capable of capturing, representing and communicating certain aspects of a complex discrete event system it lacks the ability to represent all aspects of such a system. The ProSim process modelling software allows for the development of IDEF3 process models, and the generation of Witness simulation models from such IDEF3 models.

A number of conclusions can be drawn from the material discussed in this chapter. First, there are a number of process modelling techniques and software tools available that may be used to support the requirements gathering phases of a simulation project. Secondly while these techniques can be used to model discrete event processes, the conclusion drawn is that none of the techniques currently available are capable of capturing, representing and communicating the

various aspects of discrete event systems and their interactions within a complex process in such a way as to aid in the visualisation and communication of detailed simulation information to a non-simulation expert. The shortfalls that need to be addressed in relation to these issues were outlined in the chapter 1

In an attempt to address these various shortfalls, the following chapters outline a process modelling technique, Simulation Activity Diagrams SADs, and in turn a process modelling tool, Process Modelling for Simulation (PMS), based on this technique. Both have been developed to specifically support the pre-coding phases of a simulation project. With a view to overcoming the shortfalls outlined above and in doing so it is argued that SADs and PMS are well placed to support a model developer in the requirements gathering phases and conceptual model development within the process of simulation project.

Chapter 3 SAD Development Process

The SAD development process initially involved a detailed review of process modelling techniques developed and used to support the requirements gathering/conceptual modelling phases of a simulation project. This initial review highlighted the lack of research in this area. No techniques specifically developed to support these pre-coding phases of a simulation project were found. Noted authors in the field of simulation modelling such as Law and Kelton [82], give little more than a cursory introduction to the field. Robinson [18] also highlights the lack of research in this area. This lack of research points to what may be viewed as a traditional narrow focus on simulation modelling support that fails to account for the broader modelling considerations as highlighted by a number of authors [83], [84]. As a result of this gap in the literature in relation to this specific area, the focus of the literature review changed scope to a broader review of process modelling techniques that it was felt were capable of modelling a discrete event system. By taking such a broad approach to the literature review it became apparent that there were many process modelling techniques available, which were broadly capable of satisfying some of the required criteria. Kettinger et al. [17] quoted more than one hundred in a study that was not exhaustive. As a result it was deemed impractical to attempt to review every such technique. The focus of the literature review was then narrowed to process modelling techniques capable of or deemed to be suited to supporting the pre-coding phases of a simulation project even if such techniques had not been specifically developed for such a purpose. Again many techniques were examined which were proposed as being capable of modelling a discrete event system for the purposes of among others simulation. However due to their extremely broad scope and all encompassing nature a number of these techniques such as, the Process

Specification Language (PSL) [85], Computer Integrated Manufacturing Open System Architecture (CIMOSA) [86], Toronto Virtual Enterprise Ontology (TOVE) [87] and the Purdue Enterprise Reference Architecture (PERA) [88] were deemed to be unsuitable to the specific nature of the problem area being examined. However a number of techniques were identified that were seen to be broadly focused on the problem area in question and also capable of somewhat representing complex discrete event logic. It was these techniques and a number of supporting tools that were presented and discussed within the literature review in Chapter 2. The literature review specifically focused on each of these techniques ability to satisfy the requirements listed in Chapter 1. Figure 3.1 below gives a summary of each technique reviewed under the specific categories listed in the requirements. The grading under which each technique is listed is as follows:

Technique	Good Communication / Visualisation medium	User Perspective	State flow modelling	Information flow modelling	Resource modelling	Activity Modelling	Complex branching logic	Decomposition	Elaboration Language
Petri Nets	Medium	Low	High	Low	Low	High	Low	Low	Low
ACDs	Medium	Low	High	Low	Low	High	Low	Low	Low
DEVS	Low	Low	High	Low	Medium	Medium	Medium	Low	Low
UML Activity Diagrams	High	Low	Low	Low	Low	High	Medium	Low	Low
UML Statecharts	High	Low	High	Low	Low	Low	Medium	High	Low
RADs	High	High	Low	Low	Low	High	Medium	Low	Low
GRAI	Medium	Low	Medium	High	Medium	High	Low	Medium	Low
IEM	High	Low	High	High	Medium	High	High	High	Low
EDPCs	High	Low	Low	Medium	Low	High	High	Low	Low
IDEF0	High	Low	Low	Medium	Medium	High	Low	High	Low
IDEF3	High	Low	High	Low	Medium	High	High	High	Medium

Figure 3.1 Requirements satisfaction attributed to reviewed techniques

- **High (H)** Highlights that the technique was very capable of fulfilling this requirement;
- **Medium (M)** Highlights that the technique was somewhat capable of fulfilling this requirement;
- **Low (L)** Highlights that the technique was not capable of fulfilling this requirement.

Taking each of the techniques listed in Figure 3.1 the following sections will outline the categorizations arrived at in more detail. Petri nets are to a certain extent capable of visually representing and communicating discrete event system logic, however such Petri net models are not capable of visually accounting for complex branching logic or hierarchically decomposing complex models into sub models and as a result become very cumbersome as system complexity increases. The technique also does not account for a users viewpoint, resources, information flows or a means of elaborating the graphical model in a textual manner. However the technique is capable of accurately representing state flows and the activities associated with the execution of such flows.

ACDs are again somewhat capable of visually representing and communicating certain discrete event system logic. It achieves this by means of modelling state flows and the activities that cause such state flows to be executed. However the technique fails to account for a users perspective, resources, information modelling, branching logic or a means of textually elaborating graphical models.

The DEVS formalism is capable of accurately representing the various changes in state of a discrete event system along with being somewhat capable of representing resources, activities and branching within its mathematical representation. However the formalism is not visual in nature and does not account for the users interactions with the system, information flows or a user friendly elaboration language.

UML activity diagrams are designed to represent a discrete event system as a series of activities linked together to show the various phases of activity within a

discrete event system. The technique is highly visual and communicative and also has to a certain extent a means of visually representing the logical flow of activities. However the system does not account for the users' perspective, state flows, information modelling, resource modelling or a means of elaborating the graphical models.

UML statecharts are a highly visual and communicative modelling technique that are used represent a discrete event system as a series of interrelated state flows. This technique also has a means of graphically representing the logical flow of states and hierarchically decomposing a model into sub models. However the system does not account for information flows, resources, activities, and an inclusion of a users interaction with the system or a means of textually elaborating the graphical model.

RADs are a highly visual modelling technique that accounts for the user perspective in the development of a process model of a discrete event system. The technique is to a certain extent also capable of representing the logical branching of such activities within a model. The technique however does not have the means of representing state flows, information flows, resource interactions or a means of either hierarchically decomposing or textually elaborating graphical models.

The GRAI model offers a means of modelling the detailed information and control interactions within a discrete event system. This information model is also capable of representing discrete activities and model decomposition along with to a lesser extent both state flows and resources. However the model does not account explicitly for the users perspective, branching logic or an elaboration language.

The IEM technique presents a highly visual and communicative model of a discrete event system, which is capable of graphically representing state flows, information and resource elements. The technique is also capable of hierarchically decomposing a model into sub models along with having a detailed

branching logic associated with it. However the technique does not account for a users viewpoint or have an associated elaboration language.

EDPCs are a highly graphical process modelling technique which are capable of representing a discrete event system as a series of activities. The technique is capable of representing branching logic and to a lesser extent information interactions within the system. Drawbacks of the system however include its lack of a representation of the users perspective, state flows, and resource interactions. The technique also does not have the capability to hierarchically decompose a model into sub models or have access to an associated elaboration language.

IDEF0 is a graphical modelling technique capable of representing a discrete event system as a series of interrelated activities. The technique is capable of hierarchically decomposing a model into sub models and is also to a certain extent capable of accounting for both information and resource interactions. However the technique does not account for system branching, the elaboration of graphical models, state flows or the modelling of a users perspective.

The IDEF3 process modelling technique is capable of graphically representing the various states through which a discrete event system can transition along with the various activities associated with each change of state. This technique also offers a means of representing complex system branching logic along with a means of hierarchically decomposing a model into related sub models. The technique is also capable of textually representing the graphical models, however this representation language is abstract in nature. This representation language also offers a means of representing resources associated with the graphical models. However the technique does not account for information flows or modelling from a users perspective.

Taking a view of the various themes that it is felt are necessary to address, as listed in columns in Figure 3.1, in creating a technique capable of fulfilling the requirements developed in Chapter 1 the following issues are apparent. In relation to being a good communication and visualization medium many

techniques are very good at presenting the various aspects, which they model in a highly visual and communicative manner. However in addressing the user perspective many techniques fail to appropriately address this issue. Only the RAD technique takes the view of modelling a process in terms of the role or person charged with their execution. The IEM technique also addresses this issue to a lesser extent.

The techniques that model state flows well include ACDs, DEVS, UML statecharts, IEM and IDEF3. The GRAI technique does to a lesser extent allow for the modelling of state flow, however this technique is primarily focused on the modelling of information flows. Techniques such as UML activity diagrams, RADs, EDPCs and IDEF0 are not concerned with the modelling of state flows. In terms of the modelling of information flows most techniques are capable of representing certain aspects of an information system however only the GRAI technique is capable of accurately representing the information interactions within a discrete system.

Resources are a major issue in many simulation projects. Techniques such as IEM and EDPCs are capable of accurately representing such resources within a discrete event system. To a lesser extent IDEF0, IDEF3, GRAI, RADs and DEVS can represent aspects of resources within a discrete event system. However techniques such as Petri Nest, ACDs, UML activity diagrams and UML statecharts do not have such a means of representing such resources. Activities are also well represented within many techniques such as Petri nets, ACDs, UML activity diagrams, RADs, GRAI, IEM, EDPCs, IDEF0 and IDEF3. While the DEVS technique is capable of representing activities to a lesser extent. Certain techniques such as UML statecharts are not designed to represent such activities.

Complex branching logic is well represented with techniques such as UML activity diagrams, UML statecharts, EDPCs and IDEF3 by means of the branch types used in each. Techniques such as Petri Nets, DEVS, RADs and IEM have the ability to represent such branching to a lesser extent. While techniques such

as IDEF0, GRAI and ACDs lack the capability to display such branching logic. Finally no technique examined apart from the IDEF3 technique was capable of presenting the user with an elaboration language to further explain the graphical model produced. While the IDEF3 technique did have this capability the elaboration language was abstract in nature and not easy to reason over.

As is shown in the sections above the literature review concluded that no technique examined was adequately equipped to fully support the requirements outlined in Chapter 1. As a result the development of the Simulation Activity Diagrams (SADs) was undertaken. The initial development process focused primarily on the state or entity flows through a discrete event system. This was primarily examined as the majority of process modelling techniques concentrated on representing this element of a discrete event system. An initial draft of a high level SAD diagram is shown in Figure 3.2

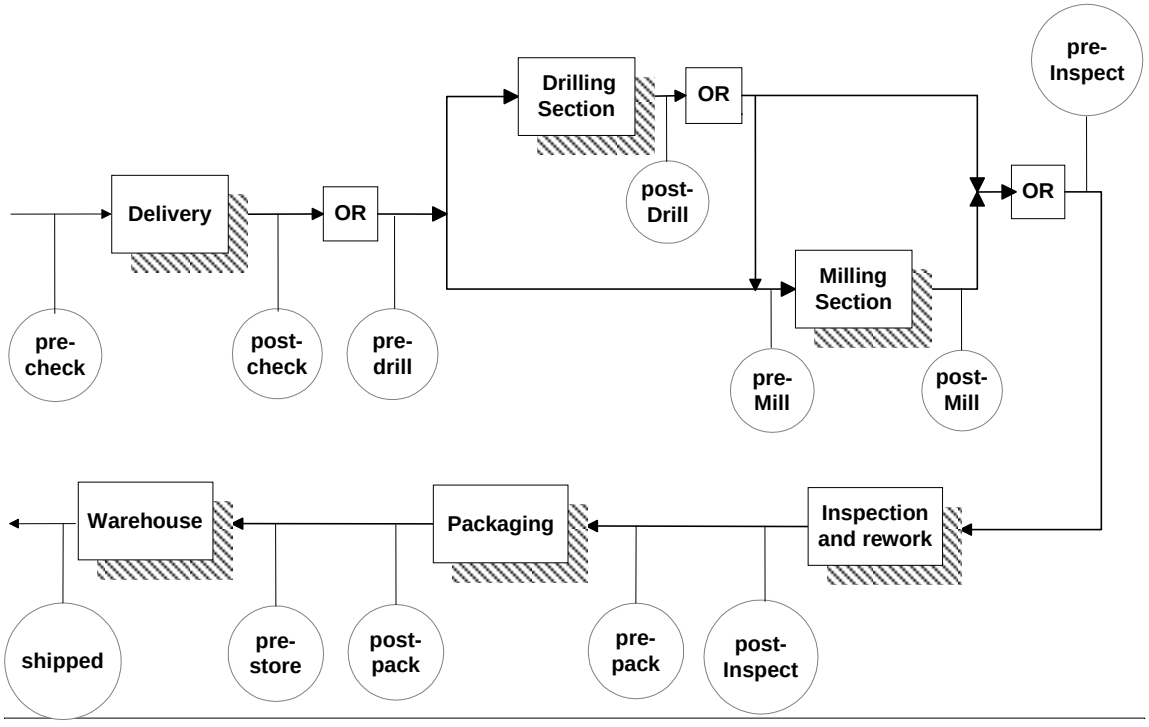

Figure 3.2 High level initial SAD diagram

This high level model shows the flow of entities through a precision component manufacturing facility. The lower level SAD model associated with the delivery area shown in the high level view, Figure 3.2, is shown in Figure 3.3 below.

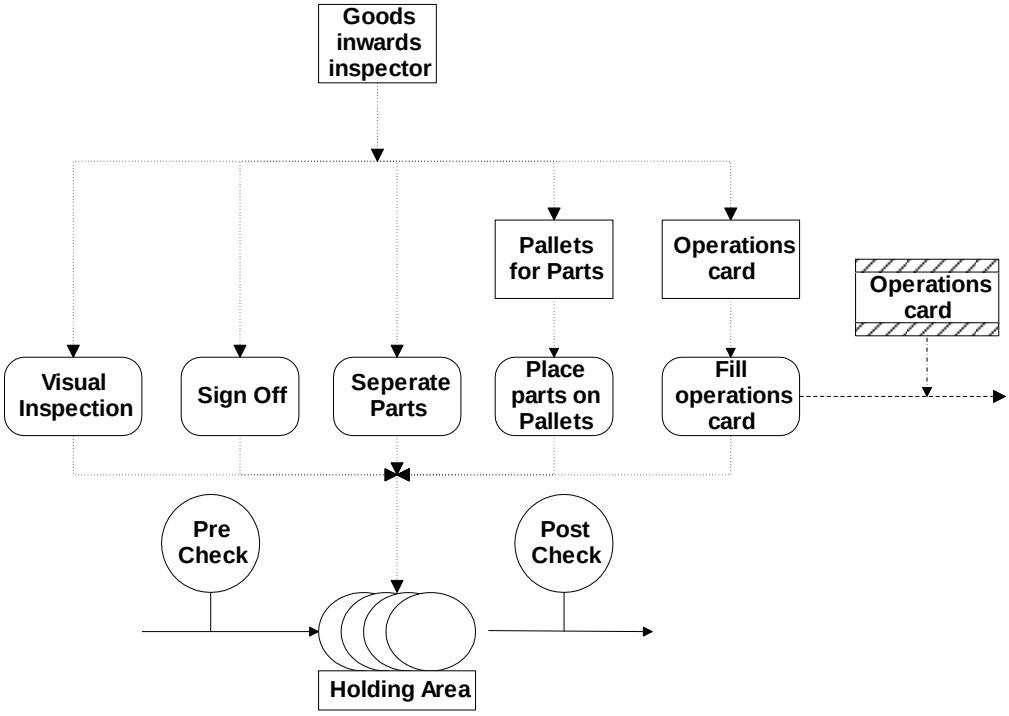

Figure 3.3 Initial SAD Draft of Delivery area.

The initial draft SAD technique was discussed with a number of members of the then Enterprise Engineering Research centre to ascertain the techniques ability to model issues of note and importance to a simulation project in a manner that facilitated communication and understanding while also promoting accurate model development. This testing was carried out by a qualified panel as shown above who may have had what might be termed an expert bias taking a less qualified audience may yield further insights and needs for functionality that may not be highlighted by the expert review group. The members involved at this review stage were as follows:

- **Mr. John Geraghty** (MEng, PhD Candidate) Industrial simulation model developer and Simulation /Operations Research researcher with particular interest in buffer allocation issues [89], [90], [91];

- **Dr. James Crawford** (PhD in Anti-thetic variates) Industrial simulation model developer and Simulation /Operations Research researcher [92];
- **Mr. S.M. Shahab Khanian** (MEng Candidate) Industrial simulation model developer and Simulation /Operations Research researcher with particular interest in complex serial automated production line simulation [93], [94];
- **Mr. Pat McNally** (MEng Candidate) Industrial simulation model developer and Simulation /Operations Research researcher with particular interest in the separation between users and developers in complex simulation models [95], [96];
- **Mr. S.M. Shafi** Khanian (MEng Candidate) Industrial simulation model developer and Simulation /Operations Research researcher with particular interest on the development of specialized process model simulators) [97], [98].

Preliminary discussions with these experts in the field highlighted a number of weaknesses in the initial draft of the SAD technique. Firstly the activity/action flows from the operator to the various actions and onwards to the primary resources were ambiguous. For instance in Figure 3.3 it is not immediately apparent if the goods inwards inspector has to execute all, none or some of the actions shown for the successful transition of an entity state from a pre-check to a post-check state. Also highlighted by this initial review was the important nature of information in a complex discrete event system. The experts commented that modern manufacturing systems relied heavily on information for control and often this formed a vital element of a simulation model. As a result it was deemed to be important to represent such information within any modelling technique developed for the support of such projects. On completion of this initial review the literature review introduced in Chapter 2 was again revisited with a view to addressing each of the issues highlighted. To address theses issues a number of approaches were undertaken, for instance the use of the branching logic used in the original draft SAD technique only in the state transitions was now adopted for

use in the activity flows, thus eliminating the ambiguous situations regarding the execution of actions. The information flow was also initially accounted for by introducing the information flow link in a SAD model on the same level as an entity flow as shown in Figure 3.4.

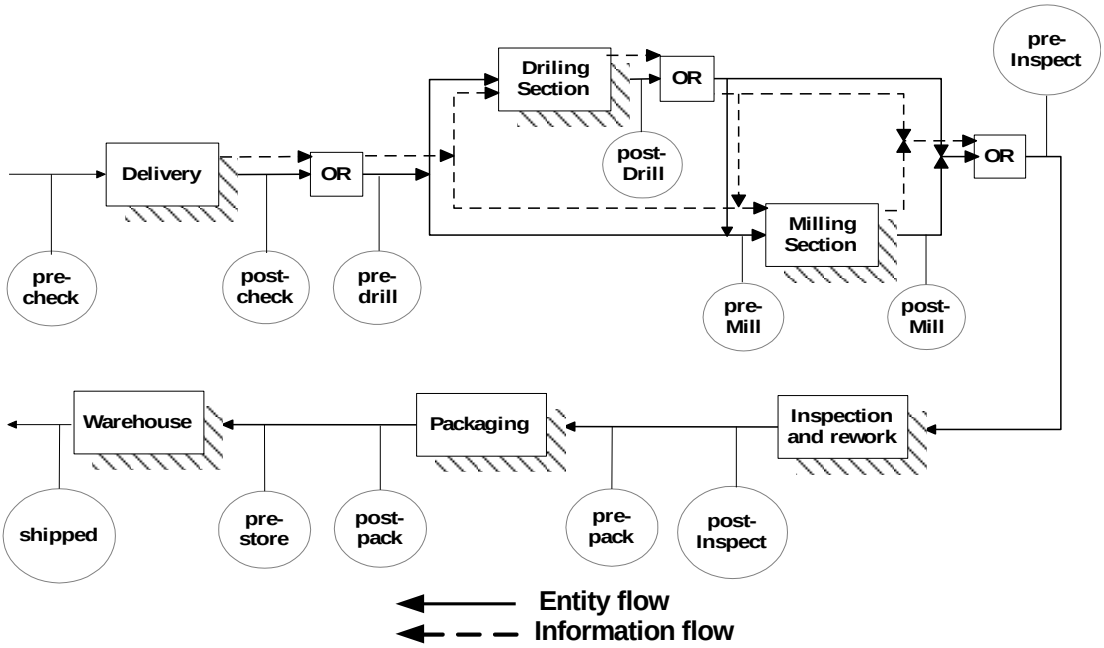

Figure 3.4 High level SAD Draft model showing information flows

In this Figure the information flows are shown as a hatched line. However again on discussion with the expert simulation panel it was found that this situation was ambiguous and difficult to reason with. From one of these discussions it emerged that the representation of the information model as a separate ‘sub model’ within the over SAD model would be advantageous. However it was also felt that both models should be capable of a certain level of interaction as was the case in actual discrete event systems. In relation to the continuing development of the SAD technique an iterative approach of discussion was undertaken with colleagues within the enterprise engineering research centre in developing a solution that met the initial requirements set down and also was felt capable of best modelling the various aspects of a discrete event system in a manner that was capable of visually communicating such aspects to non simulation experts. It is the outcome of this iterative development process that is introduced in the next chapter.

Chapter 4: Simulation Activity Diagrams

4.1 Introduction

Chapter 1 outlined the lack of a dedicated technique developed to overcome problems associated with the requirements gathering phases of a simulation project and also discussed a proposed solution to this shortfall. The literature review in the previous chapter presented the most appropriate techniques currently available to aid a model developer in this requirements gathering phase. No technique presented fulfilled all the requirements outlined in chapter 1, of being capable of capturing detailed information required for simulation modelling in a manner that is highly visual, communicative and user friendly. The technique outlined in this chapter aims to overcome these shortfalls and in so doing, support a simulation model developer in the capture, representation and communication of information during the requirements gathering phases of a simulation project. The technique aims to be highly visual and aid in the process of communication between the model developer and system users, while still aiding the model developer in gathering data for the creation of a simulation model. Figure 4.1 shows the current difficulties that can be associated with simulation models and their drawbacks as a communicative tool. The usage of the proposed modelling tool outlined in this chapter in overcoming such shortfalls is shown in Figure 4.2

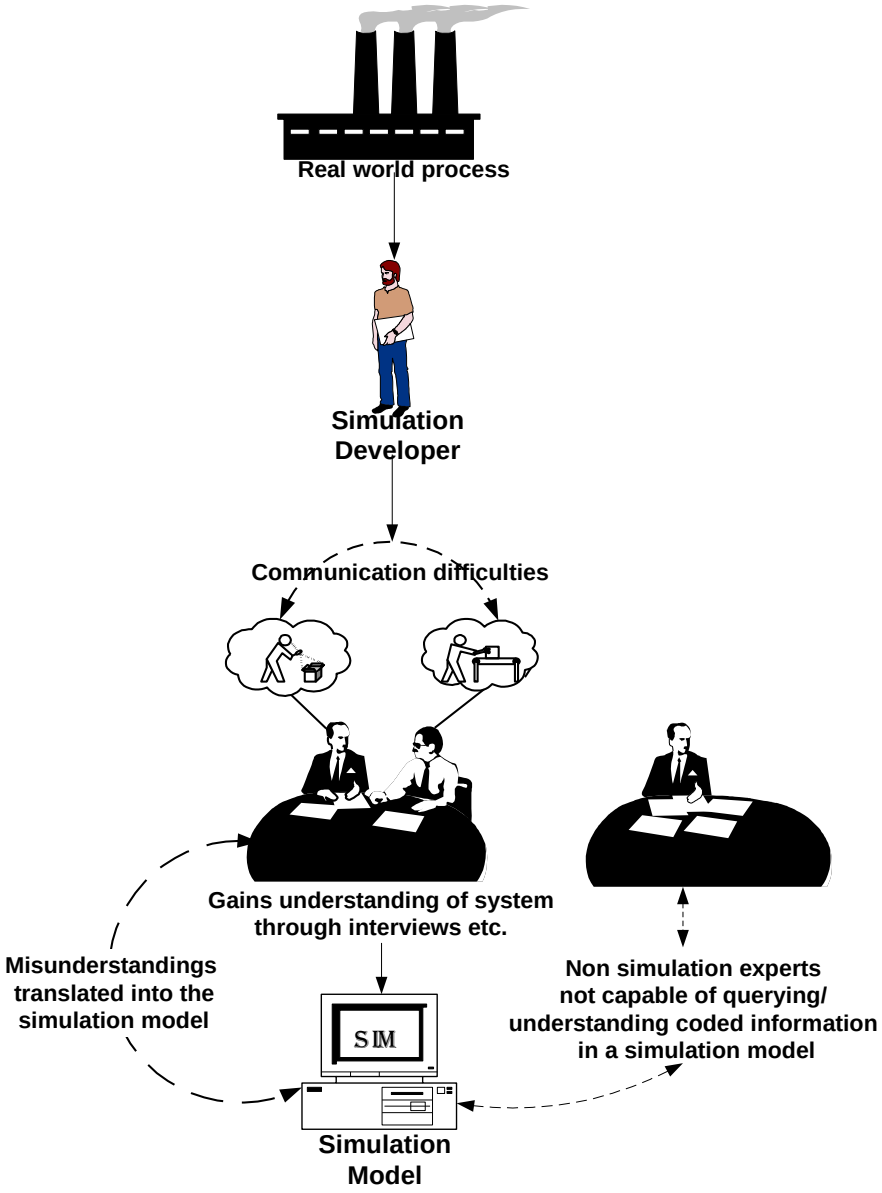

Figure 4.1 Difficulties with simulation models as a communicative tool

This chapter in providing an overview of the proposed modelling technique considers the various modelling components and how they fit together to form a complete view of the system being modelled. Since a simulation project generally deals with highly complex issues and large amounts of detailed data the modelling technique aims to present such information in a user friendly and highly visual manner.

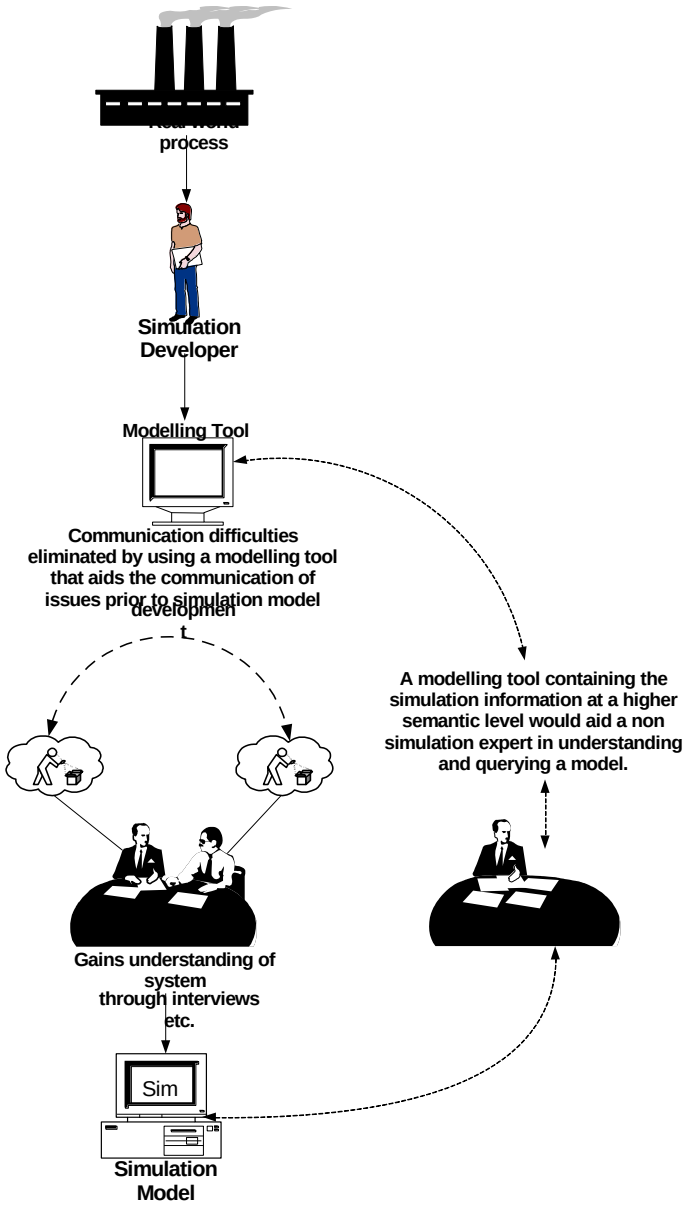

Figure 4.2 Proposed use of the SAD technique

4.2 Design Objectives

In developing the process modelling technique outlined in this chapter there were a number of specific requirements to be adhered to as outlined by the requirements listed on page 7, chapter 1.

In addressing these design objectives the technique developed uses a set of modelling elements that allow both a simulation model developer and a non

expert to reach a common understanding of the system being modelled. The technique allows the construction of a detailed and highly visual model of a system. This model can then be used as a common representation and a focal point for discussion with which, both manufacturing personnel and the developer can reach a common understanding of the operation of the system and the data requirements. The technique in this way allows the user high level access to the knowledge contained in simulation code that would otherwise be lost due to its internal programming details. The types of information that a SAD diagram is capable of capturing include, the sequence of execution of physical tasks, the sequence of execution of information or control, the decision making of and interaction between a user, primary and auxiliary resources, the physical routings for parts, the routing for information, and the various stages of transition for both parts and information. Central to all of this information will be the user, or that which initiates an activity at a given time instance. The technique places the user as the central focus from where all interactions are driven. A user can construct a high level model of a system and hierarchically decompose detailed processes into sub-processes, to aid in the rationalising over complex processes. In this way users at different levels of a system, such as a manufacturing system, are able to access data that affects them. This technique also allows the user to avoid having to reason over complex information that does not affect them. For example, a model of a manufacturing line may possibly be divided into two levels. At the first level the model may represent the flows of parts through the system, the buffer sizes and the processes through which a part has to pass before exiting the line. At a lower level the model may represent the detailed operations of each station on the line. Such a level may contain the different activities carried out at each station and the resources involved. At the higher level the model may be of use to a plant supervisor who may not be interested in the detailed operations of the system but rather the overall view of the operation. However, the lower level would be of interest to the people involved in the operations level within the line.

4.3 Simulation Activity Diagram Modelling Primitives

A model of a discrete event system consists of a series of discrete events. At these points in time, events take place that decide the progress of the system under examination. When modelling such a scenario a particular discrete event simulation model may indeed group a number of different such events together to lessen model complexity. The SAD modelling technique has been developed to provide a mechanism for the graphical representation of the grouping of such events in a highly visual and user friendly manner. The modelling primitives that are used to underpin the SAD modelling technique are introduced in the following sections, along with how to formulate these same primitives into a SAD graphical representation.

4.3.1 Timing of the events in a discrete simulation model

The nature of a discrete event system is such that the state of the system changes only as a result of the execution of an event or events at a particular instance in time that cause a change. Within a discrete event simulation engine there are two mechanisms which are used to keep track of such events. Firstly a variable known as the simulation clock is used to record the current simulation time. Secondly, to keep track of events the simulation engine maintains a list of all pending events. This list is known as an event list and its task is to maintain all pending events in chronological order, that is, events are inserted into it ordered by their time of occurrence. In particular, the most imminent event is always located at the head of the event list.

In the execution of a simulation model the simulation clock is set to zero and the initial events are loaded into the event list in chronological order. Next, the most imminent event is unloaded from the event list for execution, and the simulation clock is advanced to its occurrence time. In the course of executing the current event, the state of the system is updated, and future events are typically generated and loaded into the event list. The process modelling techniques presented in Chapter 2 fail to adequately communicate such discrete event logic

of systems in a highly visual and user-friendly manner and it is in addressing this shortfall that the Simulation Activity Diagram (SAD) technique has been developed. SAD process models provide a mechanism for graphically visualising and communicating detailed information such as that contained within a simulation model. The sequence of execution of a SAD model is from left to right and where it is applicable from the centre of the model to the extremity, as shown in Figure 4.3.

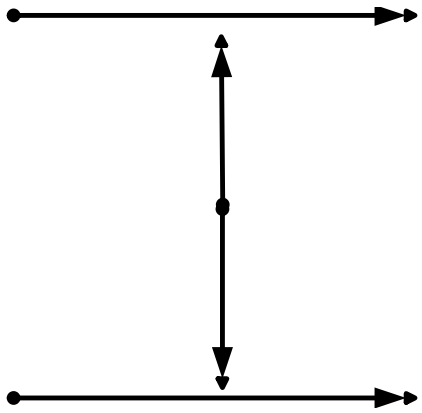

Figure 4.3 The direction of execution of events within a SAD diagram

Therefore, to graphically represent information similar to that contained within a simulation event list, the sequence of execution of a SAD model is timed as follows. A model is always initiated from the left hand side of model, by the graphical representation of the entrance event or events. The sequence of execution then proceeds to the centre of the graphical model where elements necessary for the execution of events are graphically represented. From here the sequence of execution of events proceed from the central area to the extremities of the graphical model with the SAD elements proceeding to an exit condition at the far right hand side of a graphical representation.

4.3.2 SAD Model structure

The SAD technique graphically represents every event in a simulation event list by means of an activity.

"An activity is any event that causes the change of state of a discrete event system"

In Figure 4.4 a simple discrete event system changes from state 1 to state 2 as a result of an activity, A.

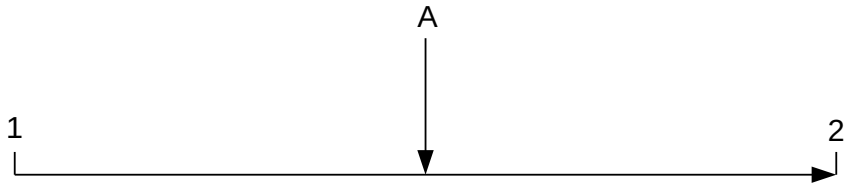

Figure 4.4 A change of state of a simple discrete event system

However, as mentioned previously, an event in a simulation event list can often represent more than one task or action carried out within a real system. Often, model developers group such events together to lessen the programming burden. For example, a simulation model developer may group the overall actions of picking a part, preparing, loading, machining and unloading of a part into one event. This can often lead to difficulties in relation to non-simulation personnel understanding simulation models. To overcome this, an activity can be subdivided into a series of what are defined as actions.

"An action element represents the individual task or tasks that have to be performed within a system at a particular instance"

This approach allows an activity or event to be further subdivided into its various individual elements or tasks. In other words an activity in a SAD model can be considered to be a list of actions that have to be executed in order for the activity to be fully completed. Therefore returning to Figure 4.4 activity A can be considered a separate list of actions that have to be executed in order for the system to transition from state 1 to state 2. These actions are graphically represented within a SAD model and are executed from the inner section of the model to the extremities and from left to right within each model. If an activity consists of three actions, Figure 4.5, each action is executed as follows.

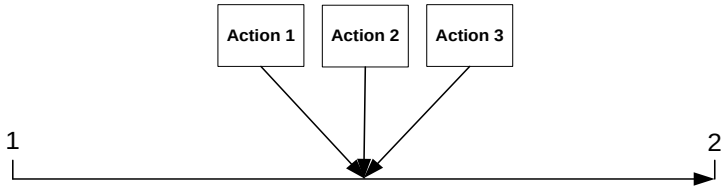

Figure 4.5 SAD Actions

The system is in state 1, before it can transition to state 2 all actions, 1, 2 and 3 must be executed. In this way an individual activity can be considered a separate mini event list or action list within the SAD model.

> ***"An action list consists of the time phased sequence of the individual actions that make up a particular activity at a particular time"***

These actions are executed from the centre of the model to the extremities and from left to right ensuring that each criterion is satisfied. Only when each action has been executed, can the full activity be executed and the system transition successfully to state 2.

Returning to Figure 4.5, such a scenario could be used to represent a simple simulation model mimicking a simple system. The system modelled may be as follows: an entity in state 1 has to be processed before transitioning to state 2. To represent this the simulation model would release an entity from state 1 after which it would take the entity and hold it for a period of time. This period of holding represents the time taken to execute actions 1, 2 and 3. In other words taking and holding the entity for the period of time taken to execute activity A from Figure 4.4. The completion of this hold period, or activity A, allows the transitioning of the entity to state 2. In terms of the simulation model this may be represented by the freeing of the entity whereby it may exit the system or move onto further stages of waiting or processing.

To most persons involved in the day to day running of such systems, be they simple or complex, the use of such terms as taking, holding and freeing and the process of directly relating such terms to their particular system is often too abstract and time consuming to be useful. Therefore to aid a model user in reasoning with such information and terms, the SAD technique further develops

the concept of an activity being composed of a series of interrelated actions, known as an action list. Taking this approach a SAD can become a graphical representation of the various events in a simulation model.

Within most systems, actions such as those in Figure 4.5 are rarely executed without other resources being used. For example the three actions that make up activity A will generally have to take place at a fixed location. Such a location may be a machine that processes the entities or a holding area that stores these entities. The SAD technique represents these two different types of locations as two separate modelling primitives, a primary resource element and a queue element.

4.3.3 Primary resource element

In any discrete system as a product transitions from one phase of change to another a transformation of some sort generally takes place. Such a transformation can only take place with the aid of a resource which facilitates such transformation taking place. In the SAD technique a primary resource element is used to graphically represent such a resource.

> ***"A primary resource element represents any resource location within a discrete event system which facilitates the transformation of a product, physical or virtual, from one state of transition to another."***

4.3.4 Queue resource element

Discrete event systems generally cycle between phases of activity and waiting. Therefore to model a discrete event system, even a simple system such as that shown in Figure 4.5 there is a need for an element to represent the various phases of waiting. Even in a simple system such as that shown in Figure 4.5 there may be a stage of waiting where the entities in state 1 wait to be transformed. There may also be a second phase of waiting where the entities that have been transformed from state 1 to state 2 wait to undergo further

transformation or to exit the system. The SAD technique represents such waiting phases by means of the queue element.

> ***"A queue modelling element represents any location or phase of a discrete event system where a product, virtual or physical, is not in an active state of transformation within the system."***

Returning to Figure 4.5 if graphical elements are now added to this diagram to represent both the primary and queue resource elements Figure 4.6 is created.

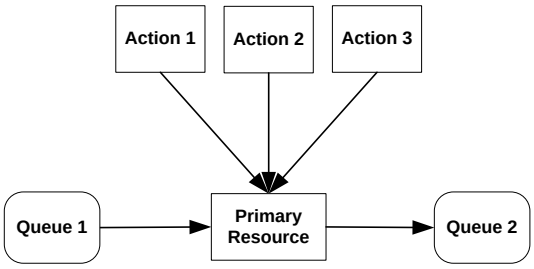

Figure 4.6 Queue and primary resource elements.

Therefore, the original simple system diagram has been embellished to show the system alternating between phases of waiting and activity.

4.3.5 SAD State Elements

Within any discrete event system, input product or products, be they physical or virtual, are taken into the system. These products then transition through a series of intermediate phases of change and, as a result of these phases of change the product or products exit the system in a changed format. The SAD modelling technique represents such products by introducing two modelling elements, an entity and an informational element.

4.3.5.1 SAD entity element

An entity element represents an actual product that is transformed by a discrete event system

> ***"An entity element represents any product, physical or virtual that is transformed as the result of transitioning through a discrete event system."***

This entity element also represents the various intermediate phases of production that such a product transitions through, by use of entity state elements that are directly associated with a particular entity element. In other words an entity can have any number of pre defined states through which it may transition during the process of transformation within a discrete event process.

Entity state element

In any system there are various phases through which a product will transition before the outcome of a finished product results. Within discrete event simulation the concept of an entity is used to represent any product or component that requires any form of transformation within a system, e.g. a customer being processed or a part being produced. A number of different products may be present in a particular system at a particular time and may require graphical representation. Within the SAD technique the concept of an entity state is used to represent the various transitional phases of every such product.

> ***"An entity state represents any of the various states that a physical or virtual object, explicitly represented within a discrete event system, transitions through during physical transformation"***

Applying this concept of an entity state to the simple system in Figure 3.6 results in Figure 4.7.

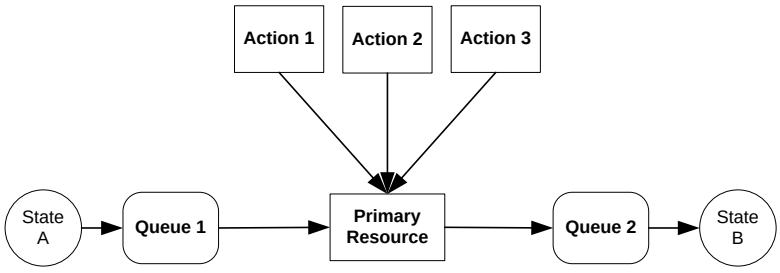

Figure 4.7 Entity states

In this simple system, a product transitions from entity state A to entity state B as a result of the execution of actions 1, 2 and 3. Therefore in this simple example the action list represents the mechanism for the transition from state A to state B. In the same way a series of action lists can be used to represent the transition of

a product through the various entity states. Therefore, a SAD becomes a series of action lists. Each action list having an informational or entity state element as its beginning and terminating point. In this way each action list represents the state that a system is currently in, along with the criteria which must be satisfied prior to the system transitioning to the following state, represented by the terminating state. This terminating state then becomes the beginning state of any following action list or lists thus linking together into a SAD model.

4.3.5.2 SAD Informational element

An informational element represents any information used in the operation or control of a discrete event system.

> ***"An informational element represents any information that is used in the control or operation of the process of transition by an entity, through a discrete event system."***

Such an informational element also represents the various intermediate phases of transition by such information, by use of the informational state elements that are directly associated with a particular informational element. This association is the same as that of entity state elements with an entity element. In this way an informational element can also have any number of pre defined informational states through which it may transition during its process of transformation within a discrete event process.

Informational state element

In any information system there are various phases through which information transitions before finally reaching its end state. Within discrete event simulation, information systems (a computerised order processing system or a kanban system) that can be used to support the operation of an actual discrete event system can be modelled. To allow the representation of such control structures within the SAD technique, the SAD informational state modelling is used.

> ***"An informational state represents any of the various states that information, used in the control or operation of the process of***

transition by an entity state through a discrete event system, can transition through."

4.3.6 Auxiliary resource element

While a primary resource is used to facilitate the transformation of any entity from one state of transition to another, the primary resource rarely operates in isolation. Generally, a primary resource is used in conjunction with other resources, known as auxiliary resources within a system. These auxiliary resources are used to support the facilitation of the transformation of any entity from one state of transition to another.

"An auxiliary resource represents any resource used in the support of any phase of transition of any state element within a system"

Therefore, within a simple system being simulated a primary resource, such as a machine may be used in the transformation of an entity from state A to state B. However this primary resource may require an operator and a number of other tools that an operator may use to operate the machine. When simulating such a system auxiliary resources such as these only become critical when such resources are scarce and as a result impact on the time taken for the advancement of a process. Such a scenario may include an operator being shared between a number of primary resources. This operator may be necessary to support the operation of each machine. As a result of this, scheduling conflicts may arise from time to time, where the operator may be required to support two primary resources simultaneously. It is generally only in such instances that auxiliary resources are modelled. However, such auxiliary resources can be extremely useful in facilitating the understanding of systems being modelled. Often such auxiliary resources represent operators and persons who operate the system on an ongoing basis. Therefore, if such systems can be modelled in terms of such auxiliary resources it would greatly enhance the model's ability to communicate effectively the detailed logic of the system being modelled.

In graphically representing these auxiliary resources within the SAD technique a distinction is drawn between two specific different types of auxiliary resource, namely an actor and a supporter auxiliary resource.

> ***"An actor auxiliary resource represents any auxiliary resource used in the direct support of the execution of an action or actions within the process of transitioning a system from one state to another"***

> ***"A supporter auxiliary resource represents any auxiliary resource used in the direct support of an actor auxiliary resource or primary resource in the execution of an action or actions within the process of transitioning a system from one state to another."***

For example within the SAD technique, an actor auxiliary resource may be used to represent a milling machine operator, while a supporter auxiliary resource may be used to represent the various equipment used by the operator to carry out the various tasks on the milling machine. This scenario may also be devoid of an operator, in such an instance the actor auxiliary resource may be used to represent the part of a discrete event system which triggers or controls the execution of given tasks. By creating such a distinction the SAD technique draws a distinction between resources such as operators that are used in the support of a system and other auxiliary resources that may be used to support operators in the execution of their actions.

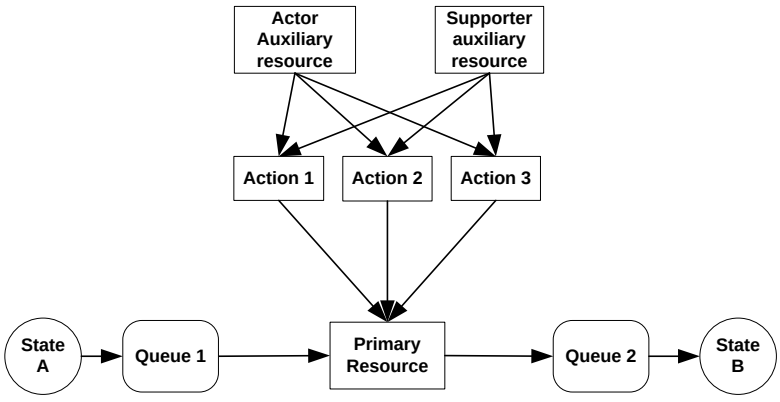

Figure 4.8 Auxiliary resources

Figure 4.8 shows a SAD diagram with two auxiliary resource elements, in such a situation the SAD diagram is executed from left to right. Where the auxiliary resources actions and primary resources are connected in a thread from the centre to the extremities of the model the execution sequence is from the central resource elements to the extremities of the model. In this instance the left to right convention still takes precedence a number of elements such as actions are in series.

4.3.7 Branching Elements

On examination of the elements in Figure 4.8 a number of semantic ambiguities become apparent. Firstly the links between auxiliary resources, “actor” and “supporter”, and the actions shown are ambiguous. In this instance the meaning of the links are unclear, either one or both of the auxiliary resources may be necessary for the execution of each action or any number of the actions. A similar ambiguity may arise within the graphical representation of the various phases of execution within a system. In the simple system shown thus far the execution sequence is linear from state A to state B. However most discrete event systems are complex in nature and rarely, if ever, linear. Instead they are often made up of some or all of the general sorts of branching as listed below.

- Points where the sequence of logical execution of either the information or physical system branch into multiple parallel lines of execution;
- Points where the sequence of logical execution of either the information or physical system branch into multiple alternative lines of execution;
- Points where the sequences of logical execution of multiple lines of execution converge back into a single line of logical execution;
- Points where the sequences of logical execution of multiple alternative lines of execution converge into a single line of execution.

To overcome such ambiguous situations the SAD technique uses a number of branching elements, which have been adopted from the IDEF 3 process

modelling technique [72], (Section 2.2.9.2). There are two general types of branching elements, fan in and fan out.

Both of these branch types can be further sub divided into conjunctive and disjunctive branch elements. Where conjunctive branch elements represent the branching and joining of multiple parallel sub systems and disjunctive branch elements represent the branching and joining of multiple alternative sub systems.

A logical, “AND”, branch element is used to represent conjunctive branching. While there are two types of disjunctive branch elements, inclusive and exclusive, represented by an “OR” and an “XOR” respectively. Finally, each of the branch elements introduced may be either synchronous or asynchronous. Where a synchronous branch element signifies that all elements either preceding or proceeding the branch element depending on its type, fan in or fan out, must either begin or end simultaneously. An asynchronous branch element does not require such simultaneous initiation or termination and is therefore the more commonly used. Figure 4.9 shows the various types of branching elements used in the SAD modelling technique.

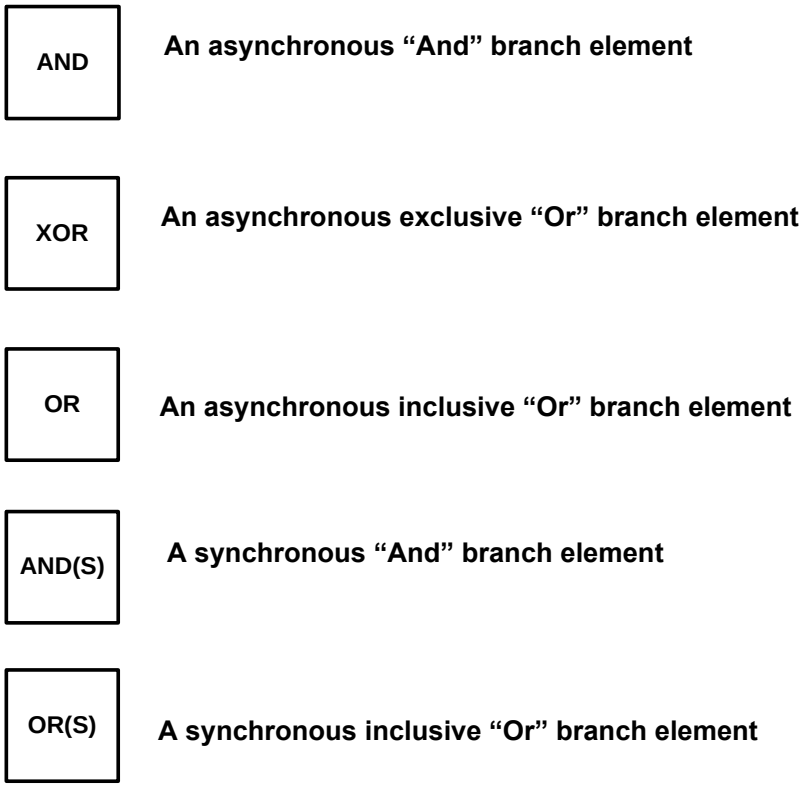

Figure 4.9 SAD Branching elements.

4.3.7.1 Using branch elements

A fan out, “AND” branch in a model means that when the execution of the model reaches that point in the process represented by such a branch, all the elements that are immediate successors of the branch will be executed. If a synchronous, “AND(S)” branch is used then the execution of that branch will mean that all of the immediate successor elements must begin execution simultaneously.

Similarly in a model where a fan in, “AND”, branch is executed all elements that immediately precede that branch will have been executed. If a synchronous, “AND(S)”, branch is used, then, for that part of the model to execute all the elements preceding must all end simultaneously. Thus, an execution of the left hand model in Figure 4.10 will consist of the execution of element, A, followed by elements B and C. Similarly the execution of the right hand model in Figure 4.10 will result in the execution of an element, C, preceded by the execution of

elements A and B; If a synchronous, “AND(S)”, branch is used, then for there to be an execution of the element, C, both elements, A and B must end simultaneously.

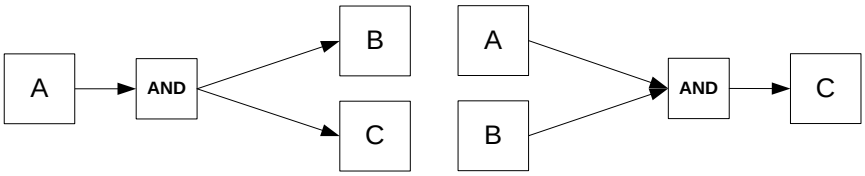

Figure 4.10 AND Branches

A fan out inclusive, “OR”, branch in a model indicates that, in an execution of that branch there will be an execution of at least one of the elements connected to the branch to the right. Similarly, a fan out exclusive, “XOR” branch in a model indicates that, in an execution of that branch, there will be an instance of exactly one of the elements connected to the branch to the right. If a synchronous inclusive, “OR(S)” branch is used, then all elements that are executed must start simultaneously. This does not apply to exclusive, “XOR” branches, since there can only be one element executed in an XOR execution. Similarly with fan in inclusive “OR” branch, there will be at least one element executed to the left of the branch. If a synchronous inclusive “OR(S)” branch is used, then, those elements that are executed, if there are more than one, must all end simultaneously. Hence, an execution of the model to the left in Figure 4.11, consists of an instance of the element A proceeded by an instance of either B or C, or both. If the models in Figure 4.11 used XOR branches, then an execution of the first model could not include an instance in which the execution of both B and C occur while an execution of the second model could not include an instance where an execution of both A and B occur.

Figure 4.11 OR Branches

Referring to Figure 4.8 auxiliary resource branching elements may be used to illustrate the use of such resources in the execution of actions. The use of branching elements allows for the graphical representation of the sequence of use of the elements in the execution of such actions. Again, referring to Figure 4.8, there are three links emanating from each of the actor and supporter auxiliary resources and linking to each of the actions, 1, 2 and 3 respectively. However, it is not readily apparent from this diagram if both resources are used in the execution of all actions and, or, if the resources are needed to be used simultaneously or not to execute the actions. To overcome such ambiguous situations, the branch elements can be used as shown in Figure 4.12. In this diagram the branching elements are used to model the divergence of the links into multiple paths by means of an asynchronous “AND” branch in each case. This graphically represents the fact that each of the auxiliary resources are used in the execution of the three actions. The convergence of these links back into a single path is also represented by a branch element in this instance a synchronous, “AND(S)” branch. This graphically represents the fact that each of the two links converging at this branch should be present simultaneously for the execution of the exiting link. In other words both the actor and supporter auxiliary resources have to be present at the same time for the execution of each of the actions 1,2 and 3. Finally the use of the and asynchronous branch, “AND”, to link actions 1, 2 and 3 with the primary resource element indicates that the actions 1, 2 and 3 have to be executed prior to the SAD model advancing past the primary resource element. In other words the three actions have to be executed prior to any transformation of an entity taking place.

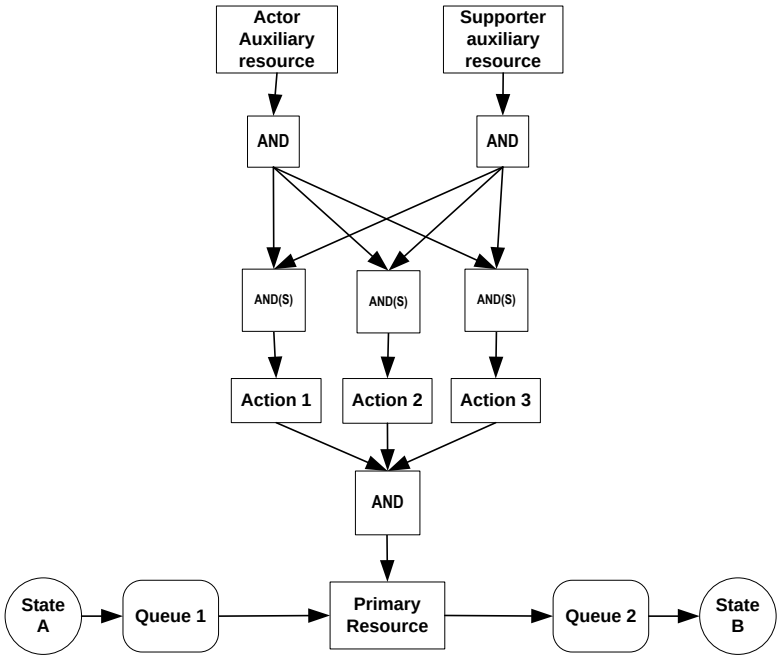

Figure 4.12 Use of branching elements

4.3.7.2 Combining Branching elements

As previously introduced the SAD technique has been developed to represent complex discrete event processes in which multiple parallel and alternative paths are capable of being linked together into a single representation of a system. The ability to model such complex representations lies in the use of the SAD branch elements to represent such discrete event processes. These same branch elements are also used to model complex associations between the various resources and actions in an action list. Some basic combinations of branch elements are illustrated here.

It is common to find processes in which a single path diverges into multiple paths and then, at some later stage converges back into a single path. SAD represents such processes by combining fan out branches and fan in branches. Figure 4.13 represents a process where a path diverges into parallel paths and then converges. Because the processes run in parallel but do not need to begin simultaneously, they are represented in this instance by asynchronous, “AND”, branches.

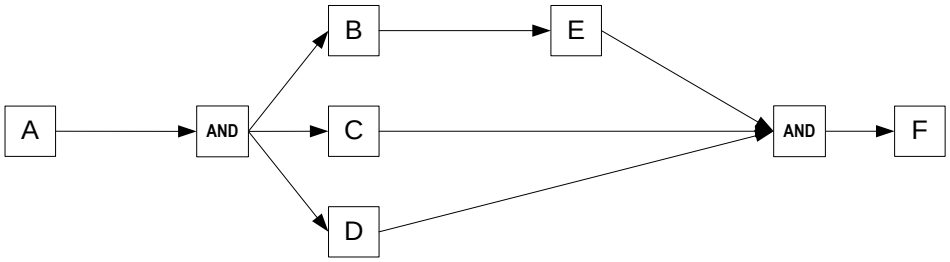

Figure 4.13 Asynchronous "AND" Branches

Because the first asynchronous "AND" branch element separates element A and elements B, C and D in an execution of this model, element A will be executed before any of the succeeding elements are executed. The execution of the model in Figure 4.13 will be as follows. After element A, the three elements (B, C and D) will be executed. Because the first And branch is asynchronous, B, C and D can begin in any order. Because all three paths converge at the second and asynchronous, "AND" branch element F will only be executed after elements E, C and D have been executed. Because this second and branch element is also asynchronous, no particular order of execution is necessary.

Figure 4.14 shows the same model but with and synchronous branches, "AND(S)", being used. Again the element A has to be executed before the succeeding elements can be executed. An and synchronous branch element, "AND(S)", indicates that in an execution of such a model the elements B, C and D will begin simultaneously. Also the use of a second and synchronous branch element, "AND(S)", indicates the simultaneous completion of the execution of all three elements D, C and E before the process continues to the execution of element F.

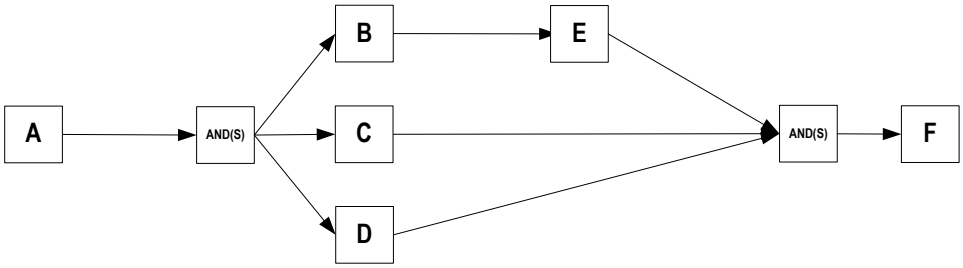

Figure 4.14 Synchronous "AND" Branches

Figure 4.15 shows the same model but with or asynchronous branches, “OR” being used. In such a model the first or asynchronous branch, “OR”, indicates that following an execution of A one or more of the elements B, C and D will be executed. Because the second branch is also an or asynchronous, “OR”“, branch element, only one of the lines of execution has to be completed before element F is executed.

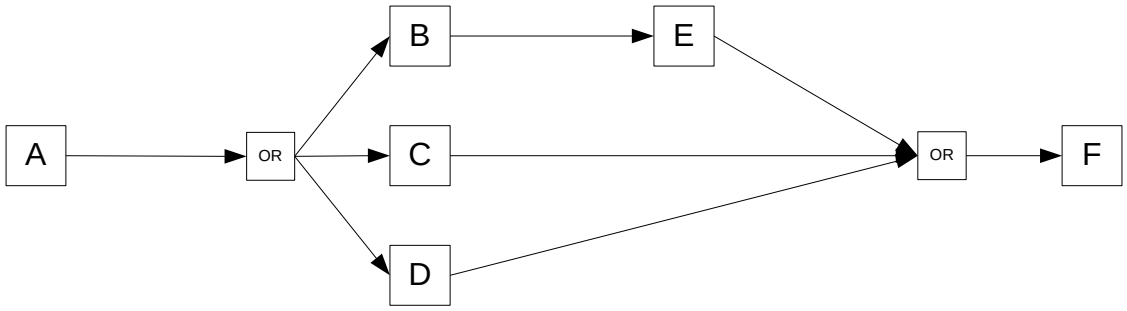

Figure 4.15 Asynchronous “OR” Branches

Two or synchronous branches, “OR(S)”, are used in the model shown in Figure 4.16. Again an or synchronous branch element, “OR(S)”, indicates that following an execution of A, one or more of the elements B, C and D will be executed. As the branch type is synchronous if more than one element is to be executed, they occur simultaneously. If the line of execution is along element B it will be followed by the execution of element E which will complete at the same time as any other lines of execution that were initiated along with the line containing element B.

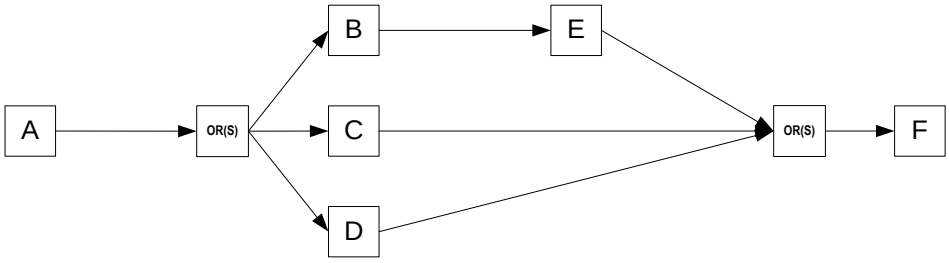

Figure 4.16 Synchronous “OR” Branches

Different branch types may also be used in the execution of models. In Figure 4.17 a simple model is presented showing such an occurrence. In this model an and asynchronous branch element, “AND”, and an asynchronous or branch element, “OR”, are used. After the execution of element A the two lines of

execution will be executed, illustrated by use of the and asynchronous branch element, “AND”. However, in this situation the use of the asynchronous or branch element, “OR”, illustrates that one or other of the lines of execution may not complete or even initiate, before the execution transitions to the asynchronous or branch element, “OR”, and the execution of E takes place.

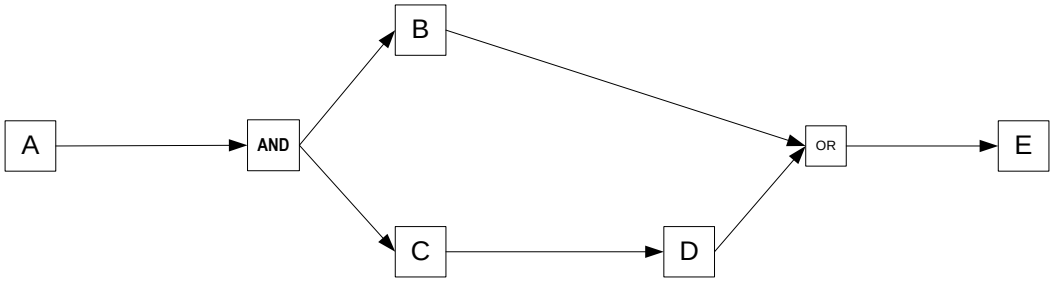

Figure 4.17 Use of different branch types together in the same model

For this model to execute successfully all lines of execution must be completed at some instance, however it is sufficient for only one of the lines of execution to be completed prior to the execution of element E.

4.3.8 Link Types

Links are the glue that connect the various elements of a SAD model together to form complete processes. Within the SAD technique there are three link types known as entity links, information links and activity links. The symbols that represent each type are shown in Figure 4.18.

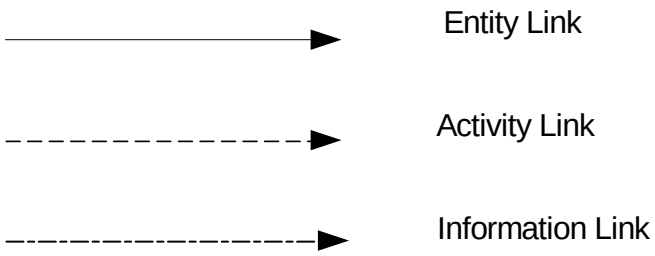

Figure 4.18 SAD Link Types

4.3.8.1 Entity Links

As introduced previously, within any discrete process input or inputs are taken in and through a series of transformations output or outputs are created. To

represent the physical flow of a product through such a discrete system and the relations between SAD elements used in the physical transformation of such products, the entity link type is introduced.

> ***"An entity link represents the physical flow of a product, actual or virtual, through a discrete system along with the relations between instances of elements used in the physical transformation of such products within a model."***

Entity links connect elements as shown in Figure 4.19, where A is the source of the entity link and B is the destination

Figure 4.19 Entity link

4.3.8.2 Information Links

In modern discrete event systems there are often two systems that operate in close co-operation with each other. Namely the system charged with the physical transformation of the product along with the system that supplies information on or to the physical system. The latter system will be referred to as the information system from here on. Such a system may be used to simply provide feedback or historical information on the physical system performance. It may also help to control the performance of such a physical system. To represent the flow of information through such a discrete system and the relations between SAD elements used in the transformation of such information, the information link type is introduced.

> ***"An information link type represents the flow of information through a discrete system along with the relations between instances of elements used in the transformation of such information within a model."***

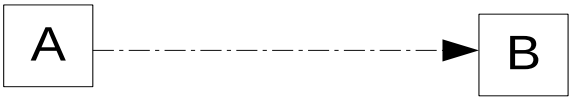

Figure 4.20 Information Link

Information links connect elements as shown in Figure 4.20. Again in this model A is the source of the information link and B is the destination.

4.3.8.3 Activity Links

Returning to the basic concepts that make up a SAD, model the concept of an action list being used to represent a SAD activity is central. In this action list the various SAD elements that are responsible for the transformation of either products or information are combined to represent the various stages of execution. To link these various SAD elements together a third link type is introduced, namely an activity link type.

> ***"An activity link type represents the relations between various SAD elements used in the execution of each SAD activity."***

Figure 4.21 Activity Link

Activity links connect elements as shown in Figure 4.21. As with the previous link types A is the source of the activity link and B is the destination.

4.3.9 SAD Frame Element

Thus far the modelling elements introduced provide a detailed view of the interaction of a discrete event system at a particular level of operation. However, discrete event systems are generally complex in nature, with highly detailed interactions taking place on a number of different levels. For example, at a

certain level of a discrete event system a supervisor may control the entire system while at a lower level operators may control various aspects of the same system. Often simulation models have to model a number of levels of operation within such systems along with their interactions with each other. These interactions can be complex and it is often advantageous for a model developer to abstract scenarios at a particular level of operation within such a process into their component elements, while also modelling the interactions that such scenarios have with the rest of the system under examination. To facilitate such a process the SAD technique introduces the frame modelling element. This element allows a model developer to model in detail a particular section or sections of a discrete system, along with showing how such sections interact with the entire system being modelled. By using such an element, a model developer can develop a hierarchical model of a particular system, thus allowing the decomposition of a system into its more complex parts as required.

> ***"The SAD frame element provides a mechanism for the hierarchical structuring of detailed interactions within a discrete event system into their component elements, while also showing how such elements interact within the overall discrete event system."***

In this way frame elements allow the model developer to decompose a process to varying levels of abstraction. By repeatedly applying such an element, it is possible to hierarchically structure a process description to any level of detail.

In Figure 4.22 the use of the frame element to decompose a simple system is illustrated.

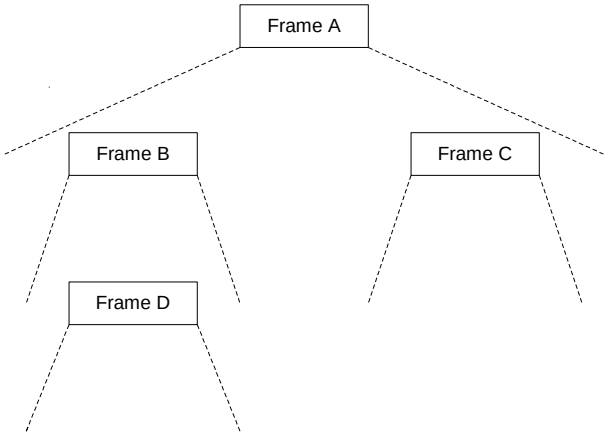

Figure 4.22 Frame elements

4.4 Developing a Simulation Activity Diagram

In the following section the logical development of a SAD model will be introduced. This introduction will be by means of a simple example of a discrete event system, which will be embellished.

In Figure 4.23 a discrete event system is shown transitioning from one state, 1, to another state, 2.

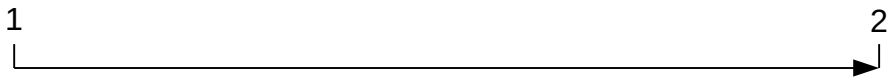

Figure 4.23 A simple system.

In transitioning between 1 and 2 the system will have to carry out at least one event or activity, A, since a discrete event system will only change state as the result of a stimulus of some kind. In the simplest system such as above there is only one activity resulting in a transition between states 1 and 2 (Figure 4.24).

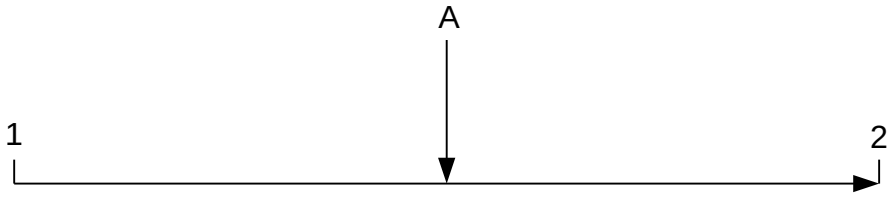

Figure 4.24 An activity in a discrete event system.

If there are more than two states that a system can transition into as a result of such an activity then a situation such as that in Figure 4.25 arises. In this case the system can transition from state 1 to either state 2 or 3 as a result of a decision, D, which is made as a result of an activity, A.

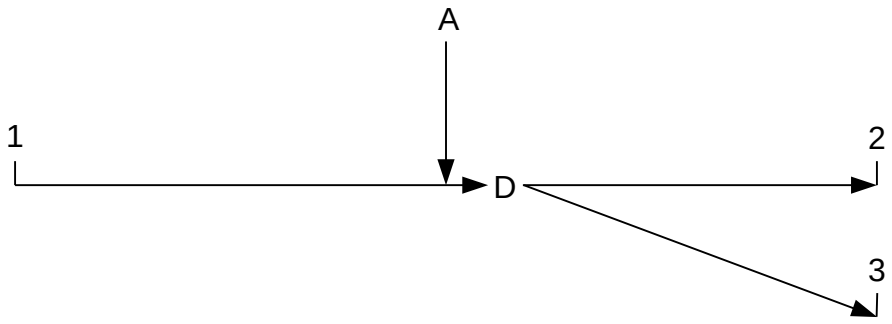

Figure 4.25 A system transitioning at a decision point, D, as a result of an activity, A.

As previously introduced such an activity can be further subdivided into a constituent action list within the SAD technique.

4.4.1 An Activity and an Action list

In the system shown in Figure 4.25 an activity, A, is responsible for the change from state 1 to either state 2 or state 3. In this section the make up of the action list associated with this particular activity is described in detail.

For an activity to be executed there has to be an actor present, which is an object or person who will facilitate the execution of the action or actions. Figure 4.26 shows such a scenario. Here actions A, B and C are executed by an auxiliary resource element, “Actor 1”. This is depicted graphically by use of the asynchronous and, “AND”, fan out branch element between the auxiliary resource element, “Actor 1”, and the three aforementioned actions, all of which are joined by the activity links. Similarly all three actions are executed on a primary resource element, “Machine X”. This is again depicted in this diagram by the use of the asynchronous and, “AND”, fan in branch between the actions A, B, C and the primary resource element, “Machine X”. These are again joined together by the activity links. Therefore Figure 4.26 presents a simple action list

in which three actions are executed to fulfil the requirements of an activity. In the action list depicted both branch types are of the asynchronous and, “AND” type. However these branches could be any of the previously introduced branch types.

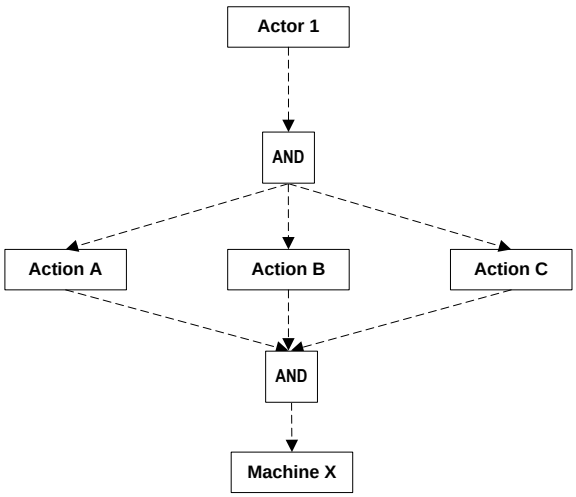

Figure 4.26 An activity with a number of actions.

If this action list is now embellished to include a supporter auxiliary resource to be used in support of the actor auxiliary resource in executing one or more of the actions being executed as part of this action list, a diagram such as that shown in Figure 4.27 is created. In this instance as before the actor auxiliary resource element, “Actor 1”, executes actions “A”, “B” and “C” on a primary resource element, “Machine X”. In this instance however the actor auxiliary resource “Actor 1” uses a supporter auxiliary resource, “Supporter 1”, in the execution of “Action A”. In other words the actor auxiliary resource element, “Actor 1”, and the supporter auxiliary resource element, “Supporter 1”, need to be present at the same instance to enable the execution of “Action A” on the primary resource element “Machine X”. This is depicted graphically by the use of the and synchronous, “AND(S)”, fan in branch between the supporter auxiliary resource, “Supporter 1”, element and the action element “Action A”.

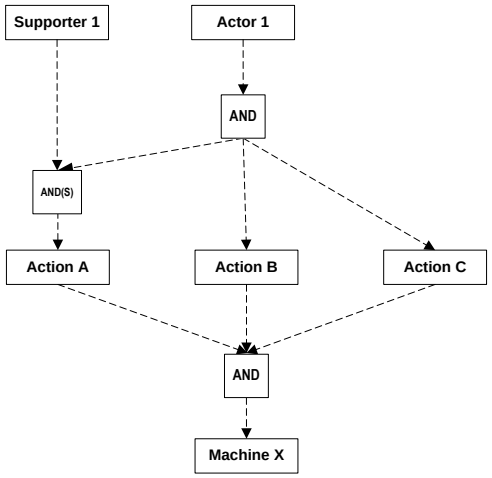

Figure 4.27 An activity incorporating resources.

This diagram graphically depicts that the actor auxiliary resource, “Actor 1”, is necessary for the execution of all three actions while the supporter auxiliary resource, “Supporter 1” is necessary to be present for the execution of a single action, “Action A”.

Returning to Figure 4.25 which depicts a system transitioning from state 1 to either state 2 or state 3 as a result of some external activity, “A”. If this external activity, “A”, is now replaced by the detailed action list shown in Figure 4.27 then we have a diagram as in Figure 4.28, a simple but complete Simulation Activity Diagram (SAD).

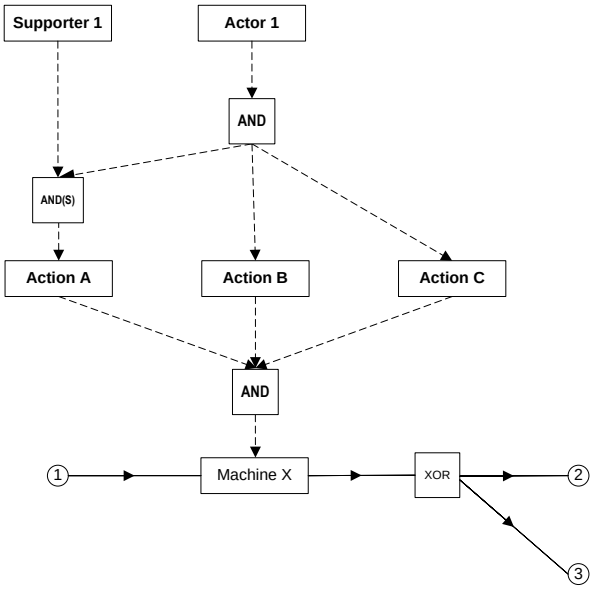

Figure 4.28 A simple Simulation Activity Diagram (SAD).

In this SAD we have a detailed view of the activity that takes place prior to the transitioning of a system from it's current state to a resultant state. This detailed view of the activity shows the logical sequence of the actions that make up the activity. Also graphically represented are the actors, or the people or resources that execute the activity, and the supporting auxiliary resources along with the primary resource where this activity is executed. The logical sequencing of the interaction of these actors, primary and auxiliary resources are all graphically represented by means of the fan in and fan out branches of various types. Therefore Figure 4.28 represents a system that transitions from state 1 to either state 2 or state 3 based on the result of the actions executed in the action list shown prior to the exclusive asynchronous or fan out branch, "XOR". This action list represents the actions that have to be executed in order to fulfil the requirements of the aforementioned activity along with the resources that are necessary to facilitate the actions.

4.4.2 Extending SADs to include systems information data

Most modern manufacturing processes are accompanied by a large amount of information that is used to support and control the various stages of processing. This information may move in many different directions, in parallel with the flow of parts or opposite to it. It may also be unrelated to direct part flows.

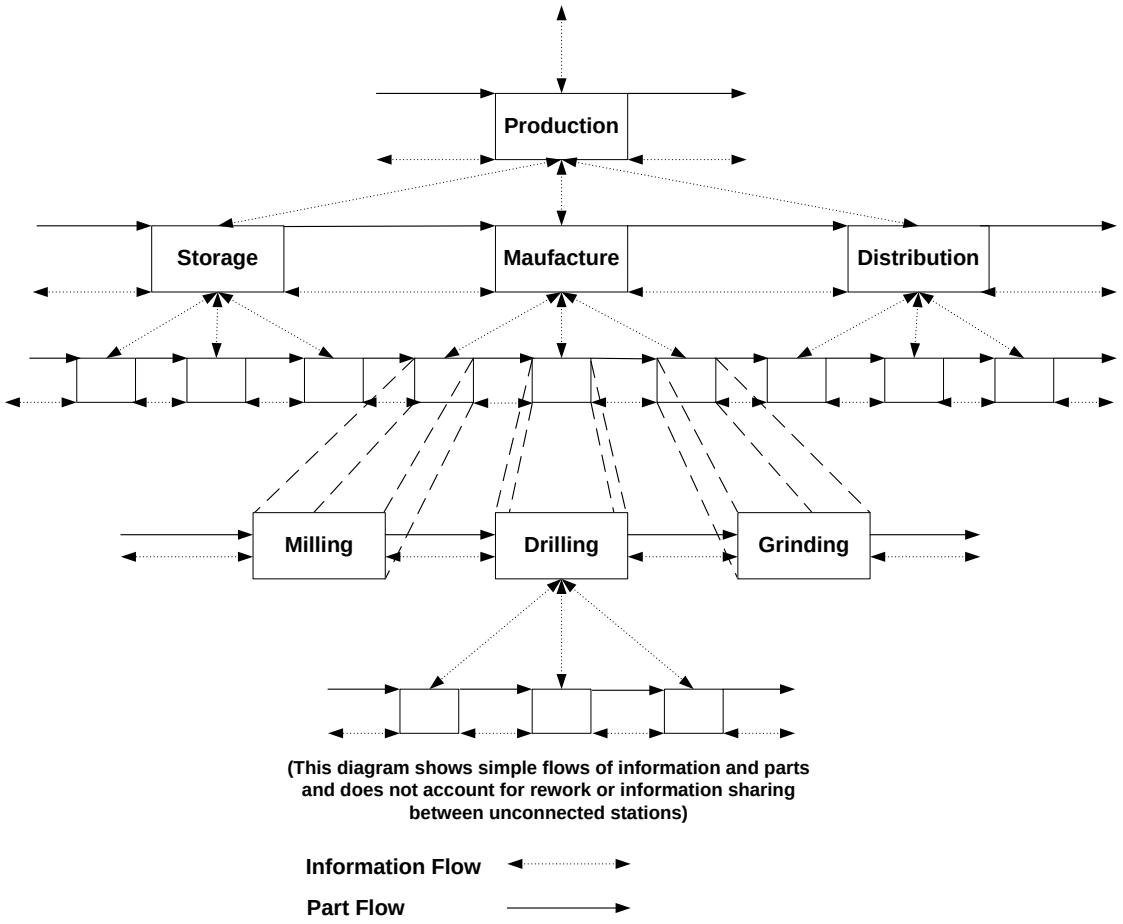

Figure 4.29 Information flows within a manufacturing system.

Such information can affect any stage of processing and any element that is involved in the process. Figure 4.29 shows an example of the complex nature of information and its interaction with a production system. Elements introduced previously included an informational element and informational states to visually represent for such information. If Figure 4.28 is again taken as an example of a simple SAD, the changes of state of the physical system are shown with an entity physically transitioning from state 1 to either states 2 or 3. If we now substitute the entity states for information states we have a similar SAD, but one that now represents an information system. Therefore, to visually model a manufacturing system that includes informational flows we are in fact modelling two separate systems that are intrinsically linked. These systems are linked by the fact that certain changes in one system may cause a change in either one or both systems. To account for this, the original SAD can be extended to incorporate a

second transitional flow of states, which in this instance represents the informational system, Figure 4.30.

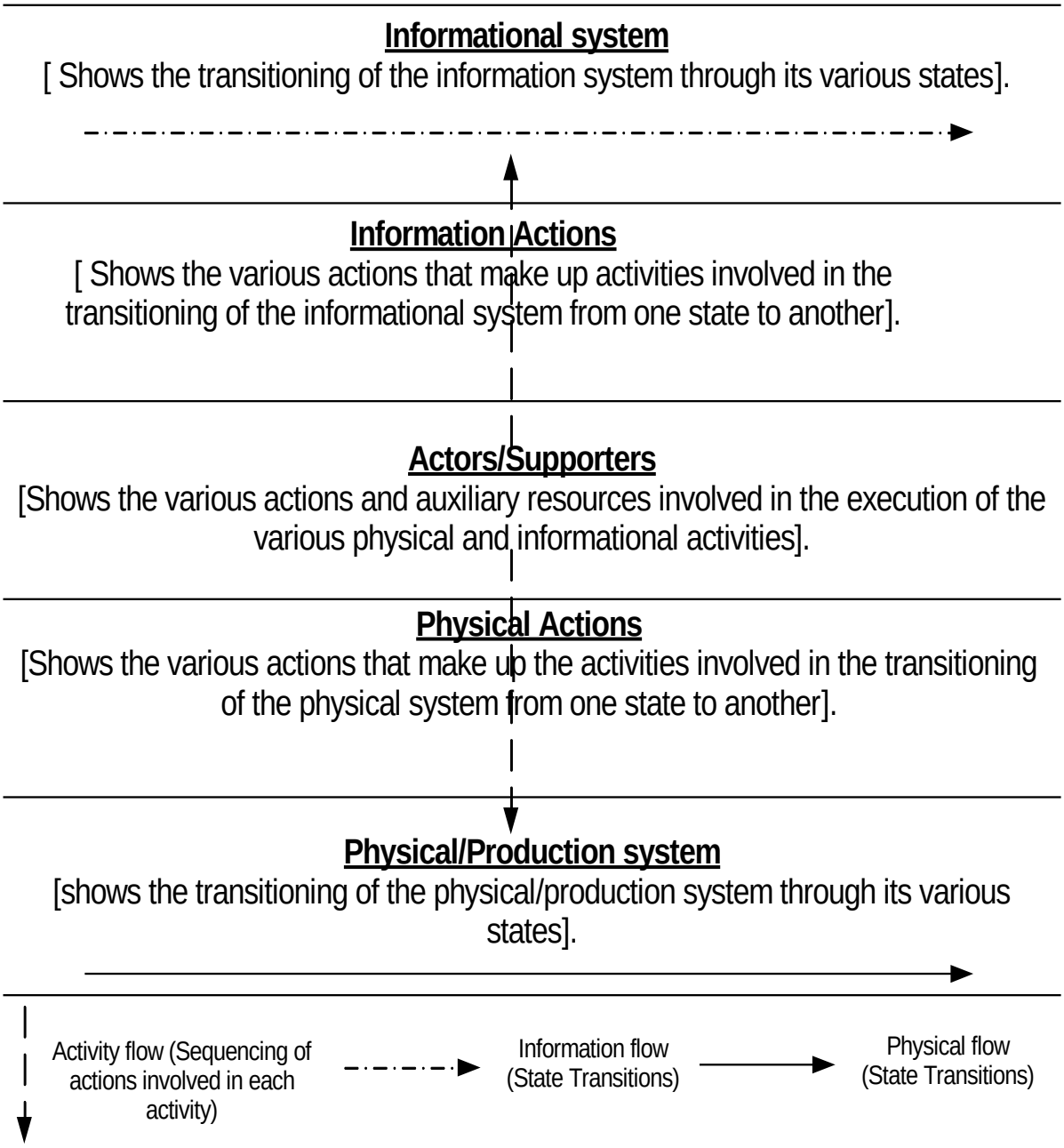

Figure 4.30 Extended SAD structure

Thus in this extended SAD, a model is broken into two sub or partial models, namely the information and physical models. The extended SAD is designed as

follows. At the centre of the model are located the actors and supporters also known as auxiliary resources. These are the drivers for both the information and physical models. This is advantageous for the purposes of communication during the requirements gathering phase of a simulation project as the persons with whom the simulation model developer will be communicating may be actors within the process. Therefore, in such instances each SAD model will be developed from the perspective of the persons interacting with the system.

The interconnecting areas between both models contain the actions to be executed. A series of these actions and the associated interactions with other SAD modelling elements make up an action list or activity. A series of these activities in turn make up a sequence of transitions for a product or family of products within a discrete event system. Figure 4.31 shows the previous SAD for a physical system but it has now been extended to include a simple informational system.

Within the informational model the system is at an informational state, “A”, and has two states that it can transition to, “B” or “C”, based on the results of a series of actions that are carried out on the primary resource element, “Resource Z”. The logical sequence of these actions along with the location of the execution of such actions is shown within the information actions section of the model. Here the actions that make up this activity are shown, as is the logical sequence of their execution and where theses actions are carried out. All of this is shown by means of the various branch types and activity links. At the centre of each SAD model is the section that contains the auxiliary resources, both actor and supporter types. In this section the resources that are used to support the execution of the activities are shown.

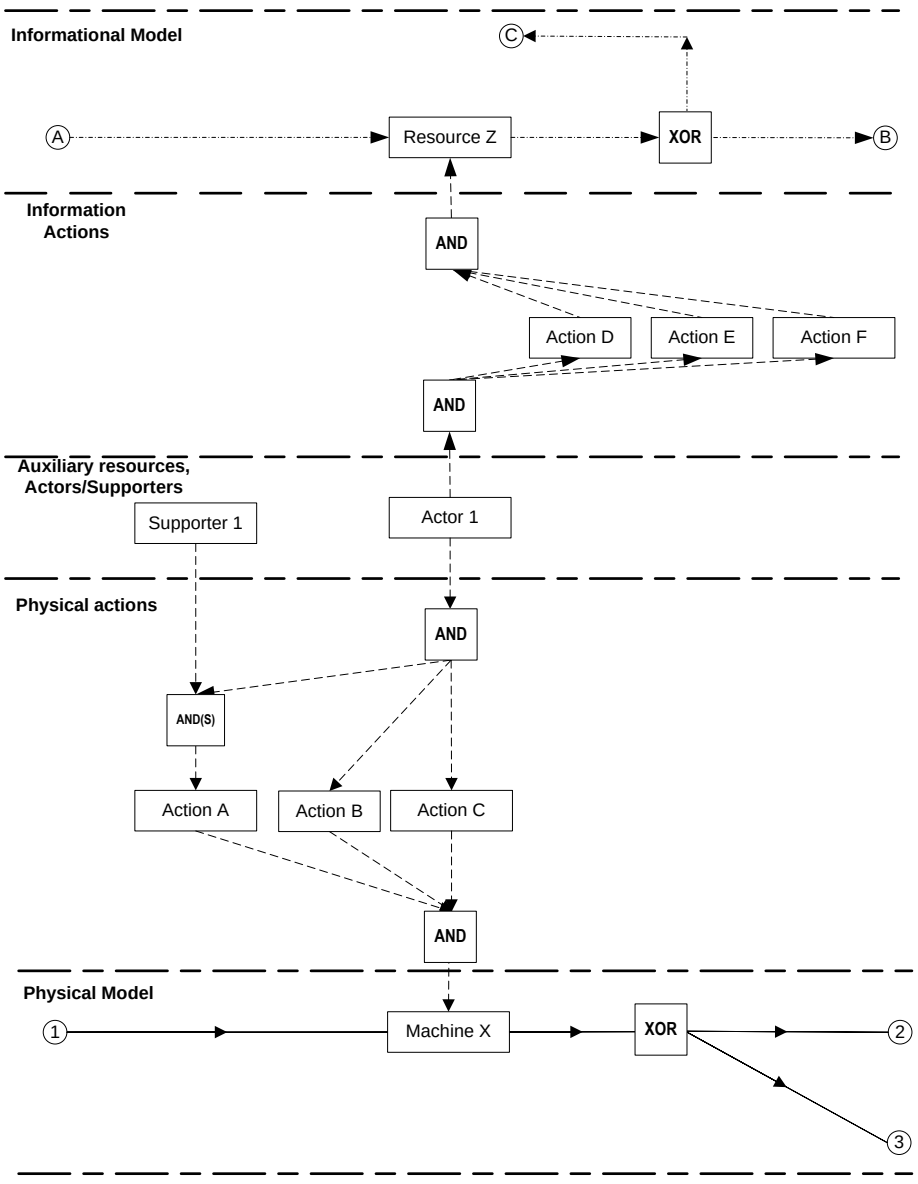

Figure 4.31 An Extended SAD.

In this simple example there are two auxiliary resource elements namely supporter auxiliary resource element, “Supporter 1” and the actor auxiliary resource element “Actor 1”. In the case of the informational model only the actor auxiliary resource element “Actor1” is used. This actor auxiliary resource element “Actor 1” is then used in the execution of all three of the actions, “D”, “E”, and “F”.

The lower half of Figure 4.31 shows the physical model. Again this section is made up of a number of sub-sections. The physical model, shown at the lower extremity of the extended SAD shows the possible physical states that the

system can transition through. Such transitions only take place as a result of the execution of all necessary actions, which are executed from left to right within the SAD model. In this case the physical system can transition from state “1” to either state “2” or state “3” as a result of the actions carried out on the primary resource element, “Machine X”. The auxiliary resources section again details what resources are used in the execution or in the support of the execution of each of the actions. In this case the actor auxiliary resource, “Actor 1” is used in the execution of each of the three actions “A”, “B” and “C”. However, again in this case the supporter auxiliary resource, “Supporter 1”, is used only in the execution of action “A”. Therefore both of the auxiliary resources “Actor 1” and “Supporter 1”, denoted by the synchronous and, “AND(S)”, fan in branch element, have to be present at the same instance for the successful execution of “Action A”. All three actions are executed on the primary resource element “Machine X”. As a result of the execution of these three actions the physical system can undergo a transition from state “1” to either state “2” or state “3”.

4.4.3 Elaboration of SAD models

Thus far the modelling elements used to develop a SAD model have been introduced to provide a means of visually modelling discrete event systems. However, such graphical models are capable of only representing a certain amount of detailed information and knowledge. Often, complex discrete event systems contain detailed information and knowledge related to process interactions that cannot be captured well by such graphical representations. To provide a means of making such information available to a model user the SAD technique also makes use of an elaboration language with which each individual SAD diagram can be described in greater detail. This structured language makes use of a number of different reserved words to allow the description of SADs. These words will now be presented briefly.

This group of words are used to describe the various interactions that take place in a SAD diagram. While such interactions are represented by various branches, which show the convergence or divergence of a system at certain points within

the visual model, such branches may have a different semantic meaning to a user based on where within the model they are used.

USES The supporter resource may at times make use of auxiliary resources to execute an action or actions, in other words a supporter USES auxiliary resources.

TO Details the action or actions that are executed by use of an auxiliary resource by a supporter resource.

AT Specifies where the action or actions are executed.

TRANSITIONS TO Specifies the change of state of entity or information from one state to another.

The following are branching that are also used by the structured language.

THEN;

AND;

AND SIMULTANEOUS;

EITHER;

OR;

OR SIMULTANEOUS.

This elaboration facility is based on the use of the SAD branch modelling elements and allowing a model developer to use these branching elements as a structured language around which can be built a detailed textual description, using the elaboration language, of each section of a SAD model. These same textual descriptions can then be presented to a model user during the presentation of a SAD model. Such elaborations allow for the explanation and representation and dissolution of any ambiguities that may arise around any aspect of a SAD model.

In the case of Figure 4.31, the SAD diagram shows a number of actions that are to be executed; these actions result in the transition of the two elements, informational and entity elements. The structured English that may be associated with this SAD diagram is given below.

```
Actor 1
        USES
        Supporter 1
        TO
                Action A
AND
Actor 1
        Action B
        AND
        Action C
        AT
                Machine X
AND
Actor 1
        Action D
        AND
        Action E
        AND
        Action F
        AT
                Resource Z
THEN
EITHER
        Entity State 1
        TRANSITIONS TO
        Entity State 2
        AND
        Informational state A
        TRANSITIONS TO
        Informational state B
OR
        Entity State 1
        TRANSITIONS TO
         Entity State 3
        AND
        Informational State A
        TRANSITIONS TO
        Informational State C
```

This is a very simple example with no great detail added to the descriptions. However it is possible for a model developer to embellish such descriptions with details as necessary.

4.4.4 Hierarchical structuring of SADs

Thus far SADs have only dealt with a simple system. However in reality, modern manufacturing systems are not that simple. Generally, such systems are a complex network of hierarchically structured systems and departments, be they a vertically integrated manufacturing system or a supply chain manufacturing

system. For instance, a factory may consist of a number of different departments, which themselves may be composed of a number of different autonomous or interlinked sections or production cells. The SADs that have been introduced to date are not capable of displaying or communicating this hierarchical nature of modern manufacturing systems. To overcome this, a frame element, as introduced previously, is used as in Figure 4.32.

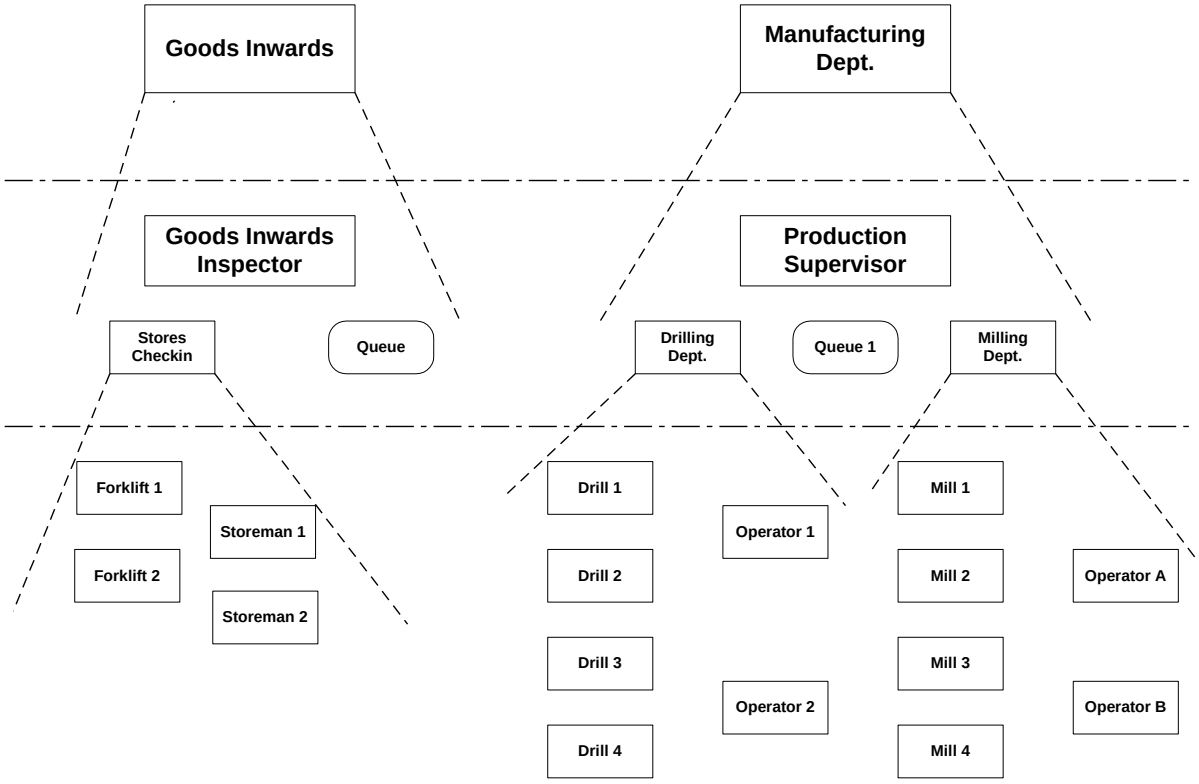

Figure 4.32 Hierarchical structure of a manufacturing system.

A frame therefore simply acts as a container element within which a more detailed SAD may be developed. In this way a frame allows the model developer to hierarchically structure a model to mirror the discrete event system under examination.

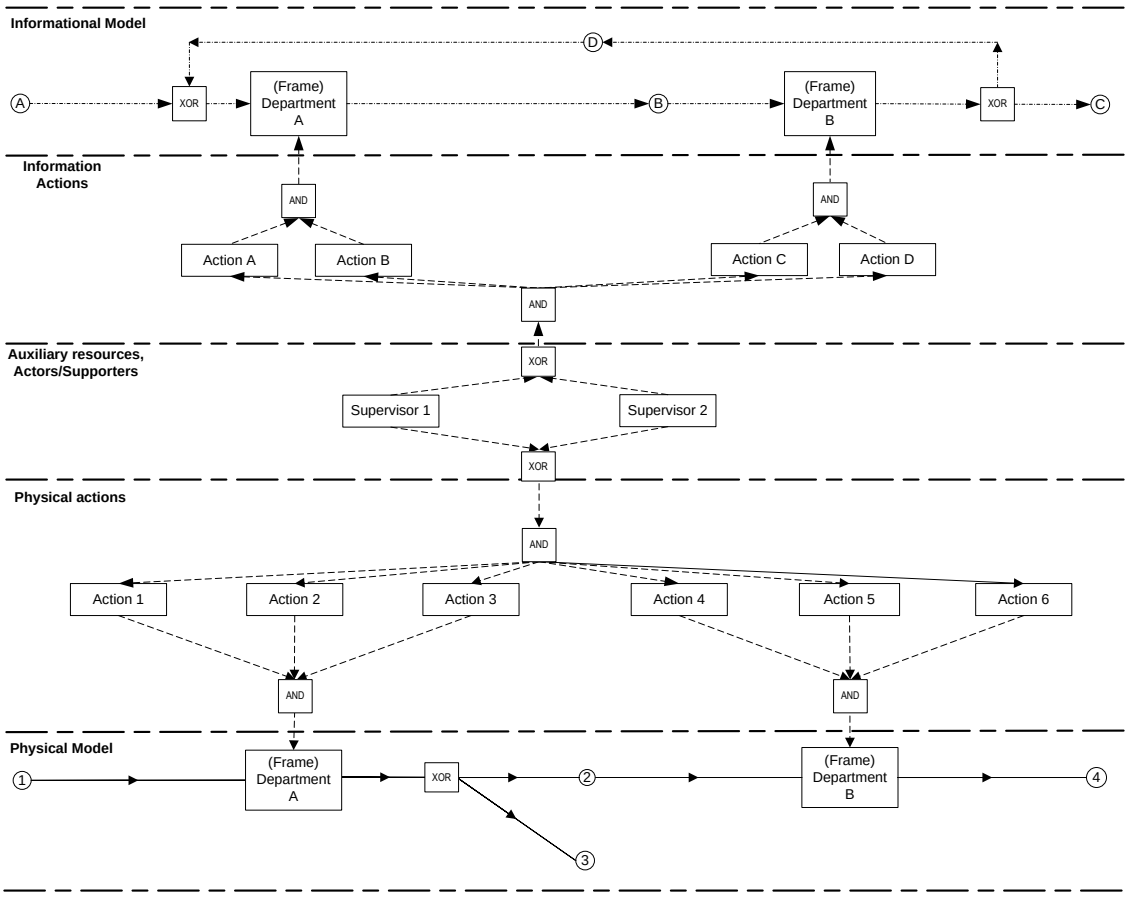

Figure 4.33 An extended SAD including frames.

Take for example a manufacturing system that consists of two departments “A” and “B”, at the highest level under investigation in this example two supervisors oversee the management of both departments. This scenario is represented in the SAD model shown in Figure 4.33. In this model either Supervisor “1” or “2” can carry out actions “1”, “2” or “3” at Department “A” and actions “4”, “5” or “6” at department “B”. The successful completion of these actions allows for the progression of the entity states from state “1” to either states “2”, “3” or “4”. Similarly in relation to the informational system either Supervisor “1” or “2” can carry out actions “A” and “B” at department “A” and actions “C” and “D” at department “B”. On successful completion of these actions the informational state “A” transitions to state “B”, and either states “C” or “D”.

Within this model of the supervisory level of the system under investigation frame elements are used to represent departments “A” and “B”. These frame elements

can then be used to develop more detailed models of lower level interactions within the system. For instance in relation to department “A”, a single operator works within this department producing output. This scenario is modelled in the SAD diagram shown in Figure 4.34. In this model “Actor 1” represents the single operator working within the department. In relation to the entity flows “Actor 1” carries out actions “A.1”, “B.1” and “C.1” at the primary resource, “Resource X”. The supporter resource, “Supporter 1”, is also used in conjunction with “Actor 1” in the execution of action “A.1”. On successful completion of these actions entity state “1” can progress to either entity state “2” or “3”. In relation to the informational model “Actor 1” again represents the operator working within department “A” and is used to execute actions “D.1”, “E.1” and “F.1” on the primary resource, “Resource Z.” On the successful completion of these actions the informational states can transition from either state “A” or “D” to state “B”.

In a similar way Figure 4.35 models the scenario for the frame element representing department “B”. In this SAD diagram there are three operators working, represented in this model by three actor auxiliary resources “A”, “B” and “C”. In relation to the entity flows in this diagram either actor “B” or “C” executes actions “2.1”, “2.2” and “2.3” at the primary resource, “Resource M”. In relation to the execution of action “2.1” the supporter auxiliary resource, “Supporter A”, is necessary for the execution of this action. On successful completion of the three actions entity state “2” transitions to entity state “4”. Referring to the Informational model, the actor auxiliary resource, “Actor A”, uses the supporter auxiliary resource, “Supporter A”, to execute action “G.1” and “H.1”. Either actor auxiliary resource “Actor B” or “Actor C” executes action “I.1” and “J.1”. All of the above actions are executed on the primary resource, “Resource W”. On completion of the required actions the informational state “B” can transition to either informational state “C” or “D”.

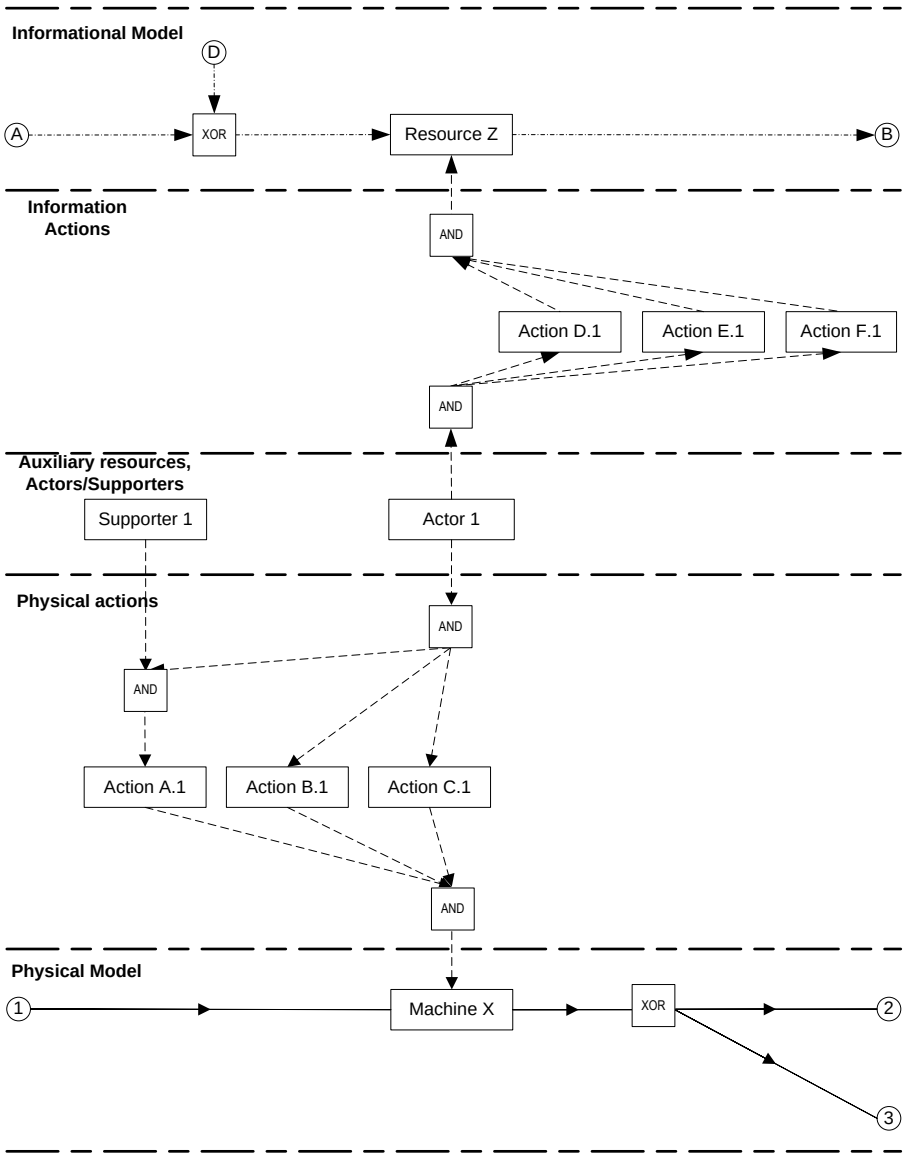

Figure 4.34 Extended SAD for Department A

In this way a model user can develop a lower level and more detailed view of the activities within each department. Such levels of abstraction allow the model developer to communicate the issues relative to a person at a certain level within a system being modelled, while separating them from unnecessary detail of other levels within the same system.

There can also be multiple occurrences of frames within a model as the same frame can be present in both the informational and physical models as in Figure 4.33. Such an occurrence can come about because an area will have both

physical entities and information entering it at a given instance. Frames can also be present within frames, thus allowing a hierarchical model of a system to be developed, showing the activities and associated resources at the levels of their utilisation within a system.

The example using the frame element shows a number of the unique aspects of this modelling technique. Most modelling techniques model either a physical or informational system. But as most modern manufacturing systems consist of both such systems operating simultaneously and interacting where necessary, and in turn many modern manufacturing simulation packages are capable of modelling both simultaneously. SADs allow the modelling of both systems along with the detailing of the interactions between both. Thus, allowing for the mirroring of both a modern manufacturing system and in turn a simulation model of the same.

Also unique to the SAD technique is the integration of decisional structures on both the horizontal and vertical plane. The horizontal plane models changes of state within both the physical and informational systems changes, while the vertical plane allows for the extensive detailing of the operations that bring about each individual state change. Finally, SADs also allow the visual display of how auxiliary resources interact in the execution of actions within each activity. This is again unique to this technique and allows a model developer to show where and when auxiliary resources are used in a model and how they affect overall system progression.

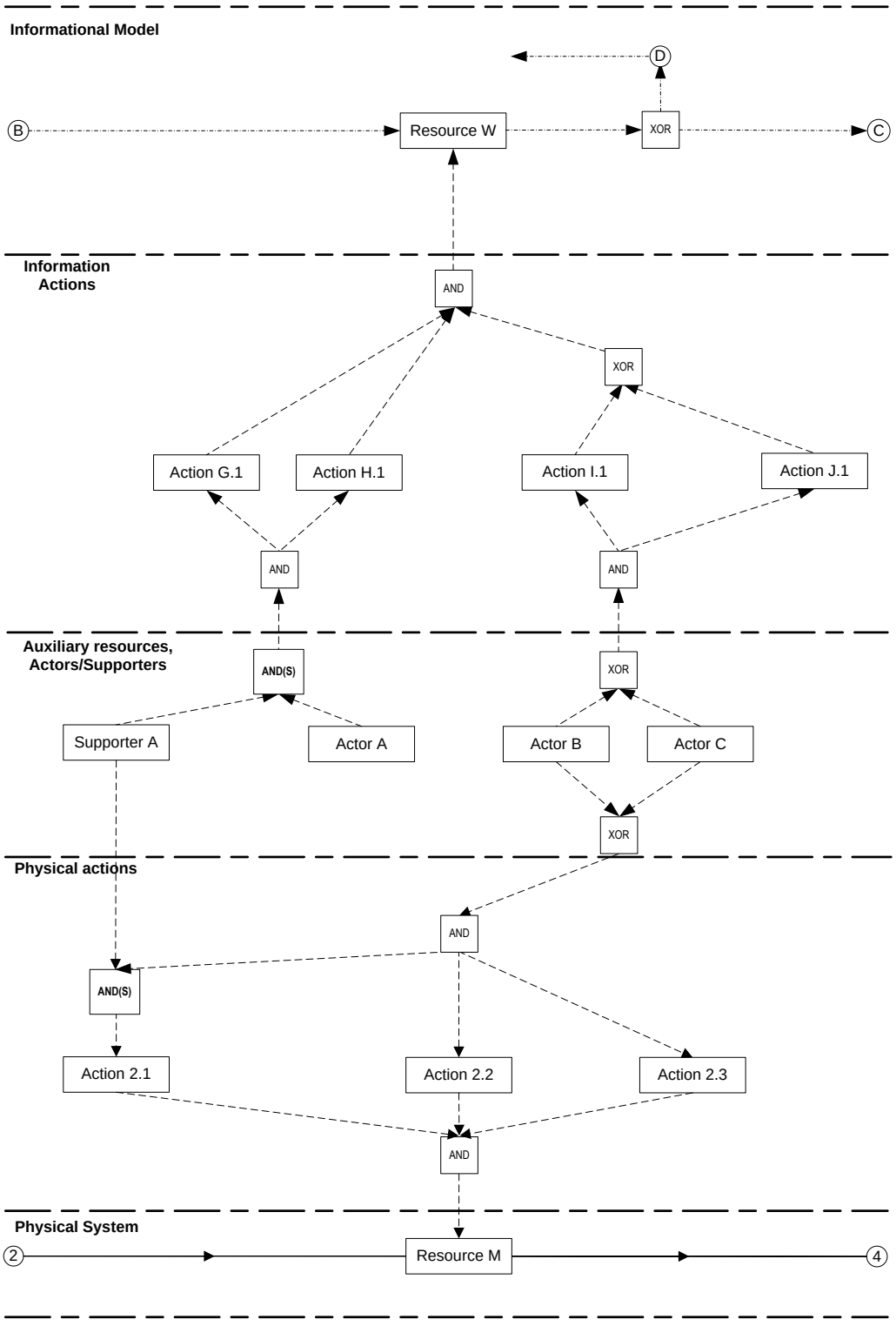

Figure 4.35 Extended SAD for Department B.

4.5 Differentiation of the SAD Technique from currently available techniques

The SAD technique that has been presented in this chapter has been developed specifically to support the requirements gathering phases and conceptual model development within a simulation project. In facilitating this requirement the technique represents both what a discrete process is and likewise, how a simulation model represents such a process. As discussed in chapter 2, there are various process modelling techniques available to a simulation model developer that can be used to aid in these pre-coding phases. The SAD technique has adopted certain aspects of a number of these techniques. Techniques such as Activity Cycle Diagrams (ACDs) and Petri Nets model a system as alternating phases of activity and waiting. Such a representation of a discrete event system is adopted in the SAD technique by the introduction of primary resource and queue elements. The SAD technique also adopts elements from within the IDEF modelling suite of tools, the IDEF 3 technique allows for the hierarchical structuring of a process model along with the use of branching elements, which have been adopted by the SAD technique. The SAD technique also adopts an approach of the Event Driven Process Chains (EDPCs) technique. These EDPCs allow for the development of a model of a discrete event system as a series of events that take place within such a system. The SAD technique adopts and extends this modelling approach by introducing the concept of an action list. The RAD approach of placing a role or the person or persons charged with a task or series of tasks centrally within the model is also adopted within the SAD technique. This is achieved through the separation of the resource into not just primary and auxiliary, but by also subdividing the auxiliary resources into actor and supporter resources, with the actor resource capable of representing a person's role within a SAD.

While such similarities exist within the SAD technique, the overall modelling approach is radically different. The SAD technique endeavours to model complex interactions such as those that take place within an actual detailed simulation

model of a real system. Again the SAD technique is designed to fulfil the design requirements as outlined in Chapter 1, page 7. Each of these requirements are represented within the SAD technique. Both the physical and informational flows within a discrete event system are modelled at either extremity of a SAD model as shown in Figure 4.35. Also modelled are the resources used in the execution of the various activities associated with the transitioning of both the physical and informational models through their various discrete states, again represented in Figure 4.35. In achieving these goals, the technique uses the various SAD modelling primitives to represent the various events that are listed in a simulation event list. To also represent more complex interactions, the SAD technique introduces the concept of an action list, which is used to represent detailed actions that collectively can make up any event within a simulation event list. Such a modelling approach allows for the modelling of a modern discrete event system and in turn a simulation model of the same. Finally the use of a structured text based elaboration within the SAD technique allows for the removal of any ambiguities that may arise within a complex model. Such an approach increases the user's access to the information and knowledge that would otherwise be lost in detailed simulation code. As a result of these modelling approaches the SAD technique uses a set of high level modelling primitives that are capable of representing complex discrete event systems. The modelling technique places a low modelling burden on the model developer while also promoting the capture, representation and communication of detailed information in a user friendly manner for models users.

4.6 Initial validation of the SAD Technique

The SAD technique introduced in this chapter initially underwent a paper based validation to determine its ability to represent discrete event systems. To achieve this a number of paper based models of discrete event systems were developed with a view to validating different aspects of the technique. The systems examined were:

- **A furnace area within a manufacturing facility**. This system was examined to ascertain the techniques ability to model complex resource interactions within a system;
- **A precision components manufacturer**. The examination of this system took the form of system interviews with a number of key personnel involved in the manufacturing process to ascertain the techniques ability to accurately represent this type of information;
- **A diamond cutter manufacturer**. This system was used to ascertain the techniques ability to model a production system;
- **A kanban system**. This system was examined to ascertain the techniques ability to model complex informational flows.

These systems are presented in more detail in chapter 5. On completion of this initial paper based validation the technique was felt to be robust enough to proceed to the development of a prototype software tool to further validate the technique.

4.7 Conclusions

The SAD modelling technique presented in this chapter was developed specifically to aid a simulation model developer in the requirements gathering phases of a simulation project. The technique was developed with a view to overcoming the shortfalls listed in chapter 2. As discussed in the previous section each of these shortfalls has been addressed within the technique presented. The flow of work and informational systems are both graphically represented as are the actions associated with the execution of these flows. The modelling of resources utilised in the execution of these actions are also graphically addressed within the technique. The resource elements are also further subdivided into primary, auxiliary actor and auxiliary supporter elements to facilitate the centring of the SAD model around the role of a person or object charged with the execution of a particular SAD. The use of a frame modelling

element also facilitates the development of a hierarchical model of any discrete event system under investigation. The technique also introduces a means of elaborating the graphical SAD representations in a simple text based format. This facilitates the communication of complex system issues that may not lend themselves to graphical representation within a model in a user friendly manner. The technique developed therefore attempts to overcome the shortfalls listed in chapter 2. As a result this technique may possibly be used as a process modelling technique to aid in the capture, representation and communication of complex discrete event information in the requirements gathering phase of a simulation project. The following chapter introduces a prototype software tool, Process Modelling for Simulation (PMS) developed to implement the SAD technique introduced in this chapter.

Chapter 5: Process Modelling for Simulation (PMS) Development

5.1 Introduction

The SAD process modelling technique has been designed to aid a simulation project developer in gathering discrete event system information and visually representing and communicating such information to non-simulation personnel involved in simulation projects. The previous chapter introduced the technique along with detailing how it overcomes the shortfalls listed in chapter 2. To aid in such an endeavour, a technique such as SADs needs to have an associated software tool to support its use. This chapter introduces the Process Modelling for Simulation (PMS) software prototype, based on the Simulation Activity Diagram (SAD) process modelling technique which was introduced in chapter 3. This chapter is divided into the following sections;

- **Software Development Platform.** This section outlines the Microsoft Foundation Class (MFC) application framework with which the PMS software was developed.
- **PMS Software Design.** In this section the design of the classes used in the implementation of the PMS software are discussed. An overview of the proposed operation of the PMS software is also given.
- **PMS Software Overview.** Here the PMS software will be introduced by stepping through the process of building a simple SAD model.

5.2 Software Development Platform

Prior to developing the PMS prototype a number of different development options were examined. Firstly, graphical flowcharting tools such as Micrographx and Visio were examined. Such an approach was not taken as the packages that were available at the start of the development phase did not have the embedded programming capabilities to allow the customisable changes necessary to develop a software prototype such as PMS.

The second type of tools examined were tools capable of real time monitoring of operating systems, such as GLG toolkit. Such applications were examined but not used for development of the PMS prototype as they were specialised for the monitoring and recording of real-time data on actual systems. The PMS prototype was not to be designed to mimic such situations, but rather for the gathering and visualisation of data from a variety of different sources as mentioned previously. As a result, such an approach was discounted in favour of developing the PMS prototype from a programming language, the languages of choice being Visual Basic and C++. C++ was chosen as the development language as it has been the development language of choice for object oriented software applications. It was felt that the PMS prototype implementation would benefit more from the object oriented aspects of C++ than the visual aspects of Visual Basic.

Having decided on the development language the next choice was an application framework within which to develop the prototype. The Microsoft Foundation Class (MFC) application framework was chosen for the PMS prototype. This application framework has been evolved by Microsoft as a C++ based programming interface for the development of Windows based applications.

The application framework can be considered as defining the skeleton of the application and supplies standard user-interface implementations that can be placed into the skeleton. The use of the class concept in C++ allows for the extension of the language by means of pre developed class libraries that can be delivered with C++ compilers or developed and sold by third party vendors.

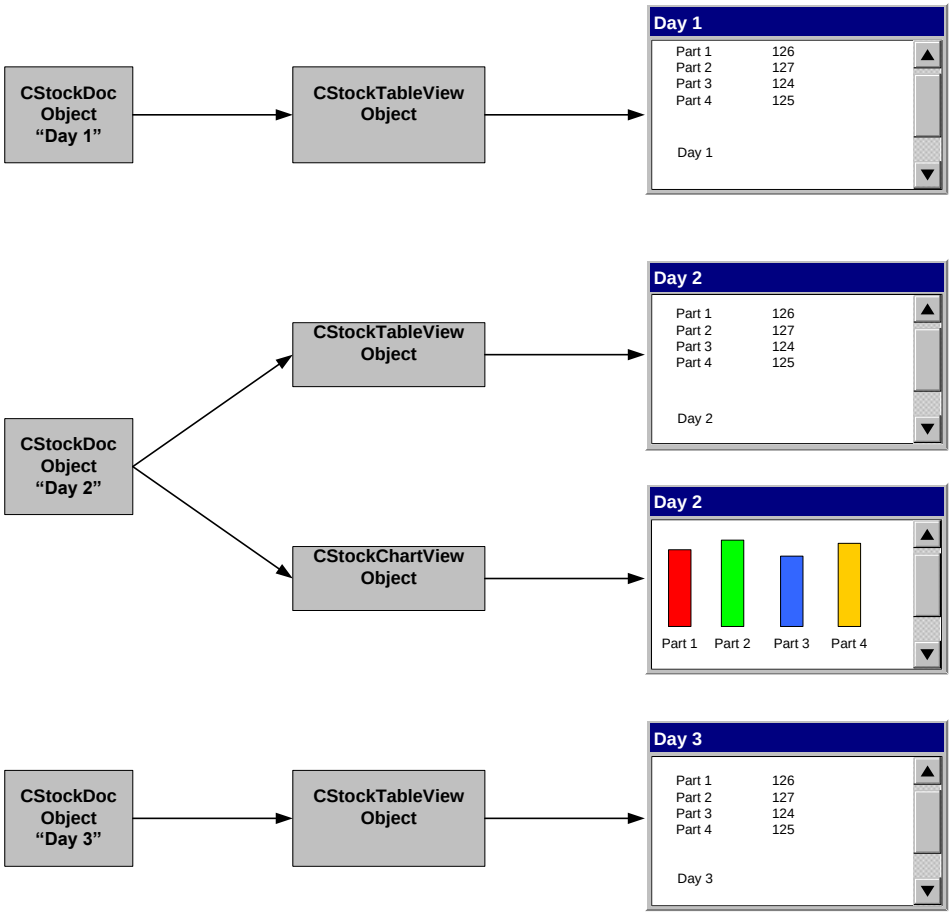

Figure 5.1 Documents and views in the MFC application framework

A typical MFC application will consist of an application and frame class plus two other classes that represent the "document" and the "view". This document/view architecture is the core of the application framework. This approach separates the data from the user's view of the data. A benefit of such an approach is multiple views of the same data. For example, consider a document containing the daily production quantities for a month. In this instance there are both a Table and a chart view of the data available to the user. The user updates the data from the Table view, but as a result the chart view is updated as both windows display the same information but in different views.

In the MFC library application, documents and views are represented by instances of C++ classes. In Figure 5.1 three objects of class CStockDoc corresponding to three days: 1, 2 and 3. All three documents have a Table view

attached and one document also has a chart view attached. Therefore there are three objects of class CStockTableView and one of class CStockChartView.

5.3 PMS Software Design

The prototype application called PMS has been developed to implement the SAD process modelling technique. The focus of the application has been to demonstrate how the SAD technique can be used to aid in the capture, representation and communication of discrete event system information in a high level, user friendly manner, so as to promote consensus building.

5.3.1 PMS Architecture

Figure 4.2 depicts the components of the PMS application.

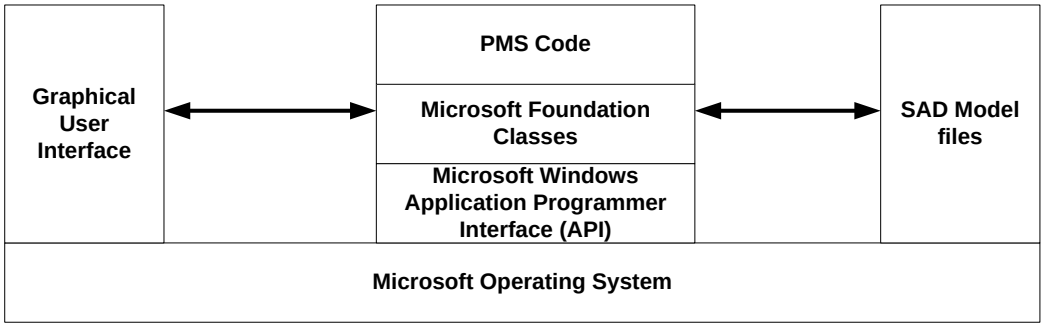

Figure 5.2 PMS High Level Architecture

The PMS development resulted in PMS code that employs Microsoft Foundation Classes, which wraps Microsoft Windows API, to provide various application features. The connections are usually in the form of Object Oriented (OO) Inheritance. Using OO Inheritance new C++ classes “get” the characteristics and capabilities of the class that they inherit and can add new characteristics, or “override” inherited characteristics and capabilities of the classes.

The PMS prototype defines several C++ classes that inherit properties and methods of Microsoft Foundation Classes (MFC) and services of Application Programmer Interfaces (API)s. These MFC Classes and APIs in turn interface

when compiled with the APIs and primitives of the Microsoft Operating System. In this way the software environment on which the PMS runs is created.

5.3.2 PMS Specific Code

The PMS specific code consists of:

- Classes that inherit from MFC classes;
- New PMS base classes;
- Coded logic to support the operator"s use of the SAD methodology.

Figure 5.3 outlines the sections of the MFC hierarchy chart which are used via inheritance in the PMS code as described previously. The points of inheritance are outlined in blue in Figure 5.3, for example CObject, and will appear as entry points or base objects in the PMS classes with inherited MFC Classes, Figure 5.5. Several MFC class branches relating to windows control classes are derived from CWnd such as CComboBox and CToolBarCtrl. As mentioned previously these classes or branches inherit the properties of the class that they are derived from.

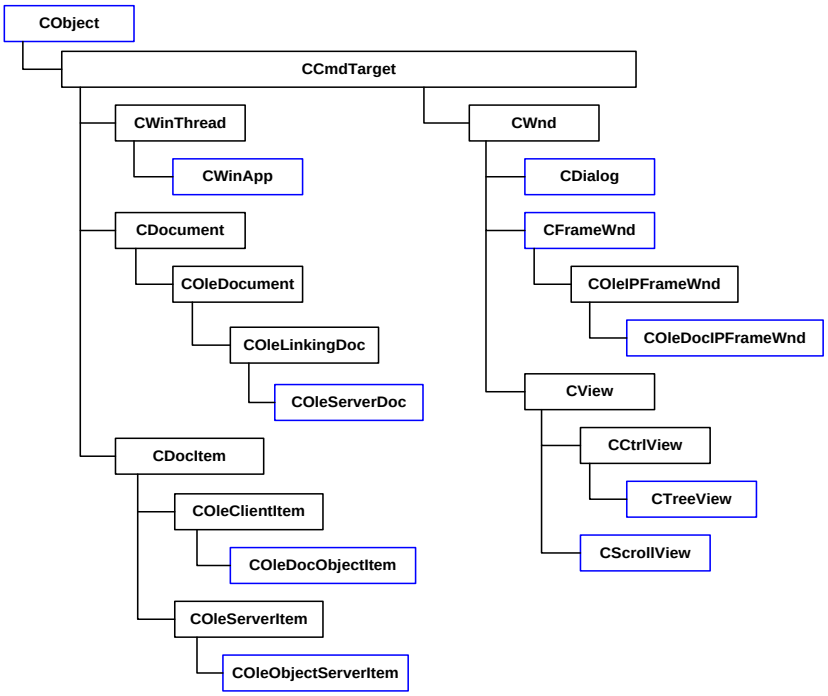

Figure 5.3 Partial MFC Hierarchy Chart – Inheritance

Other classes and interfaces that have been omitted from this section for brevity and due to automation within the application include data exchange mechanisms that automate the transfers of data between the PMS code and the GUI, and between the PMS code and the SADs that are saved to, and loaded from, files. The final classes and interfaces omitted here are those that form part of general "good" coding practice such as exception, tracing and debugging support code.

Figure 5.4 shows another partial MFC chart that outlines the major MFC classes that are directly instantiated within the PMS application.

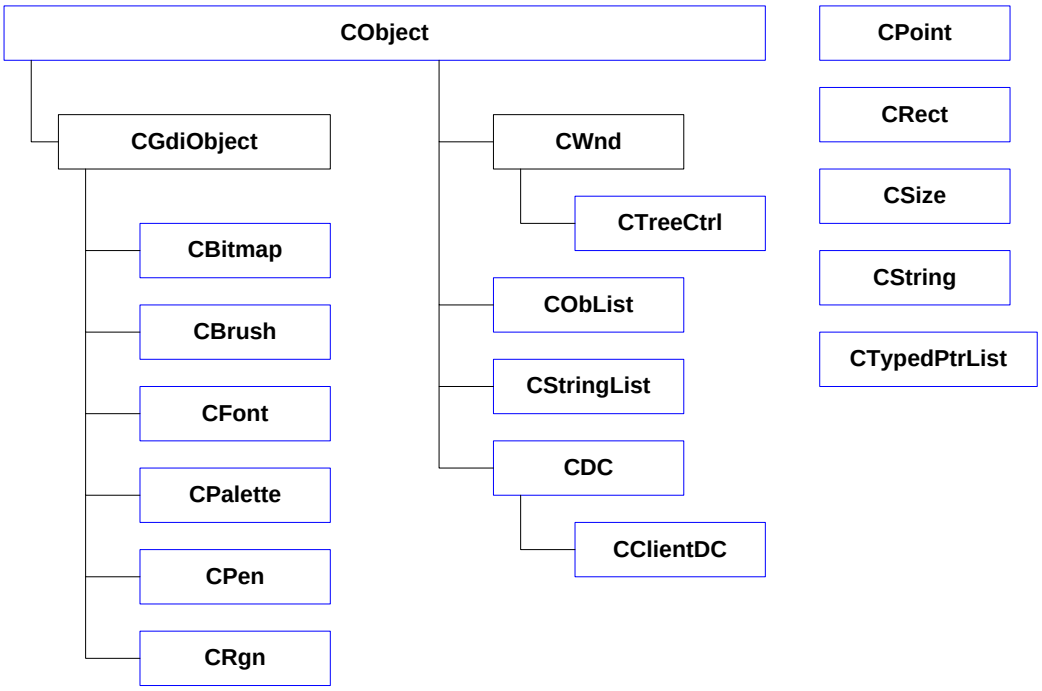

Figure 5.4. Partial MFC Hierarchy Chart - Instantiation

Figure 5.5 shows the PMS classes (outlined in orange) and the MFC classes (outlined in blue) from which they were derived. The exception is CDrawTool, which is not derived from a MFC class.

The CPMSApp is the MS Windows Application entry point to the PMS code. CPMSDoc and CPMSView implement the well-accepted Document-View Architecture for GUI applications. COrgView and CMainFrame implement the two main user panels of the PMS GUI.

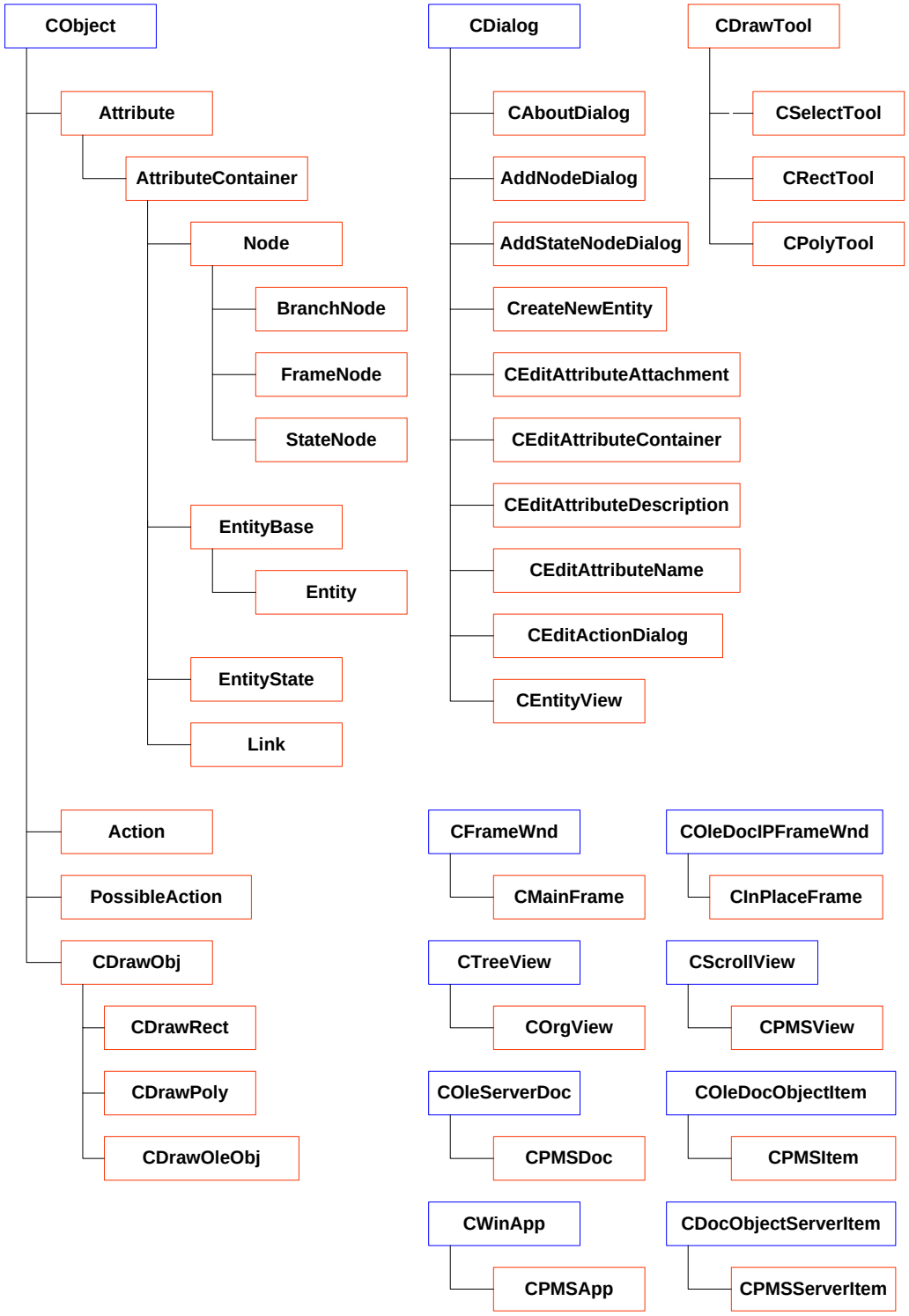

Figure 5.5 PMS Classes with inherited MFC Classes

All the classes derived from CDialog and those derived from CDrawTool are used to manage input from the user. “Attribute” and all its derived classes are used to manage the internal representation of the SAD methodology that has

been entered by the user. CDrawObj and its derived classes are used to hold the image primitives that will be output to the GUI as part of the Framework. It is this PMS software prototype design that is introduced in the following section.

5.3.3 PMS Software Design Overview

The PMS software operates as follows. A model developer will initially define the various entity and information elements to be modelled. Each of these elements will have the facility available for the definition of the various states, each will transition through during its various stages within the discrete event system. The entity element will also have the option available for the model developer to define family members for an entity. A model developer having defined the various elements to be modelled will then be able to develop the actual SAD models around the various transitions through the predefined states. In this way the state elements, both informational and entity, will form the initial and exit state for each SAD diagram within a model. The states will also be used at various levels of SAD models, with the entry and exit states for each frame being similar to the initiating and exit states internally within the same frame.

To develop each SAD model the model developer will have access to all of the modelling elements introduced in chapter 3. Each of these elements will be represented graphically within the PMS modelling tool. On the creation of an instance of such a graphical element, the model developer will initially be presented with a dialog box into which can be added a name and description of the element. This dialog will also give the option of the model developer creating a new instance of an element or linking the element to a previously created element. This will allow for the use of elements that have multiple occurrences within a model. On the creation of a branch element the model developer will be given the additional option of choosing the type of branch element to be created, similar to those introduced in chapter 3. Each graphical element will have access to a properties dialog box where a description and attachments may be added to describe aspects of interest related to such elements. Within this dialog there will also be a facility for a model developer to add attributes related to elements. This

will be to allow for the collection of particular information relating to certain attributes of a particular element within a model. Having created the initial model elements the model developer can then add the various link types to the model. To achieve this, the model developer will add an action, to allow for the modelling of an action list as introduced in chapter 3. This action will be initiated from one of the states created within a particular model, be they informational or entity. On the initiation of such an action, the model developer will have the option of adding the various link types entity, activity, or informational to a particular SAD model. On the completion of an action each SAD will give the model developer the option of viewing the elaboration text of the particular SAD model on view. This option of viewing the elaboration text will be accessed from the operator or supporter auxiliary resource element, thus, placing the role of this element or person centrally within the overall model. In this way facilitating communication of operational issues, to the individual whose role may be modelled at a particular instance. This elaboration text will have a text based description of the graphical representation along with a description of any of the elements and details of attachments and attributes attached to the elements along with the option for the model developer to access the attached items. From this elaboration the model developer will have the option to step through the particular SAD model on display while at the same time, being stepped through the graphical SAD model on display. There will be a number of step through options available to the model developer, these being conditional or user defined. The above section describes the overall design objectives for the PMS tool. However, to date this functionality has not been fully implemented within the software. The following section outlines what has been developed and how this can be used to develop a SAD model.

5.4 PMS Software Overview/SAD model development process

The following section steps through the process of developing a SAD model within the prototype PMS modelling application. The process outlined here is the

process by which a model developer might go about developing a model during the requirements gathering phase of a simulation project. Figure 5.6 shows the start up screen of the PMS modelling environment.

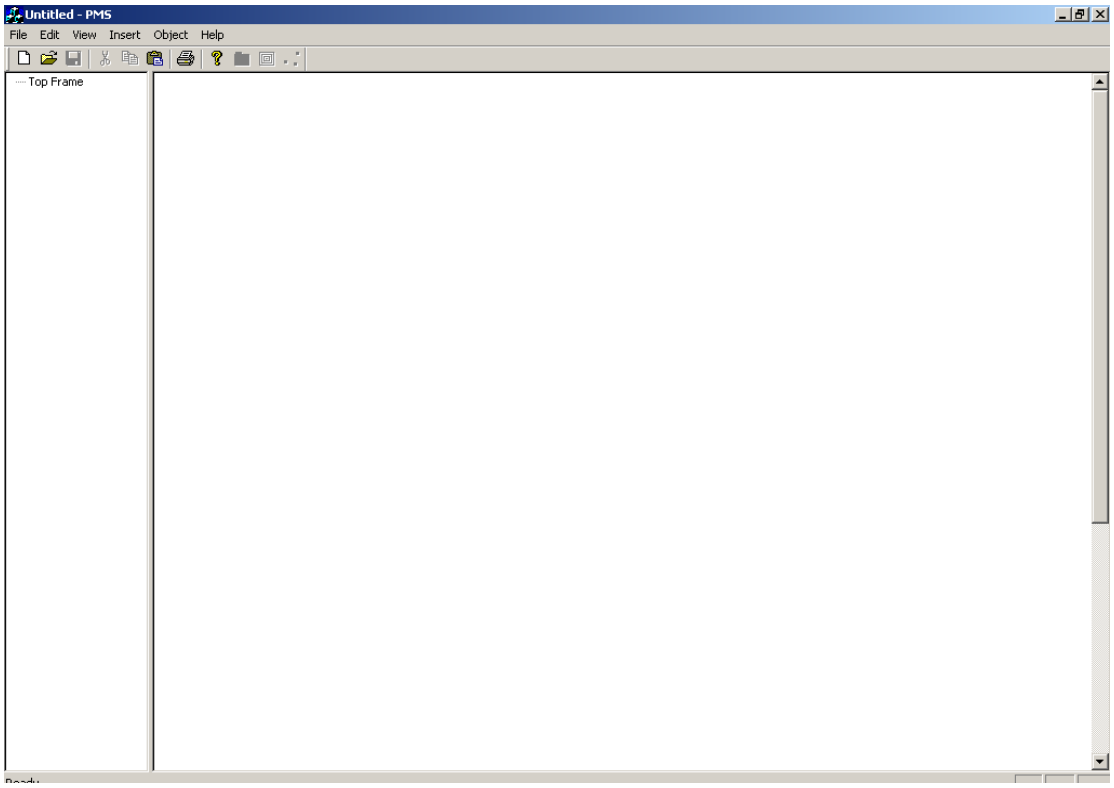

Figure 5.6 PMS Modelling environment start screen

From here the user has to create the various entities that are to be used in the model. To do this they chose the entities option in the view menu as shown in Figure 5.7.

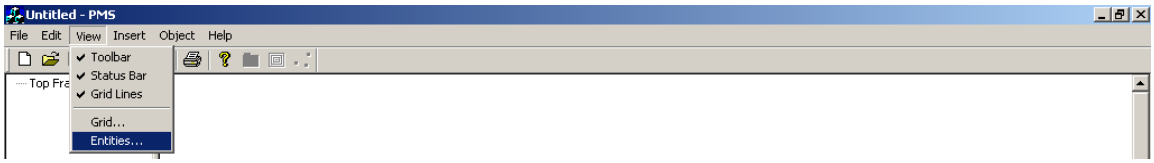

Figure 5.7 PMS option to create entities

Having chosen this option the dialog box shown in Figure 5.8 is displayed. This dialog allows a user to create both entity and information elements that are represented visually by the information and entity state elements in the PMS environment. Here the user can create new instances of entity and information elements along with creating the various states for each. The user also has an option here to create families of entity and informational elements. This can often

be the case in a simulation model where an entity is used to represent a number of similar elements within a discrete event system.

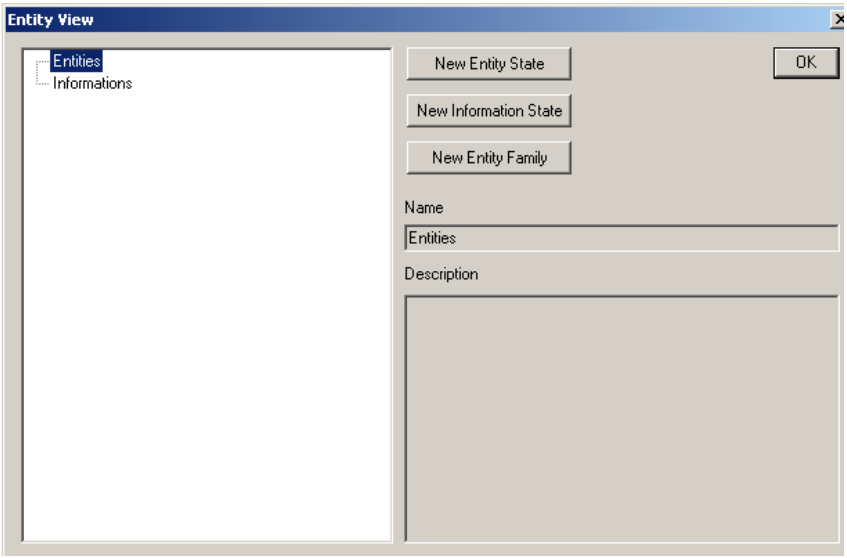

Figure 5.8 Entities or Information creation options

Figure 5.9 shows the same dialog box with the entity and information elements that are being modelled in this instance, along with their various states being displayed.

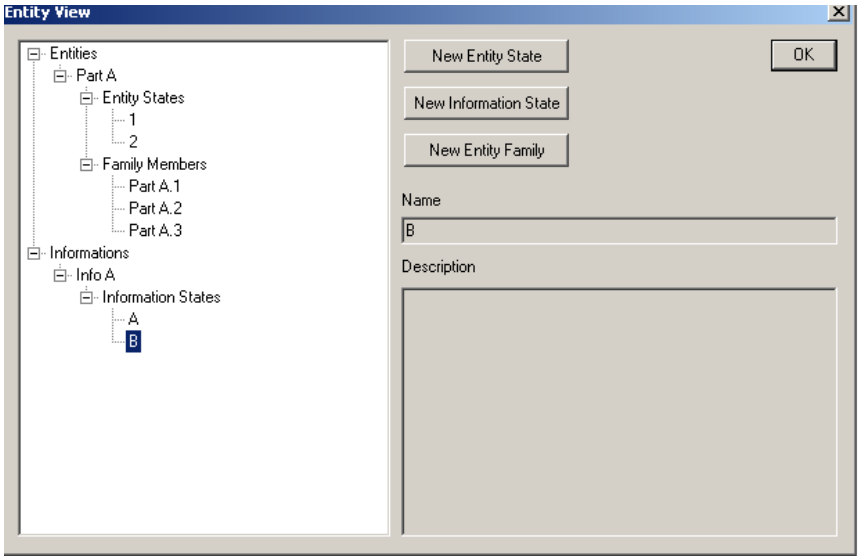

Figure 5.9 Entities with various states created

The user can at any stage of developing a SAD model return to this entity screen and edit, delete or create any entity elements. Having created the various elements and their states that will be modelled within the PMS modelling application, the user can now proceed to the main model building area to develop

the various SAD models. To achieve this the user moves to the insert drop down menu, here the user has the option to enter the various modelling elements as shown in Figure 5.10.

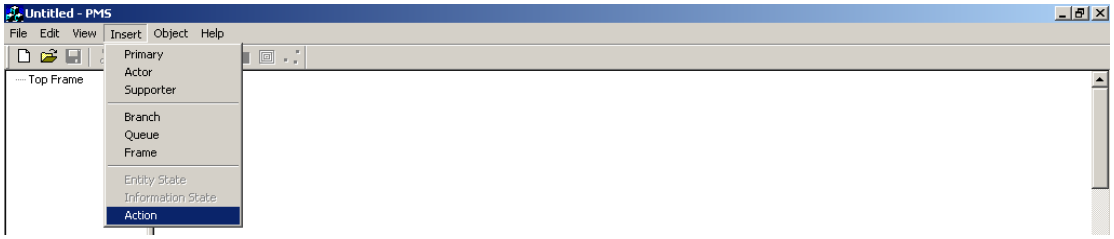

Figure 5.10 Modelling elements that can be added to build the model.

The dialog box shown in Figure 5.11 is a standard dialog box for the creation of any of the following elements: a primary, auxiliary or operator resource, queue, frame and action element.

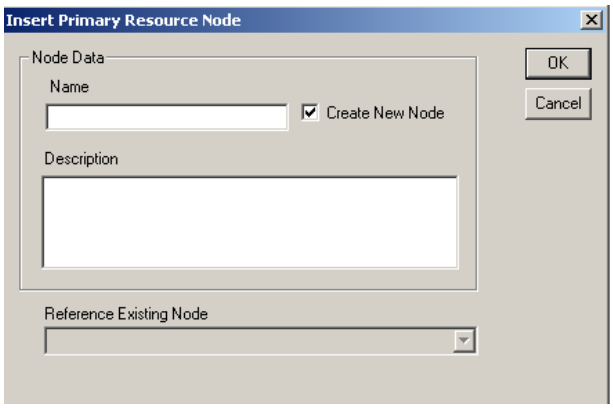

Figure 5.11 A standard details dialog box for the addition of a modelling element.

The user is asked to enter the name and description of the element being created along with whether or not the element is a new instance or referencing a previously created element.

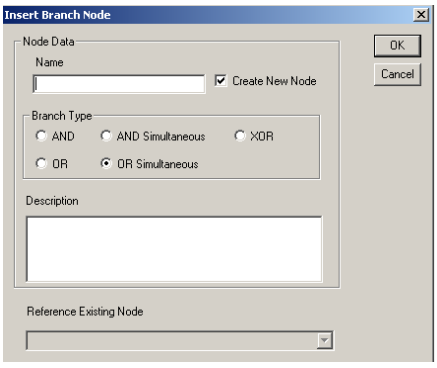

Figure 5.12 A standard details dialog box for the addition of a branch modelling element

The only dialog that differs from this is the branch dialog, shown in Figure 5.12. In this dialog the user is asked to enter both the name and description as previous and also to choose the branch type to be created.

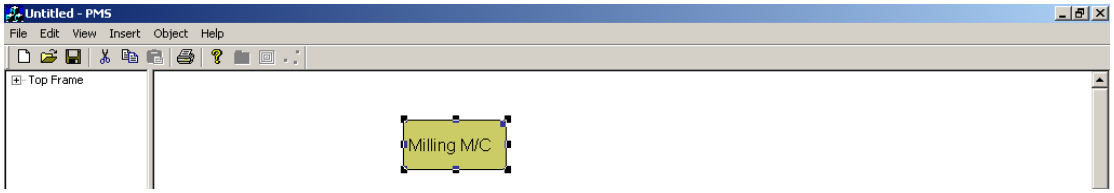

Figure 5.13 A Primary resource element for a Milling M/C

As mentioned above Figure 5.11 shows the standard dialog box, in this case the dialog box represents a primary resource element and this element is shown after its creation in Figure 5.13. The addition of other elements follows a similar pattern to that of the primary resource element. The only elements that differ in this procedure are the information and entity state elements. These elements have been created previously and are therefore entered into this section of the model be simply picking from drop down lists as shown in Figure 5.14.

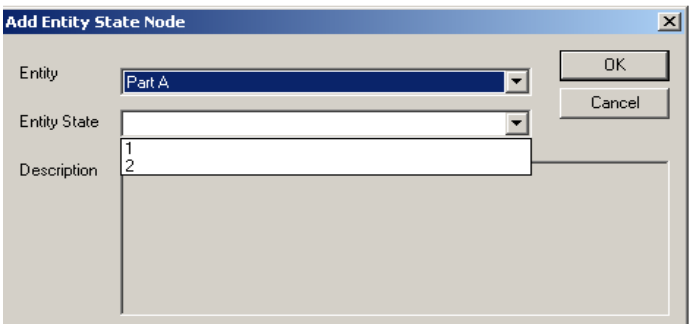

Figure 5.14 A standard dialog box for the addition of an entity or information state.

In this instance the user is asked to choose one of the previously created entities from a drop down list and then to choose a particular entity state for the entity. The same procedure is followed for the addition of an information state element. From this point the model developer iterates through the creation process for the various elements within the model. Figure 5.15 shows the elements entered for a simple SAD diagram in the PMS tool.

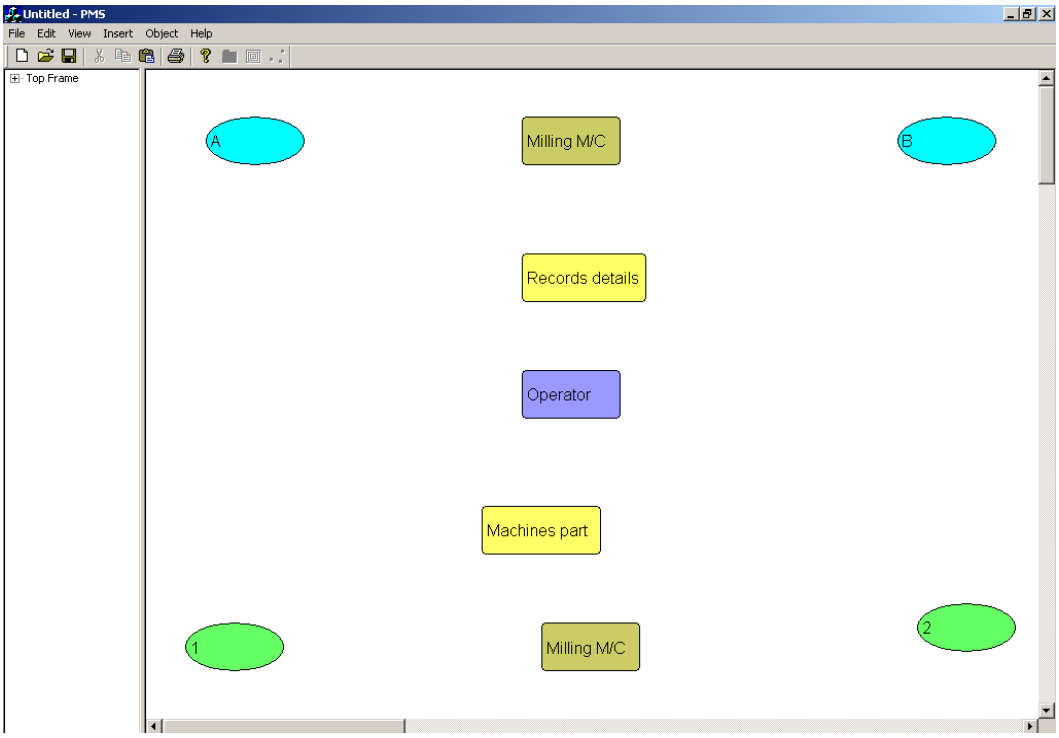

Figure 5.15 Elements for a simple SAD diagram

Having created such a model the next step for a model developer is to create the various links between the modelling elements.

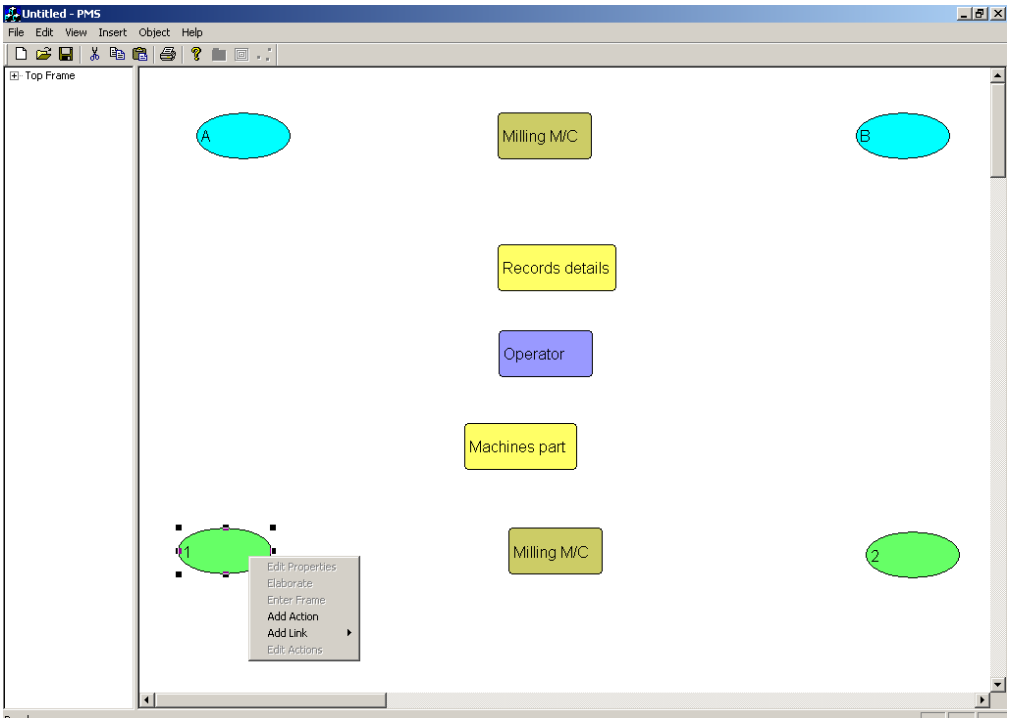

Figure 5.16 SAD model popup menu

To achieve this, a model developer can use the right click button on a mouse with the mouse pointer on either a entity state, an information state or a branch element. As shown in Figure 5.16 a popup menu with a number of options becomes available to the user. The option of interest here is “Add Action”, this is to mirror the concept of an action list as introduced in chapter 3. On choosing this option to add an action, a model developer can add various link types as required. Figure 5.17 shows the same model again, in this instance when the user right clicks on the mouse button with the mouse pointer over any element he/she has the option of adding the links as shown in the popup menu.

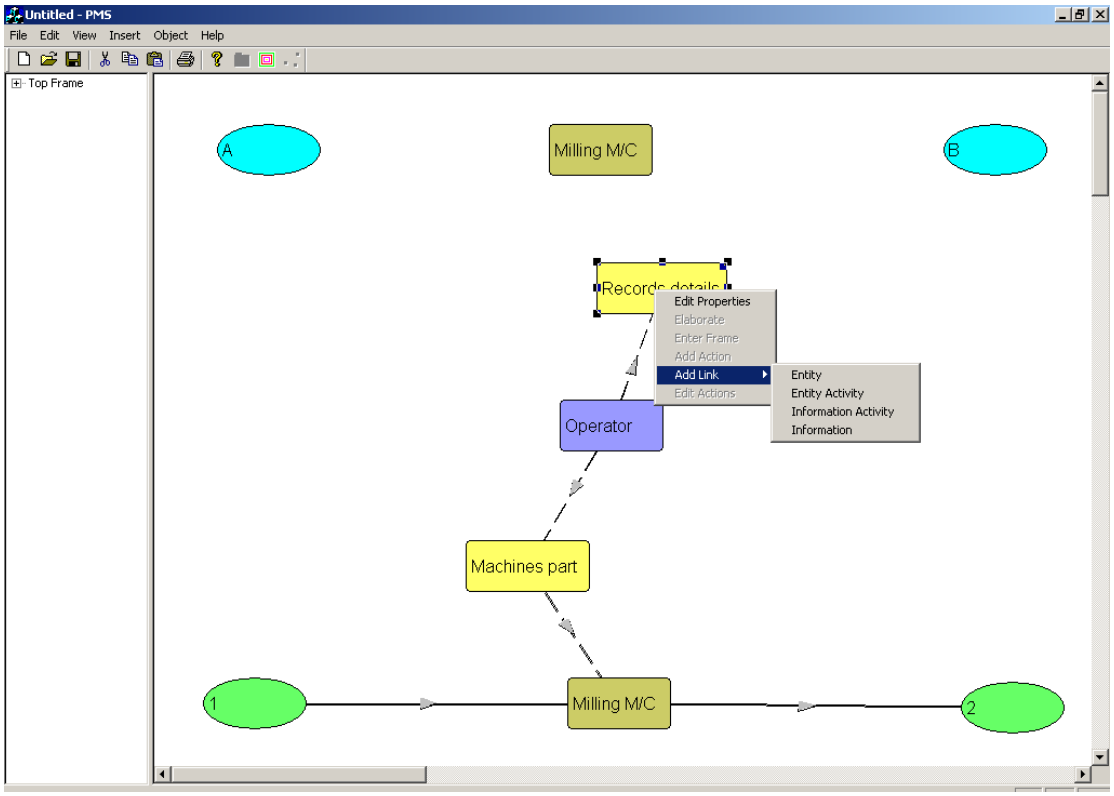

Figure 5.17 Adding links and the add link popup menu

From here the model developer adds the various model links as required. From here the model developer can now add the various data that may be used in support of any aspects of the model. To do this the model developer has access to an edit dialog box through the right click popup menu as introduced previously.

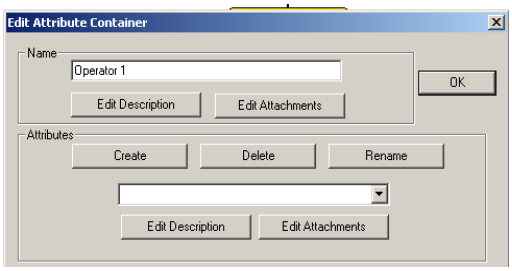

Figure 5.18 Edit properties dialog box

This edit properties dialog box is shown in Figure 5.18. As can be seen, there are a number of different options available to the model developer from here. The model developer has the option to edit the description of any element by means of this dialog. This edit description will take a description of any element to allow a model developer elaborate on the element if necessary. This edit description dialog is shown in Figure 5.19, with a description for this element entered.

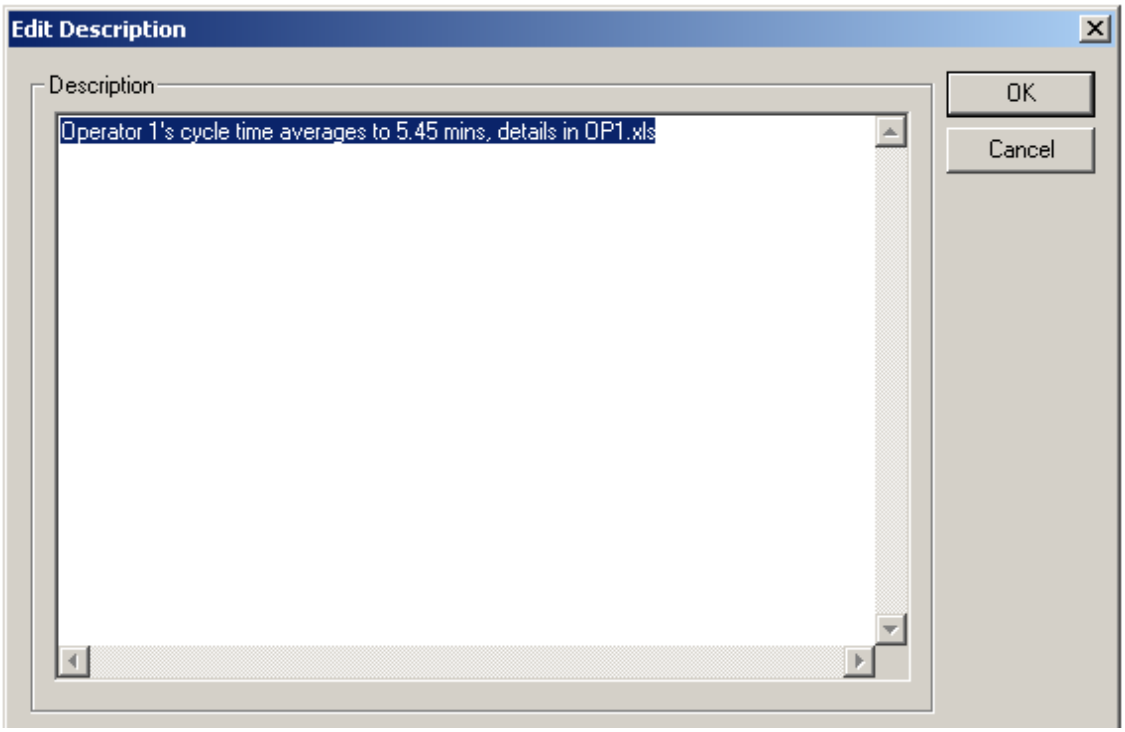

Figure 5.19 Edit Description dialog box

From the same dialog shown in Figure 5.18 the model developer also has the option to edit or attach a number of attachments. Such attachments can take the form of MS Excel, Word or Access documents. When the model developer chooses to attach such a document, the dialog shown in Figure 5.20 is

presented. From here the model developer can choose to attach a file to the chosen element. If the model developer chooses to do so, the standard open dialog shown in Figure 5.21 is presented.

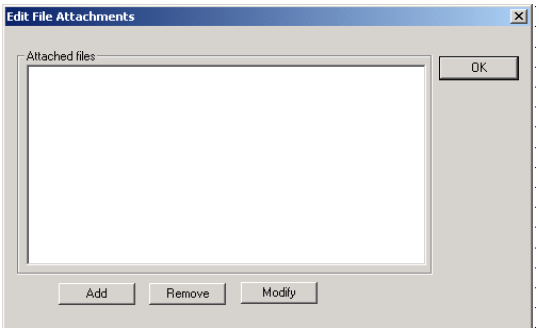

Figure 5.20 Attach document dialog

From Figure 5.21 the model developer chooses a file to attach to the element.

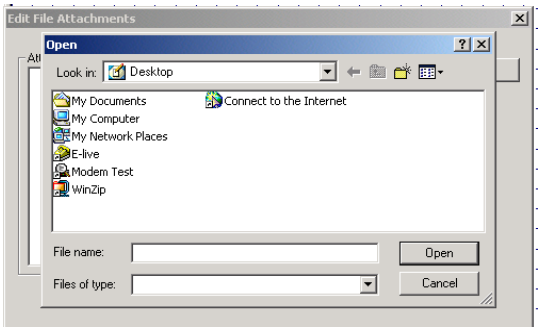

Figure 5.21 open dialog

On completing this action the file is attached to the element. Such a scenario is shown in Figure 5.22, where a document named Skills is displayed.

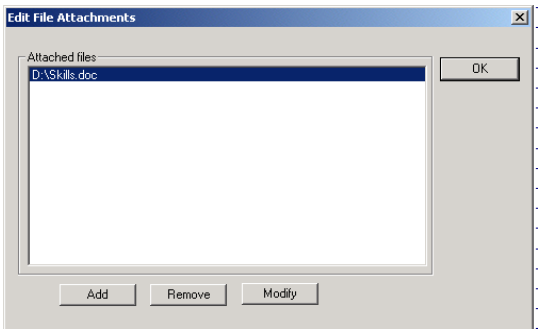

Figure 5.22 Attached document added to the attach dialog

From here the model developer can either continue to add attachments as required or return to the edit properties dialog, Figure 5.18, from where specific attributes can be created for each element for which information needs to be

recorded. If a model developer chooses to create an attribute for a particular element the first dialog that will be encountered is shown in Figure 5.23.

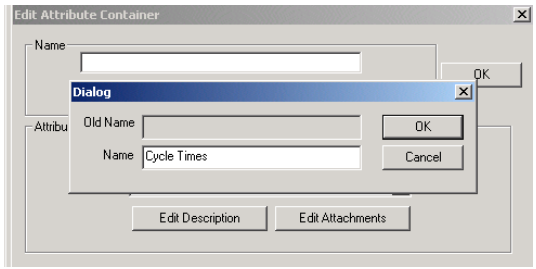

Figure 5.23 Create/Edit attribute dialog

Here a new attribute called “Cycle Times” is being created for a particular element. Having created this attribute the model developer can now add a description and attach documents recording information on such an attribute.

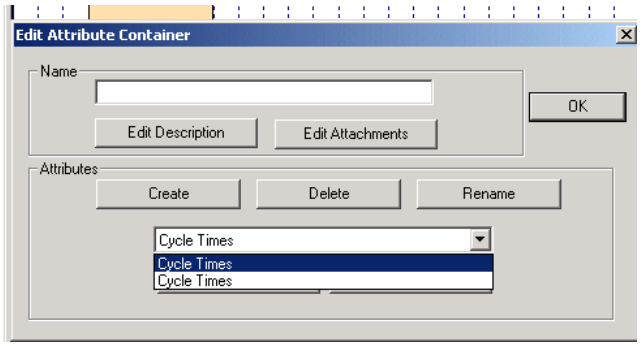

Figure 5.24 Attribute selection drop down list

To do this the model developer selects the attribute of choice from the attribute selection drop down list as shown in Figure 5.24. Having chosen an attribute the user can use the edit description and edit attachments within the attributes area of the edit properties dialog as shown in Figure 5.18. On the completion of the SAD model within the PMS tool a model developer can then access the elaboration text for a SAD. To access this the model developer right clicks on the operator element within a SAD as shown in Figure 5.25.

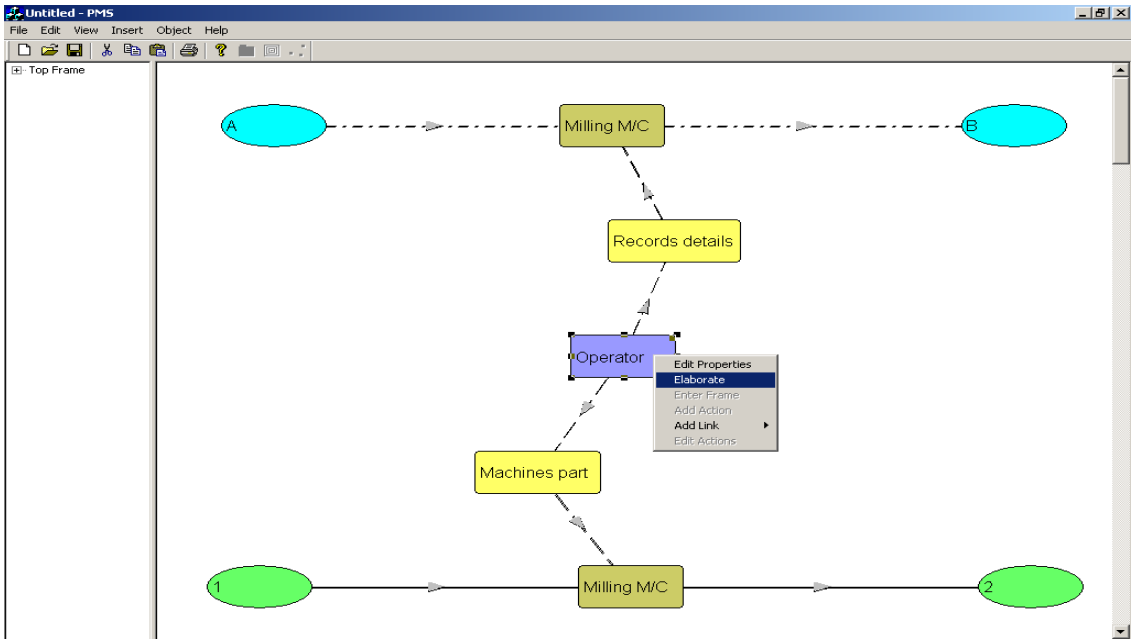

Figure 5.25 Elaborate function

On accessing this function the model developer has access to the elaboration similar to that shown in Figure 5.26

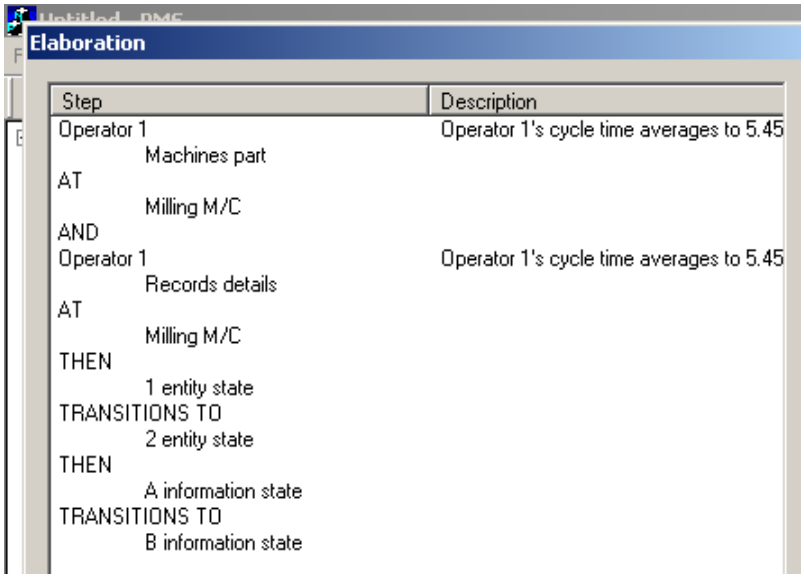

Figure 5.26 PMS Elaboration

Within this elaboration the user has access to a simple text based elaboration along with descriptions on any of the elements created, to further explain their usage. This elaboration text is automatically generated within the PMS software. This is achieved by means of the Elaborate function outlined above. This function

is executed within the PMS tool by the formation, sorting and condensing of a navigation list within the modelling environment. The methods used for the execution of this elaboration are listed in order and briefly outlined below.

Navigate Node

Searches the currently displayed SAD Frame model for all supporter and auxiliary resource nodes.

For each of these nodes a navigation thread through the model is formed.

These threads are forward navigated only and will generally result in many partial path duplications.

Sort Primary Threads

This method sorts the first primary thread nodes (Actor and Supporter nodes) such that each of these nodes and associated threads are in the same left to right order as in the visual model.

This method also recursively navigates each thread to sort list members into the same left to right order according to the visual placement in the actual SAD model.

Sort List

Each thread is now recursively searched for more lists on each of which the same sorting process is performed. As a result the navigation list is made up of navigation objects which are sorted in the same left to right order as in the visual model.

Condense Primary Threads

This method is used to remove duplications within the primary threads.

Condense list

This method is used to remove duplication across the primary threads within a navigation list.

This results in a navigation list with no duplications.

However there may be several single navigation object threads where two or more activity nodes have the same “Parent” fan out branch and the same “Child” fan in branch

Combine Branch Threads

To overcome the issue of single navigation object threads the Combine Branch Threads method is used. Each navigation object has a tag thread member and in this situation the second and other navigation objects in a group of singletons are added to the first tag thread list of the first navigation object in the group and removed as threads from the list. This is only carried out for action nodes.

Elaborate

The elaboration method navigates through the navigation list and creates an elaboration object for each navigation object that remains. Each elaboration object is stored in a list and contains the simple language primitives and / or navigation objects. It is the contents of these elaboration objects that are displayed in the elaboration window in the PMS software.

5.5 PMS Hierarchical Modelling

The PMS modelling tool also allows for the model developer to create a hierarchical model as required. This is achieved using the frame element as introduced in chapter 3. Each of these frame elements is capable of being subdivided into lower level models. To achieve this, the user simply double clicks on the frame element. This gives the user access to the lower level model. A frame element is shown in Figure 5.27.

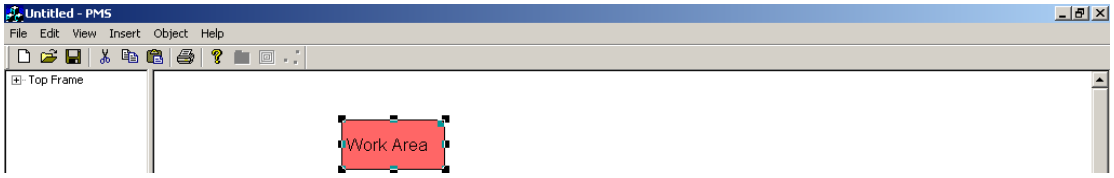

Figure 5.27 A Frame element.

The user can then migrate to lower levels by double clicking on other frame elements within sub models. At any stage the user can also migrate up through model levels by using the migrate upwards button shown in Figure 5.28.

Figure 5.28 Migrate upwards button

Therefore any frame element can be used to develop a lower level model of a certain part or area of a higher level model. Figure 5.29 shows a section of a high level SAD model.

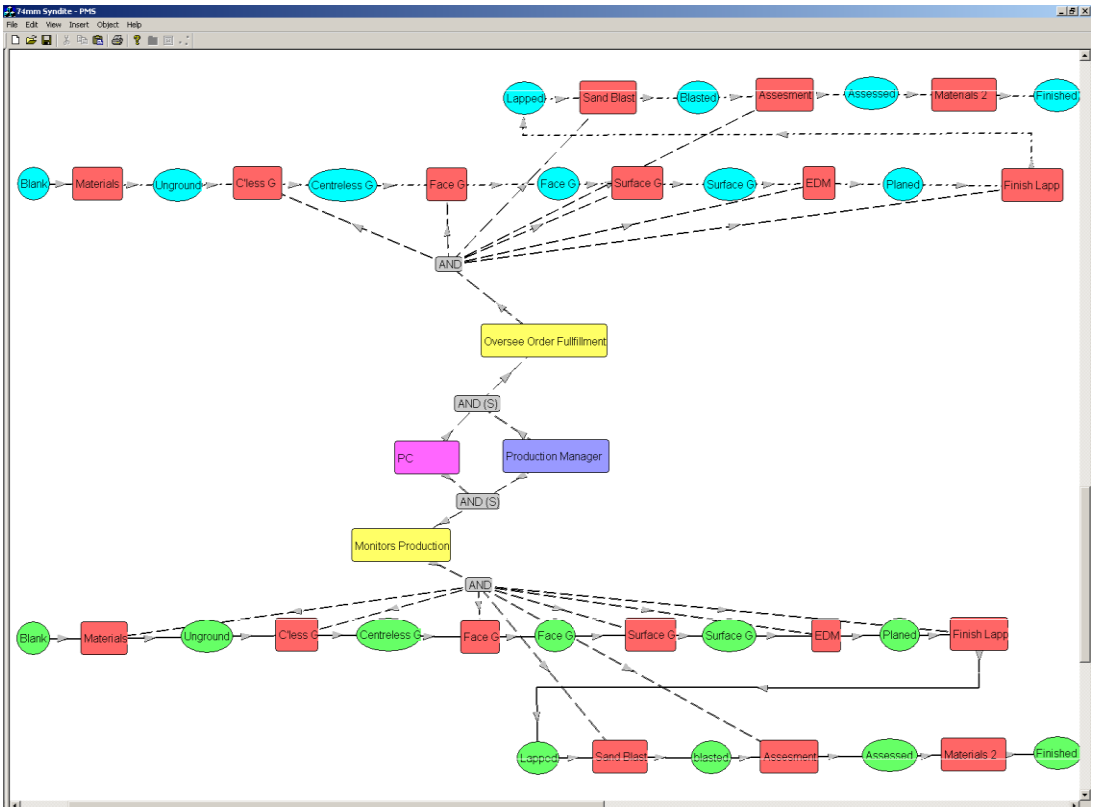

Figure 5.29 A High level SAD diagram containing frame elements

Each of these frame elements can then be used as a sub model to graphically represent more complex interactions related to a particular area. Figure 5.30 shows a SAD diagram contained within the “Materials” frame shown in Figure 5.29.

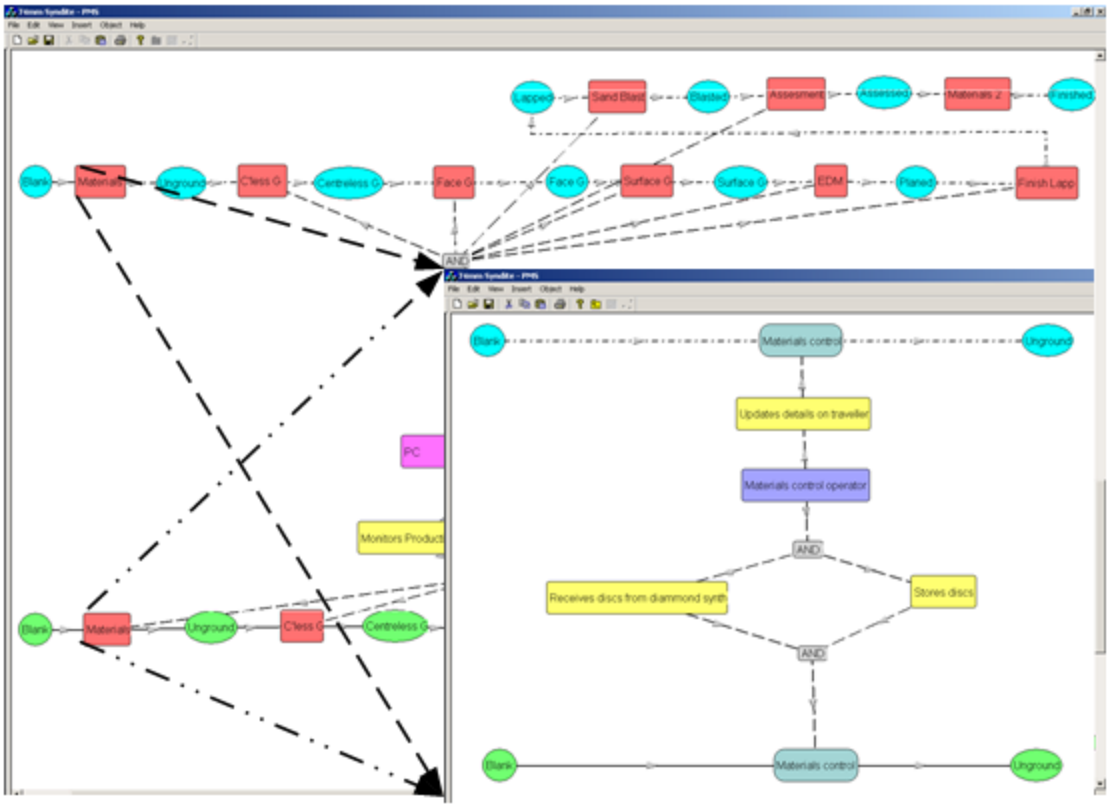

Figure 5.30 Sub model of a discrete event system contained within a Frame element

In this way a model developer can develop a hierarchical model of a discrete event system as required.

5.6 Proposed usage of the SAD technique/PMS Tool

Figure 5.31 shows the current support offered by the SAD technique. Its current sphere of usage along with proposed extensions to this sphere will be discussed in the following section. The SAD technique and PMS tool can currently be used to support a simulation model developer during the requirements gathering phase of a simulation project. As can be seen from Figure 5.31 such a phase would involve discussions with systems personnel on the requirements and the model being developed. To this end the PMS tool combines the high level semantics of the SAD technique with the automatic generation of a high level textual language to support communication and understanding between the model developer and systems personnel. A further enhancement to this will be

the step through facility, which will explicitly link the textual language and the SAD model to further support communication and understanding.

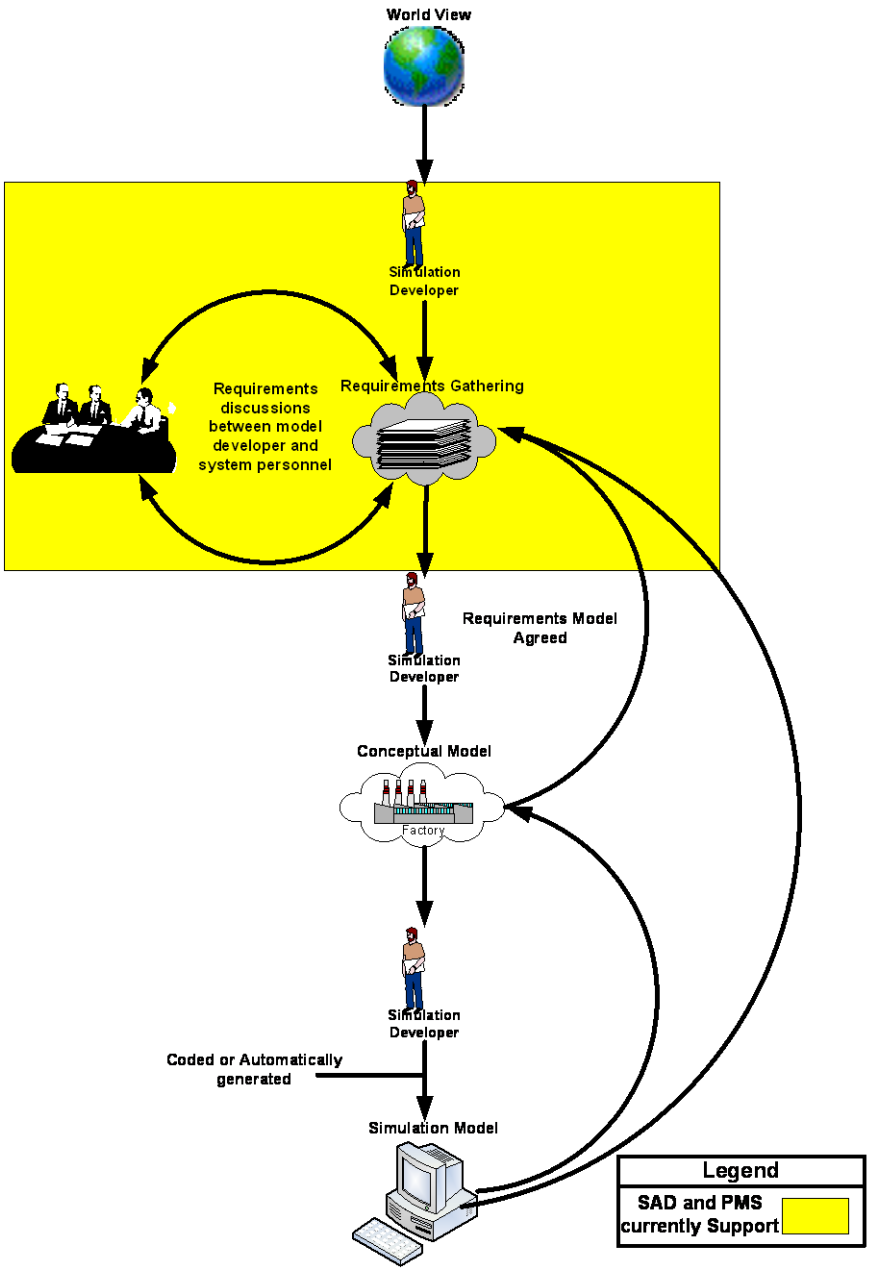

Figure 5.31 SAD and PMS Current sphere of usage

However as can be seen from Figure 5.31 while the requirements gathering phase of a simulation project is supported currently the conceptual modelling phase, which is the next phase in the progressing of a simulation project is not. To facilitate the support of this phase of a simulation project it is proposed to

develop a versioning module within the PMS tool. Such a versioning module would allow for the requirements model to be reduced or versioned within a separate screen thus allowing for the conceptual model to be developed, while still being explicitly linked to the requirements model. The explicit linking of the requirements model and conceptual model in this way would further support communication and understanding of the overall simulation model being developed as the conceptual model developed would be used to form the basis of the simulation model as shown in Figure 5.31.

5.7 Discussion

The PMS prototype software outlined in this chapter was developed to introduce the concept of a software tool capable of developing and supporting SAD models. While there are many means of developing such prototypes the means chosen in this case was C++ due to its highly customisable ability, thus giving the programmer a fully customisable programming platform. The software prototype concentrated on developing an implementation capable of representing the various modelling elements of the SAD technique. Therefore, the PMS prototype is capable of hierarchically developing a highly visual model of a discrete event system, which is capable of communicating detailed system issues. This is achieved by means of the SAD modelling technique, which firstly allows a user to create and document the various elements within a system, both physical and informational, along with their various intermediate states of transition. Having developed such a SAD model of a discrete event system a user is also given the option of further developing the model by creating attributes for any element within the PMS tool and describing such attributes by means of descriptions and attached documents. The user is given the option of viewing the SAD elaboration language to further highlight system logic and dispel any ambiguities that may arise in the modelling of complex system information. In this way the PMS software allows a model developer to build a model that can be used as a means of visually capturing, representing and communicating discrete event system

information. Such models are capable of being used as an aid to a simulation model developer in the requirements gathering phase of a simulation project, when such information is gathered for the purpose of gaining enough detailed understanding of a particular system to develop a conceptual model and ultimately a simulation model of the same system. Such models can aid this process by giving a model developer a structured means of gathering information in a highly visual and communicative manner. Finally such a model could be ultimately used as a repository for discrete event system information gathered during the development of a simulation model, in a manner that can make it accessible to persons other than the simulation model developer. For example, it is envisaged that SAD models could be used to support continuous improvement projects within the manufacturing domain. To date the SAD technique and PMS prototype are not capable of fully supporting all pre-coding phases of a simulation project, however the current sphere of coverage of the technique and prototype along with proposed extensions to expand their support to other pre-coding phases are discussed.

5.8 Conclusions

The developed PMS prototype allows a user to create a SAD model capable of accurately representing a discrete event system in a highly visual manner. By using the SAD modelling elements a user can create a SAD model. Such a model can then have various information added, by means of dialog boxes, to aid in the representation and communication of system issues. These dialogs allow a model developer to describe various modelling elements in the context of a particular SAD model, and attach files to support such descriptions. Various attributes can also be defined for each element and in a similar way to the elements themselves, any attribute created can have a description and any number of attachments added. The PMS prototype also allows for the elaboration of SAD models using a simple structured text to aid a model developer in

communicating discrete event system logic in a user friendly manner, to persons who may not be familiar with the inner workings of a simulation model. In this way the PMS software is also capable of allowing non-simulation experts access to detailed discrete event system information that may otherwise be lost in the inner workings and low level code of a simulation modelThe following chapter is used to introduce a number of examples developed in the initial validation of SAD models.

Chapter 6: Validation of the SAD Technique

6.1 Introduction

As was outlined in chapter 3 an initial paper based validation of the SAD technique was conducted and based on the satisfactory outcome of this the development of the PMS software was carried out. The outcome of this development process was outlined in chapter 4. These validation tests were conducted to determine the technique's ability to accurately model and communicate various aspects of discrete systems and their associated information. The examples presented in this chapter, which were briefly outlined in chapter 3, have been implemented in the PMS modelling software tool as introduced in chapter 4. The scenarios which will be examined are as follows:

- A production system taken from the perspective of the operators (system owners) manning the line, In this scenario interviews are used to gain familiarity with the process, the SAD technique is used to model the scenario based on the outcome of a series of such interviews;
- A theoretical production system with a Kanban control system. The flow of information to control the system flows in the opposite direction to the flow of production;
- A batch flow-shop type production system where the operators have a lot of decision making power in relation to the advancement of the system and the types of parts that are produced at a given time;
- An overall production line used in the manufacture of a number of different products.

Each of the scenarios outlined above will be modelled using the SAD modelling elements, Figure 6.1, that were introduced in chapter 3. Along with these modelling elements the SAD elaboration language will be used to aid in the communication of operational information. The chapter then concludes with a conclusions section.

6.2 Overview of a precision component manufacturing system

The system outlined in this section is based on the results of a series of system interviews which were conducted with a number of workers in a precision component manufacturing facility in Galway, Ireland. In the early stages of any simulation project, indeed any project, it is necessary to gain a detailed understanding of the operation of the system being studied. This understanding can be gained by a variety of means;

- The examination of historical production data;
- The review of standard operating procedures;
- The observation of the actual system to be modelled and;
- Interviews with system users from a variety of levels within the system.

It is the last point of interviewing system users on which this first example will focus. Such persons generally have a detailed knowledge of their particular areas of operation within a system.

These many sources of knowledge have then to be correlated by the person undertaking the project and in turn presented to the system owners and management in a format that can be easily reasoned over by all persons involved. This aids the model developer in gaining a proper understanding of the system and eliminates any ambiguities in understanding at an early stage, thus, reducing the risk of project overruns. This example examines a series of system interviews from the perspective of persons operating the different sections of the line in question. Such interviews can often form an initial reference from where more detailed information is gathered on the system.

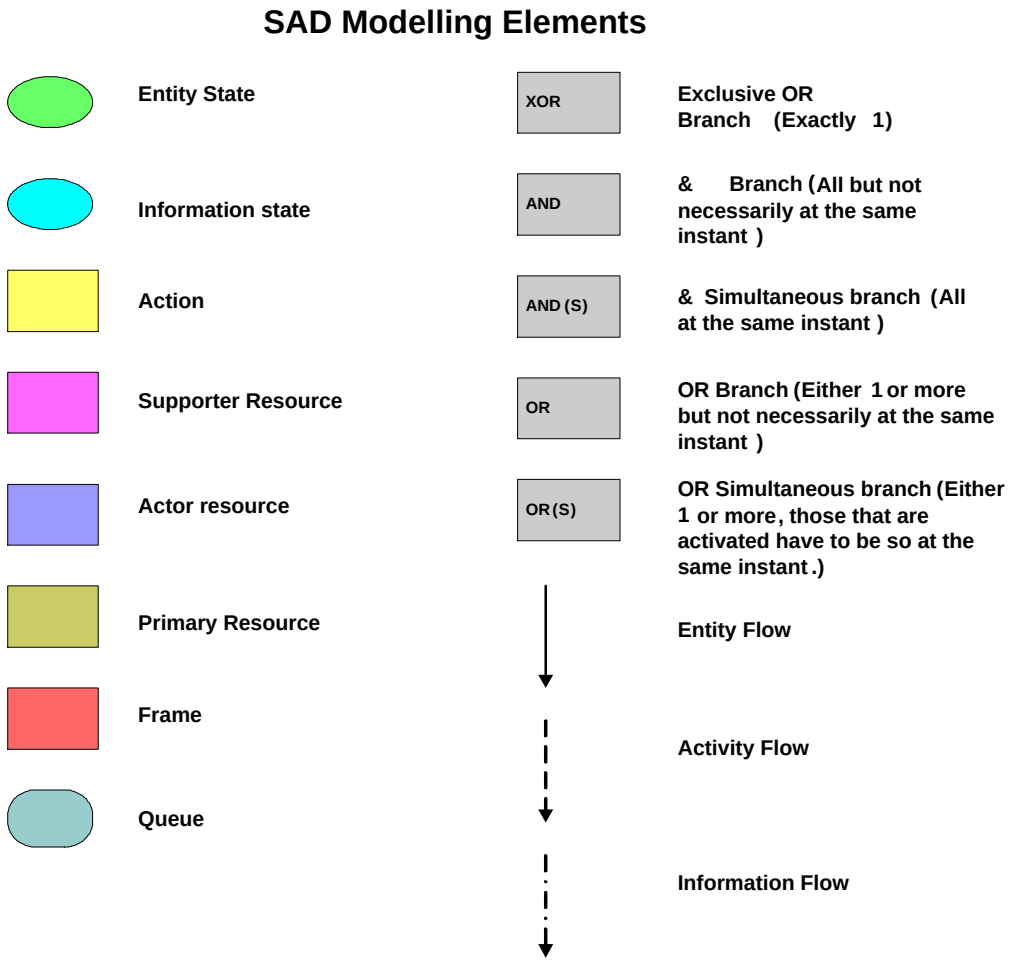

Figure 6.1 SAD Modelling elements

In this instance the series of interviews were used to develop a number of SAD diagrams and accompanying elaborations. The high level overview SAD, Figure 6.2 and its associated elaboration along with the SAD presented in section 6.2.1 are based around the results of a number of these interviews. The full version of this example is presented in Appendix A. As mentioned previously Figure 6.2 shows the highest level of the system modelled in this case. Here the various actions carried out by the production manager are shown as are the various flows of information and entities through the manufacturing facility. An elaboration language description of this highest level diagram is shown in Table 6.1. Figure 6.3 and Table 6.2 represent a SAD and an associated elaboration for the inspection area as outlined in section 6.2.1.

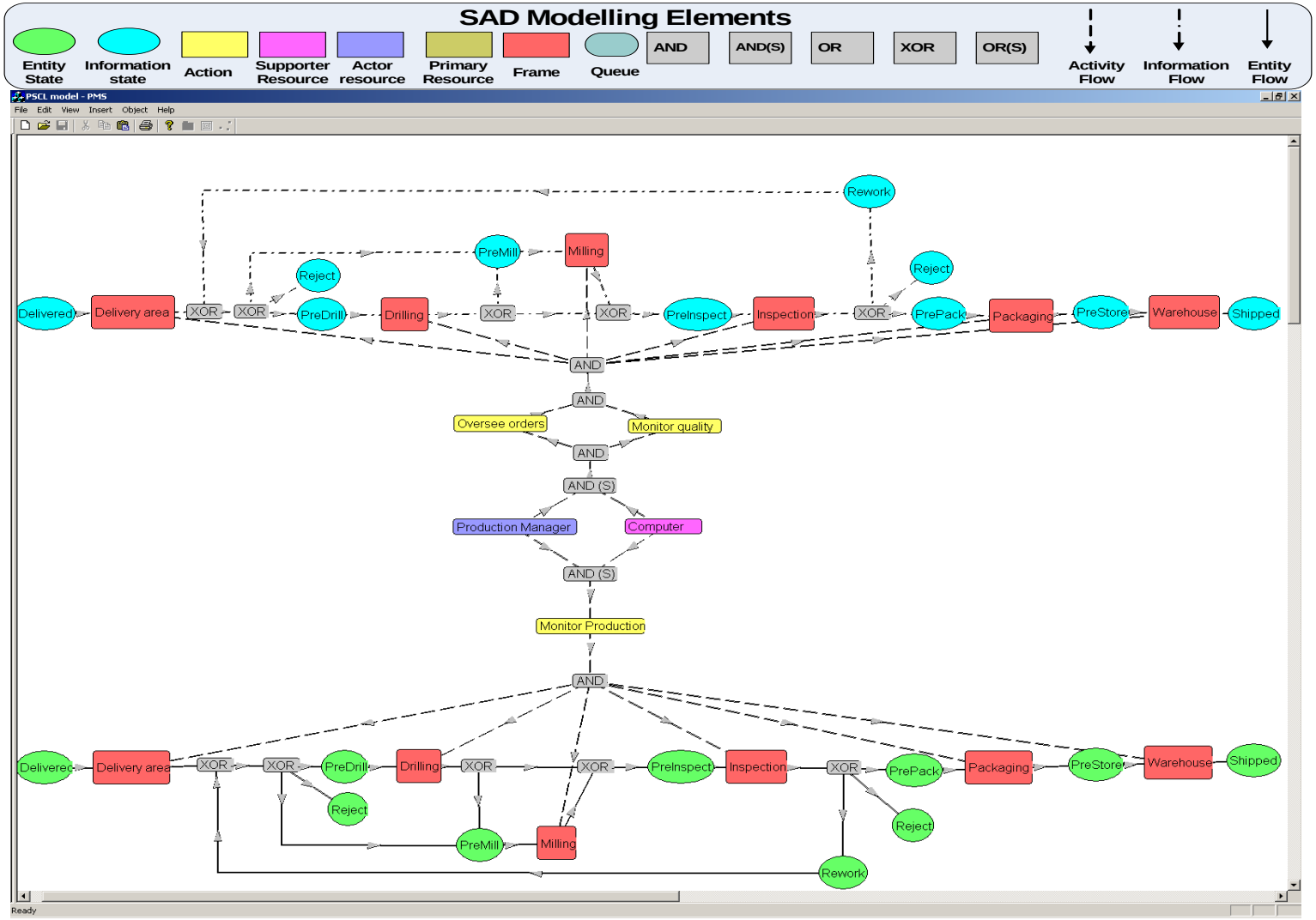

Figure 6.2 Highest level of precision component manufacturing system.

Elaboration of the Activity
Production Manager
USES
Computer
TO
Monitor Production
AT
Delivery area
AND
Drilling
AND
Milling
AND
Inspection
AND
Packaging
AND
Warehouse
AND
Production Manager
USES
Computer
TO
Oversee orders
AND
Monitor quality
AT
Delivery area
AND
Drilling
AND
Milling
AND
Inspection
AND
Packaging
AND
Warehouse
THEN
Delivered entity state
TRANSITIONS TO
Shipped entity state
AND
Delivered information state
TRANSITIONS TO
Shipped information state

Table 6.1 Elaboration description the Highest level of the precision component manufacturing SAD diagram

The high level SAD presented in Figure 6.2 consists of a number of frame elements, which are used to allow the hierarchical decomposition of a SAD diagram or particular system into more detailed SAD diagrams or subsystems. In this instance the frame elements are used to represent the following sub systems

or work areas; Delivery area, Drilling, Milling, Inspection, Packaging and Warehousing. The following section presents the SAD diagram and elaboration associated with the Inspection frame element. In other words this SAD diagram is used to represent more detailed information associated with the Inspection subsystem of the system being modelled. The remaining subsystems represented in the high level SAD diagram Figure 6.2 are presented in Appendix A.

6.2.1 Inspection

The inspection area consists of an inspection table where one operator inspects every part passing through the station. If the parts pass the inspection of the operator they are placed directly on a pallet for transfer to the packaging area. If the parts are found to be oversized for drilling or undersized for milling they are placed on a pallet for disposal. If the parts are found to be under sized for drilling or oversized for milling they are placed on pallets for transfer to their respective rework sections of the delivery holding area. The inspection area is modelled as shown in Figure 6.3, with elaboration language description of this area being contained in Table 6.2.

The following is the description given by the inspection operator;

> ***"Parts are placed into the inspection buffer from where I pick and inspect all parts. The inspection is a simple operation where I check the critical dimensions of each piece using a height gauge and a vernier callipers, the quality of the surface finish is also tested using an electronic surface tester. On the basis of these two tests I decide if a part needs to be reworked or not. If the part does not need to be reworked it is placed on a pallet for transportation directly to the packaging area. Where the part needs rework, it is placed on either a pallet for milling rework operations, drilling rework operations or both, for transport to the necessary holding section on completion of a batch. Oversized parts are also placed on a pallet for dumping."***

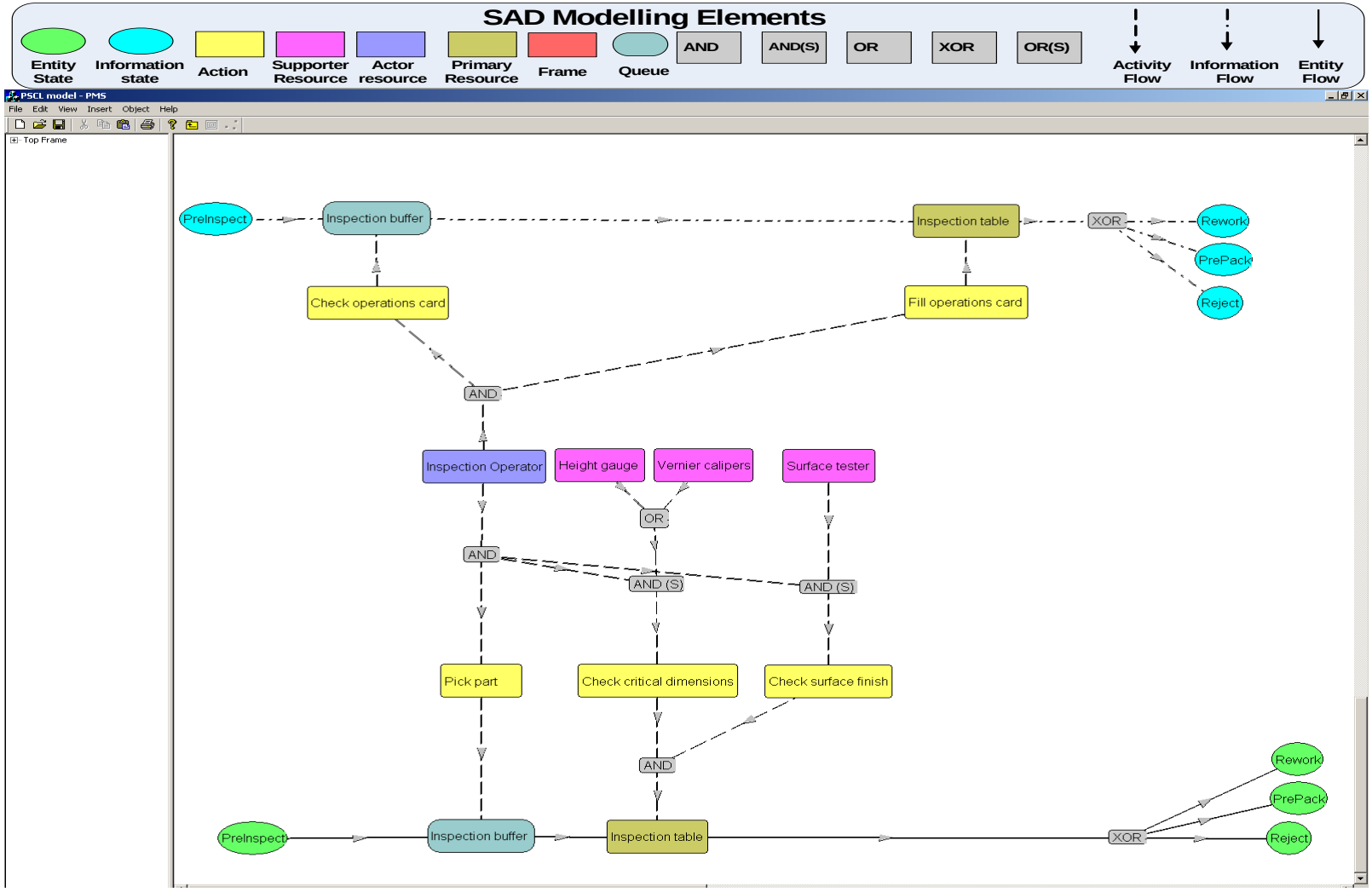

Figure 6.3 Inspection Area

Elaboration of the Activity
Inspection Operator
Picks part
AT
Inspection buffer
The Inspection buffer treats parts in a First In First Out (FIFO) manner
AND
USES
Height gauge
OR
Vernier calipers
The details of the critical dimension tests performed on the parts in the Inspection area are contained in the attached document (Dimension_tests.doc)
TO
Check critical dimensions
The setup times for this operation average 1.36 mins and the details of this are recorded in the attached document (Dimension_test_setup.xls) The average time taken for this operation is 5.8 mins, with the details contained in the attached document (Dimension_Op_Times.xls)
AND
USES
Surface tester
The details of the Surface finish tests performed on the parts in the Inspection area are contained in the attached document (Surface_tests.doc)
TO
Check surface finish
The setup times for this operation average 2.56 mins and the details of this are recorded in the attached document (Surface_test_setup.xls) The average time taken for this operation is 3.2 mins, with the details contained in the attached document (surface_Test_Times.xls) The Mean Time to Failure (MTF) and the Mean Time to Repair (MTR) for this operation are attached in the following documents respectively (Surface_test_MTF.xls) (Surface_test_MTR.xls)
AT
Inspection table
AND
Inspection Operator
Check operations card
AT
Inspection buffer
AND
Fill operations card
AT
Inspection table

THEN
PreInspect entity state
TRANSITIONS TO
EITHER
Rework entity state
OR
Prepack entity state
OR
Reject entity state
This transition is based on the results of the tests carried out on the parts by the inspection operator.
AND
PreInspect information state
TRANSITIONS TO
EITHER
Rework information state
OR
PrePack information state
OR
Reject information state
The transition here represents the transition of the operations card, which details each operation and in the case of the inspection operation, the outcome of the operation, which accompanies each batch of parts through the system.

Table 6.2 Elaboration language description for the inspection area

6.3 Modelling Production Control Systems

Many modern production systems use control systems to regulate production flow. Therefore, to accurately model such systems, a process modelling technique needs to be capable of representing both the physical transformations and the information or control systems associated with such physical transformations. To represent such a scenario the SAD technique was used to model a theoretical Kanban control system as introduced in the following section.

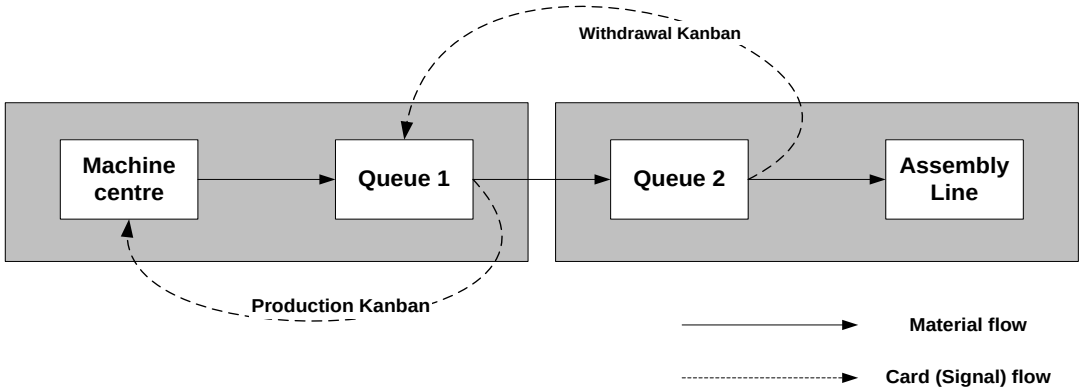

Figure 6.4 Types of Kanban card

The Kanban approach calls for a control system that is simple and self-regulating and provides good management visibility. The shop floor/vendor release and control system is called Kanban, from the Japanese word meaning card. It is a paperless system, using dedicated containers and recycling travelling requisition cards. This is referred to as a Kanban pull system, because the authority to produce or supply comes from downstream operations. While work schedules are planned based on schedules they are executed based on Kanbans, which are completely manual. There are two types of Kanban card, Figure 6.4. The production Kanban authorises the manufacture of a container of parts. The withdrawal Kanban authorises the withdrawal or movement of a container of parts. The number of parts in a container is fixed. When production rates change containers are added or deleted from the system.

6.3.1 SAD Model of a Kanban production control system

The following section models a theoretical Kanban production control system as introduced previously. When the assembly area takes the first part of type A from a full container, a worker takes the withdrawal Kanban from the container, and takes the card to the machine centre storage area. In the machine centre area, the worker finds a container of part A, removes the production Kanban, and replaces it with the withdrawal Kanban. Placement of this card on the container authorises the movement of the container to the assembly area. The freed production Kanban is placed on a rack by the machine centre, which authorises the production of another lot of material. The cards on the rack become the dispatch list for the machine centre. In a Kanban control system such as this the control system regulates the production system. As a result the SAD diagrams information system is used to show the way in which the production system is controlled. The following three SAD diagrams and accompanying elaborations shows how SAD diagrams can be used in a system as shown in Figure 6.4 above. Figure 6.5 shows the high level SAD diagram representing the overview or high level representation of the Kanban control system. In these examples it was decided to model the informational flow of the Kanban cards using three Information states. There are two physical cards but withdraw kanbans travel with both full and empty containers between work areas. Hence, three states are used, representing both cards and the container type. The flows of information and physical parts between two work areas represented by two frame elements, machine area and assembly area, are shown. These frame elements are further elaborated in the following pages. The elaboration associated with this SAD diagram is presented in Table 6.3.

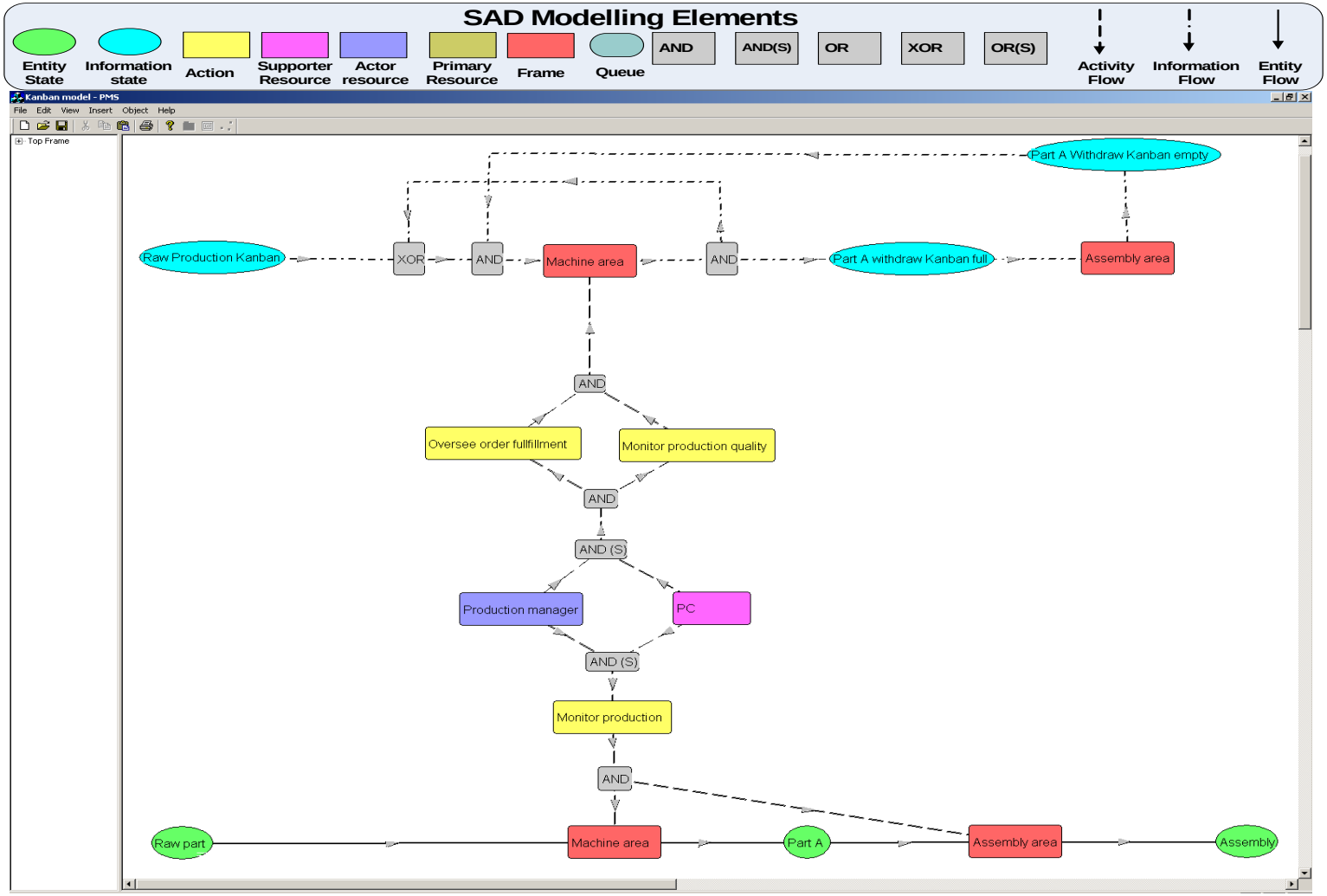

Figure 6.5 Kanban control example high level view

Elaboration of the Activity
Production manager
USES
PC
TO
Monitor production
AT
Machine area
AND
Assembly area
AND
Production manager
USES
PC
TO
Oversee order fulfillment
AND
Monitor production quality
AT
Machine area
AND
Assembly area
THEN
Raw part entity state
TRANSITIONS TO
Assembly entity state
AND
Raw Production Kanban information state
TRANSITIONS TO
Part A Withdraw kanban empty information state

Table 6.3 Kanban High level SAD elaboration

The SAD diagram presented in Figure 6.6 presents a further, more detailed representation of the Kanban control of the machining area within the example being examined. This diagram shows how both the machine and assembly operators carry out the manual control of production through use of Kanban cards. The elaboration associated with this SAD diagram is presented in Table 6.4 The assembly area is further described by means of the SAD diagram presented if Figure 6.7. In this SAD the progression of both the physical production system and the Kanban control system in association with the machining area are shown, as are the operator interactions with both the physical and control/information system. The elaboration associated with this SAD is presented in Table 6.5.

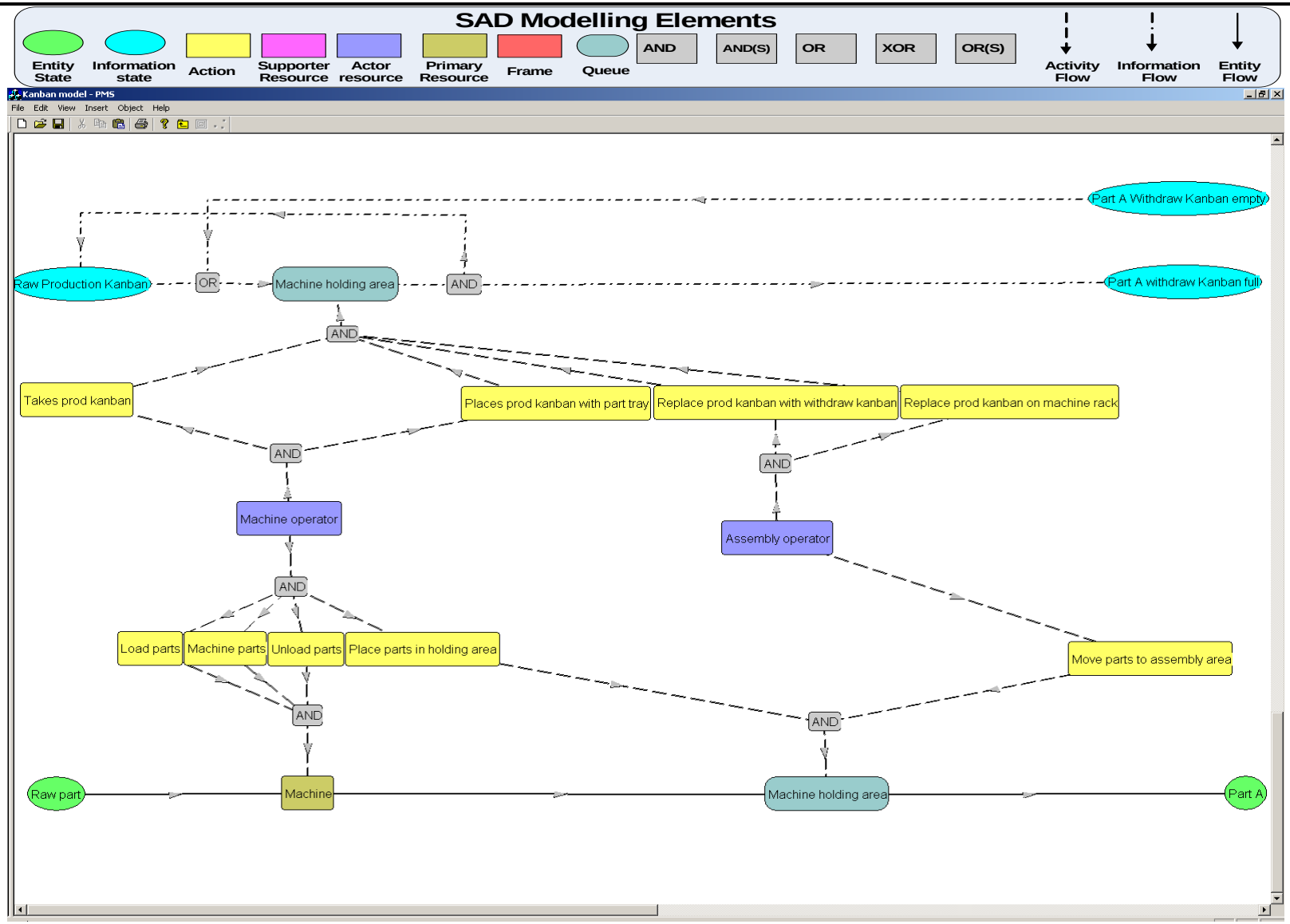

Figure 6.6 Kanban control of machining area

Elaboration of the Activity
Machine operator
Load parts
AND
Machine parts
AND
Unload parts
AT
Machine
AND
Place parts in holding area
AT
Machine holding area
AND
Machine operator
Takes prod kanban
AND
Places prod kanban with part tray
AT
Machine holding area
AND
Assembly operator
Move parts to assembly area
AT
Machine holding area
AND
Assembly operator
Replace prod kanban with withdraw kanban
AND
Replace prod kanban on machine rack
AT
Machine holding area
THEN
Raw part entity state
TRANSITIONS TO
Part A entity state
This transition is physically executed by the Assembly Operator who collects a batch of parts and brings them accompanied by the appropriate Kanban card to the Assembly area
AND
Raw Production Kanban information state
TRANSITIONS TO
Part A withdraw kanban full information state
AND
Raw Production Kanban information state
This transition is physically executed by the Assembly Operator who replaces the Production Kanban with a Withdrawal Kanban prior to the removal of the batch from the machining area. On doing this the Assembly operator places the production Kanban at the machining area thus freeing up more production

Table 6.4 Kanban machining area elaboration

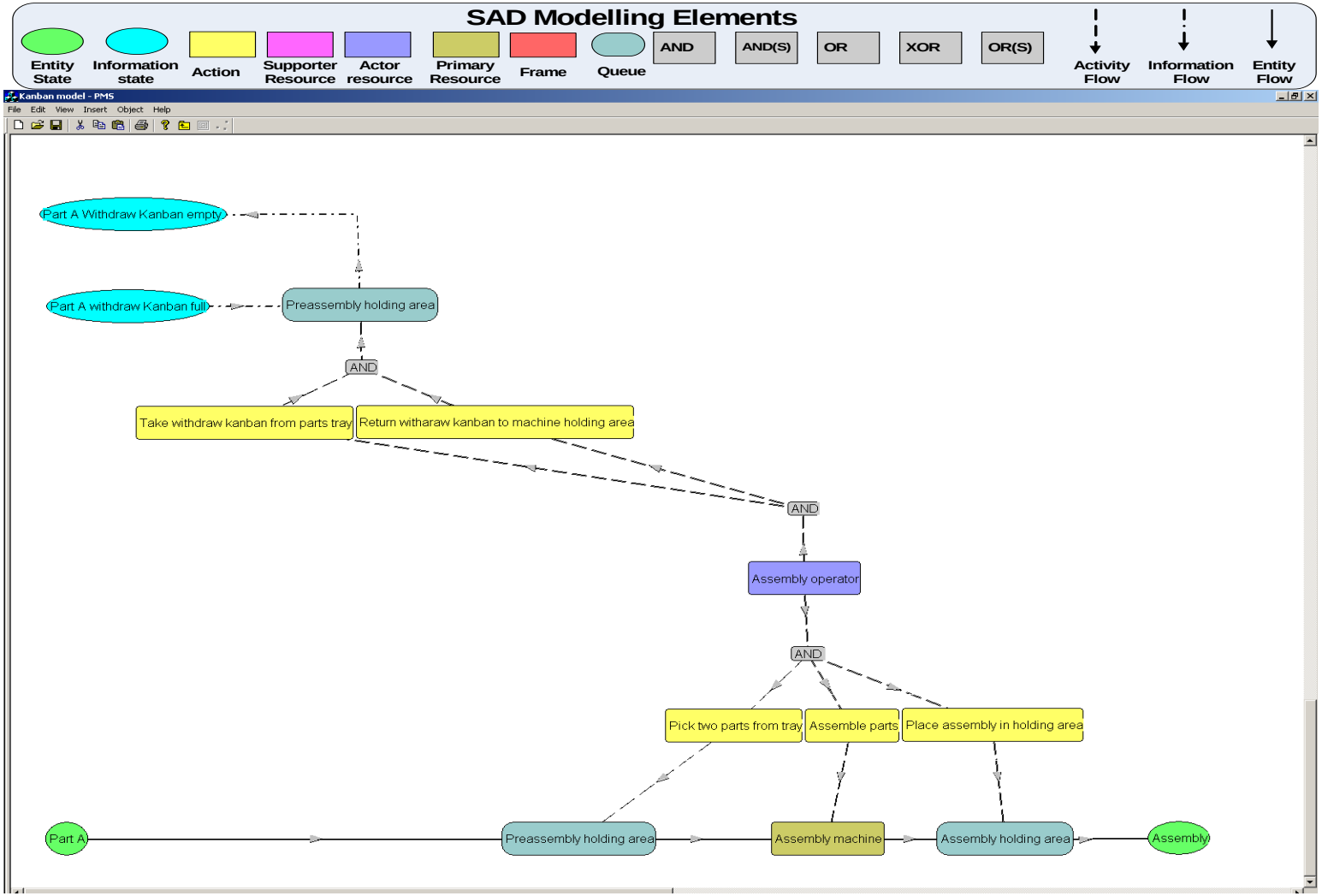

Figure 6.7 Kanban control example assembly area

Elaboration of the Activity
Assembly operator
Pick two parts from parts tray
AT
Preassembly holding Area
AND
Assemble parts
AT
Assembly machine
AND
Place assembly in holding area
At
Assembly holding area
AND
Assembly operator
Take withdraw kanban from parts tray
AND
Return withdraw kanban to machine holding area
AT
Preassembly holding area
THEN
Part A entity state
TRANSITIONS TO
Assembly entity state
AND
Part A withdraw Kanban full information state
TRANSITIONS TO
Part A Withdraw Kanban empty information state

Table 6.5 Kanban assembly area elaboration

6.4 Modelling a section of a batch flow-shop

The company modelled in this section produce mining consumables with the particular manufacturing system modelled producing mining rods. The manufacturing system can be classified as a batch flow-shop, consisting of four major work regions. The first region consists of pre-carburising operations. The second work region relates to the carburising or induction-hardening phase of the production process. The third work region encompasses the post-carburising operations and finishing operations and the final work region represents the final inspection of the product before dispatch to the relevant customer. The second work area is quiet complex in terms of the decisions made by operators and the amount of control vested in them. It is on modelling this operator control and decision making process that the following SAD example will concentrate.

6.4.1 Work Region two, carburising

Rods that require carburising are staged in the carburising area, until a sufficient quantity of rods required for the specific carburising setting are ready to be loaded onto a carburising jig for placement in the carburising furnace.

Before the rods are carburised certain preparatory operations are performed, e.g. inserting a carburising rope. To enter the furnace the rods are manually loaded onto a carburising jig. The carburising jig consists of a column, attached to which at varying intervals is a six sectioned “spider”. Placed within each section of this “spider” is a honeycomb tray, which allows the rods to be hung vertically in each section. The spider, honeycomb trays and rods contained therein are collectively known as a “tier”. The length of the rods being carburised determines the number of tiers on the jig. For very long rods only 1 tier is useable, for very short rods four tiers can be used. The diameter and shape of the rods determine the type of honeycomb tray that is used.

When the jig has been filled to capacity or near capacity, the operators use a crane to place the loaded jig into the furnace. The carburising furnace operates on a number of different carburising settings depending on the type of rods to be carburised. After the jig containing the rods is carburised, it must be transferred immediately to the cooling tower to be cooled under controlled conditions to ensure the required hardness is achieved by the carburising process. After the cooling tower the operators allow the jig to air cool until the rods are cool enough to be unloaded. The unloading operation is a manual operation, where the parts are unloaded and passed to the next work region.

6.4.2 Modelling the carburising area

The following section presents a SAD diagram developed to communicate the various interactions between the operators and the carburising part of the manufacturing system. Such interactions require the model developer to gather and communicate detailed information on a system. It is also necessary to be able to present such detailed information in a way to aid the model developer in communicating it to operational personnel for validation. To aid in the latter point the PMS software outlined in chapter 5 allows a

model developer to link documents containing detailed or specific information, which it may not be possible to graphically represent with the SAD technique. For instance, it is not possible to directly model precedence rules within the SAD diagramming technique, however it is possible to detail such precedence rules by attaching information such as this in the form of a document to the elements within the PMS software and in turn the elaboration language. It is in dealing with such scenarios that this example concentrates. The full example, along with the accompanying tables of information, are contained in Appendix B.

In this system parts arrive into the furnace area and wait until all operations such as roping, application of anti-carburising paint and stamping of the batch number have been performed. At this point the parts are split-up into separate holding areas based on the carburising setting, the carburising setting and cycle times are shown in Table 6.8. Within each of these carburising setting holding areas there are four further holding sections based on the product length. It is from these areas that a jig is built. A jig is made up of tiers of rods, of which there are a maximum of four on each jig. Each tier has six trays containing honeycombs into which rods are slotted. There are four types of trays;

- **Type A** Can hold a maximum of 16 rods;
- **Type B** Can hold a maximum of 12 rods;
- **Type C** Can hold a maximum of 9 rods;
- **Type D** Can hold a maximum of 3 rods.

It is also possible to build a jig containing trays of more than one type. A jig has a maximum length of 20 feet (6.1m) and can be placed in the furnace on completion of building providing the furnace is free. If parts are in the holding area for more than eight hours and there are not enough parts available of the particular type to build an entire jig then partially built jigs may be used.

The maximum numbers of rods that can be arranged on a jig are detailed in Tables B 4 to B 6. Table B 4 assumes that all rods on the jig are the same. This does not have to apply in reality. Provided that the rods all have the

same carburising cycle code, a jig can contain rods of varying types, lengths, diameters and shapes. There can even be different tray types on a single tier.

There are two furnace operators who are required to carry out the following prioritised operations;

- Load/unload the furnace;
- Build/dismantle a jig ;
- Load/unload the air cooling tower;
- Pre-jig building operations.

Pre-jig building operations consist of inserting rayon ropes, applying anti carburising paint and stamping the batch number on parts. The unloading of a jig takes thirty minutes and is taken as the highest priority or most important job within the furnace area, the operation descriptions and their priorities are shown in Table 6.6.

Priority	Description
1	Unloading the furnace
2	Building a jig (To ensure there is always a jig available)
3	Dismantle a jig
4	Load/Unload the air cooling tower
5	Pre-jig building operations

Table 6.6 Furnace operation priorities

Unloading the furnace occupies the operators for 30 minutes. This task is assigned the highest priority in the model and therefore, whenever it occurs the operators stop working on all other tasks and are pulled to the furnace. Building or dismantling of jigs is given the next highest priority. All other tasks have very low priority and cannot be started unless the aforementioned operations are not possible. Operators will attempt to build a jig before dismantling one so as to ensure that a jig will be available when the furnace requires one. However, jigs are a limited resource in that there are only three jigs in the furnace area. Also, jig building may not be complete when the furnace next becomes empty. The resources required to load and unload a jig are given in Tables B7 to B10.

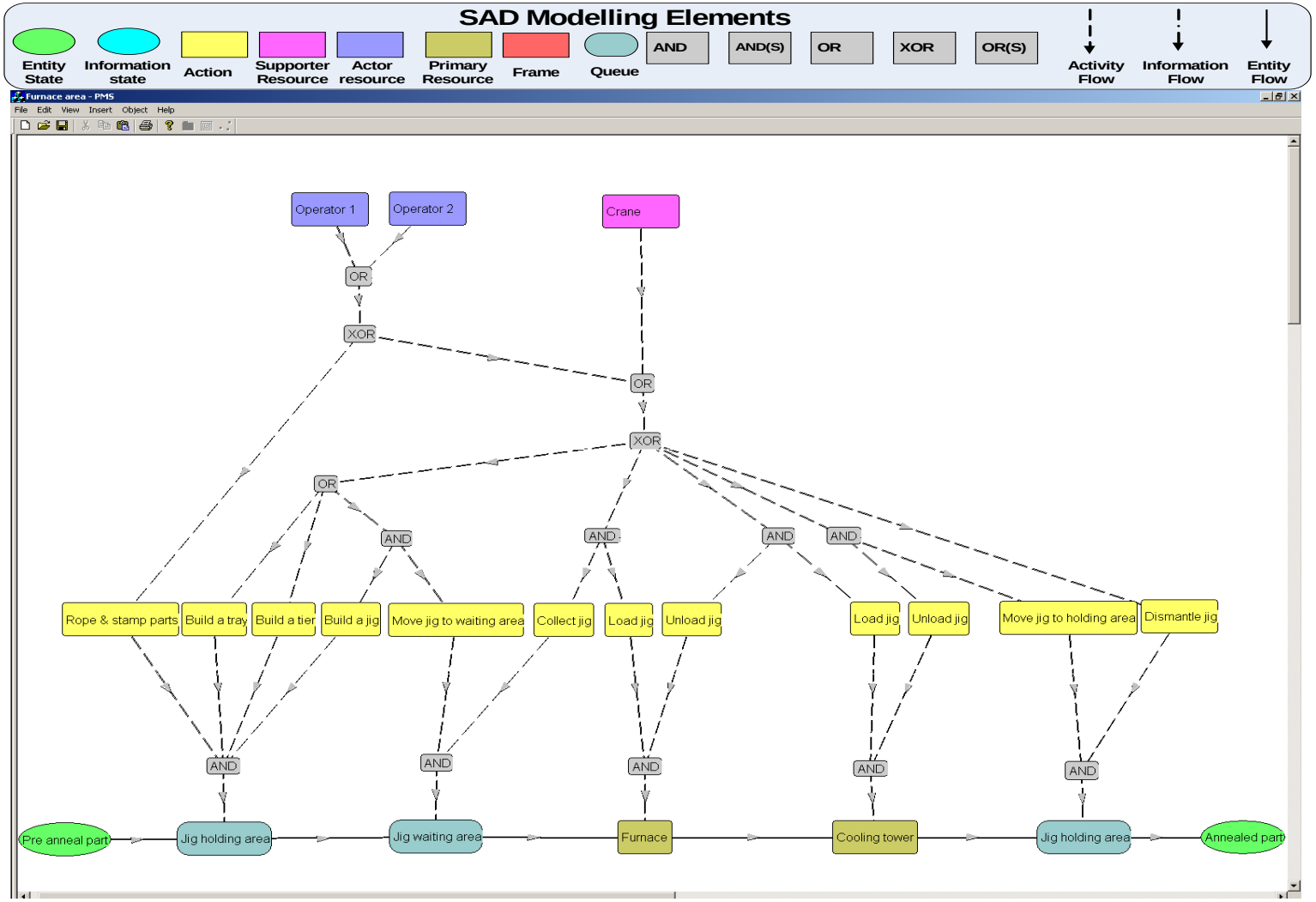

Figure 6.8 Furnace area SAD

Elaboration of the Activity
Operator 1
OR
Operator 2
EITHER The operations are outlined here in the sequence of execution to produce a part, however priority rules apply to the sequence of operations within the area and these priority rules are contained in an attached document (Furnace-operation- priorities.doc)
Rope & stamp parts
OR
OR
USES
Crane
The number of operators and need for a crane is dependant on the size of parts being placed on the tray/tier or jig. Details are contained in the following four attached documents. (Load-requirements-hex-rods.xls) (Load-requirements-round-rods.xls) (Unload-requirements-hex-rods.xls) (Unload-requirements-round-rods.xls)
TO
EITHER
Build a tray
There are four types of tray the details of which are contained in the attached document (tray-types.xls)
OR
Build a tier
A tier consists of six trays
OR
Build a jig
A jig is made up of a maximum of four tiers and each tier is made up of a number of trays. The number of tiers and trays used and the number of parts is dependant on the size and weight of parts with maximum limits on each. The details for this are contained within the following attached documents. (Max-Furnace-utilisation.xls) (Round-rod-weights.xls) (Hex-Rod-weights.xls) While fully built jigs are preferred, parts in the holding section for longer than eight hours may be used on partially built jigs.
AT
Jig holding area
AND
Move jig to waiting area
AT
Jig waiting area
OR
Collect jig
AT
Jig waiting area
AND
Load jig

	AT
	Furnace
	The furnace cycle times vary with the details contained in the attached document (Furnace-cycle-times.xls)
OR	
	Unload jig
	AT
	Furnace
	AND
	Load jig
	AT
	Cooling tower
OR	
	Unload jig
	AT
	Cooling tower
	AND
	Move jig to holding area
	AT
	Jig holding area
OR	
	Dismantle jig
	AT
	Jig holding area
AND	
THEN	
Pre anneal part entity state	
TRANSITIONS TO	
Annealed part entity state	

Table 6.7 Furnace area elaboration

	Setting	*Cycle Time (Hrs.)*
Frequently Used		
	2	8.5
	7	10.5
	8	4.5
	10	8.5
	12	6.5
	14	6.5
Occasionally Used		
	3	4.5
	6	6
	11	4.5
	13	8.5
	17	10.5

Table 6.8 Carburising Furnace Cycle Times

The SAD diagram for this area is shown in Figure 6.8, with the associated elaboration language being presented in Table 6.7

6.4 Modelling a Production line

When dealing with the development of simulation models for discrete event systems, a model developer often has to contend with a large amount of

information gathered from a variety of sources within a facility. The model developer then has to present this information in a manner that clearly communicates it to personnel involved in the operation and management of the system. The SAD technique facilitates the communication of such information by allowing the division of a part or family of parts into its various states of processing, both informational and physical. As each SAD diagram has to have both an entry and exit state this division into various states then allows a model developer to divide a system into many related areas or SADs. The following example models a production line used for the manufacture of diamond cutter discs. In this example, the SAD technique is used to model the overall production line, by giving an overview of the line and then providing more detailed information on the various production areas, represented by their own individual SAD diagrams. In this section only one area is presented, with the remaining areas contained in Appendix C.

6.4.1 74mm Syndite Line Product Description

The facility examined here consists of two main areas of production. These areas are divided in relation to activity type. There are the “bulk process” processing lines, which are used to complete work on material moving from bulk storage to buffer stock, and a second set of lines “finish cut”, which are used to complete products from buffer stock to a finished product. The bulk process lines consist of four dedicated lines. The first is dedicated to producing 74mm diameter discs of all sizes. The second line produces 57mm discs, with the third line being used to produce syndril products and the fourth and final line is used in the production of minority products and is known as “others”. It is in the modelling of the 74mm line that the following example concentrates. In the manufacture of 74mm diamond cutter discs, there are 17 products which are processed on the 74mm Syndite line. The product codes for these can be seen in Table 6.9.

Name	Product Code	Item No.
1	USYR7416 – 36005 002	HC000519
2	USYR7416 – 36005 010	HC000122
3	USYR7416 – 36005 025	HC000500
4	USYR7419 – 36005 002	HC000520
5	USYR7419 – 36005 010	HC000510
6	USYR7419 – 36005 025	HC000502
7	USYR7420 – 36005 002	HC000521
8	USYR7420 – 36005 010	HC000123
9	USYR7420 – 36005 025	HC000501
10	USYR7432 – 36005 002	HC000522
11	USYR7432 – 36005 010	HC000124
12	USYR7432 – 36005 025	HC000503
13	USHR7416 – 36005 025	HC000512
14	USHR7419 – 36005 025	HC000511
15	USHR7420 – 36005 025	HC000513
16	USHR7432 – 36005 025	HC000514
17	USQR7480 – 36007 025	HC000518

Table 6.9 74mm Syndite Line Products and Item Numbers.

The product code gives the details of the part. For example:

U SY R 74 16 – 360 05 002

- The U signifies that the part is for buffer stock;
- The SY signifies the type of part (SY = Syndite type CTB, SH = Syndite type CTH, SC = Syndite type CTC and SQ = Syndril);
- The R signifies that the part is a round part (i.e. a whole disc);
- The 74 signifies that the usable area of the disc (i.e. 74mm);
- The 16 signifies the thickness of the disc (i.e. 1.6mm);
- The 360 signifies that the part is a complete 360°;
- The 05 signifies the diamond layer thickness (i.e. .05mm);
- The 002 signifies the diamond grade.

Every part which flows through the 74mm Syndite line will follows an identical route. Table 6.10 gives an outline of the routing for the line, the actual processes used, the number of machines in each area, the number of operators per shift, the number of shifts per process and the number of discs, which are processed on each machine in one run.

As every part modelled in this particular production area follows the same routing it is possible to model all parts with one part family used to represent

this scenario. This approach is supported in the PMS modelling software, which allows for the modelling of part families as introduced in chapter 5.

Op. No.	Process	Machines	Operators	Shifts	Process batch per m/c
10	Centreless Grind	1	1	2	10
20	Face Grinding	2	1	3	1
30	Surface Grinding	2	1	3	10
40	EDM Planning	11	2	3	9
50	Finish Lapping	4	1	3	40
60	Sandblasting	1	1	1	1
70	Assessment	2	6	1	36

Table 6.10 74mm Syndite Process.

The high level SAD for this line is presented in Figure 6.9. In this diagram the various areas within the production line are graphically represented by frame elements. Each frame element is further described in the full example presented in Appendix C. The flow of both physical parts and information between each of these areas is also represented in the high level SAD in Figure 6.9. The elaboration language associated with this diagram is presented in Table 6.11. The SAD diagram and associated elaboration language for one of the areas within the production line, “Surface grinding”, is presented in section 6.4.2.

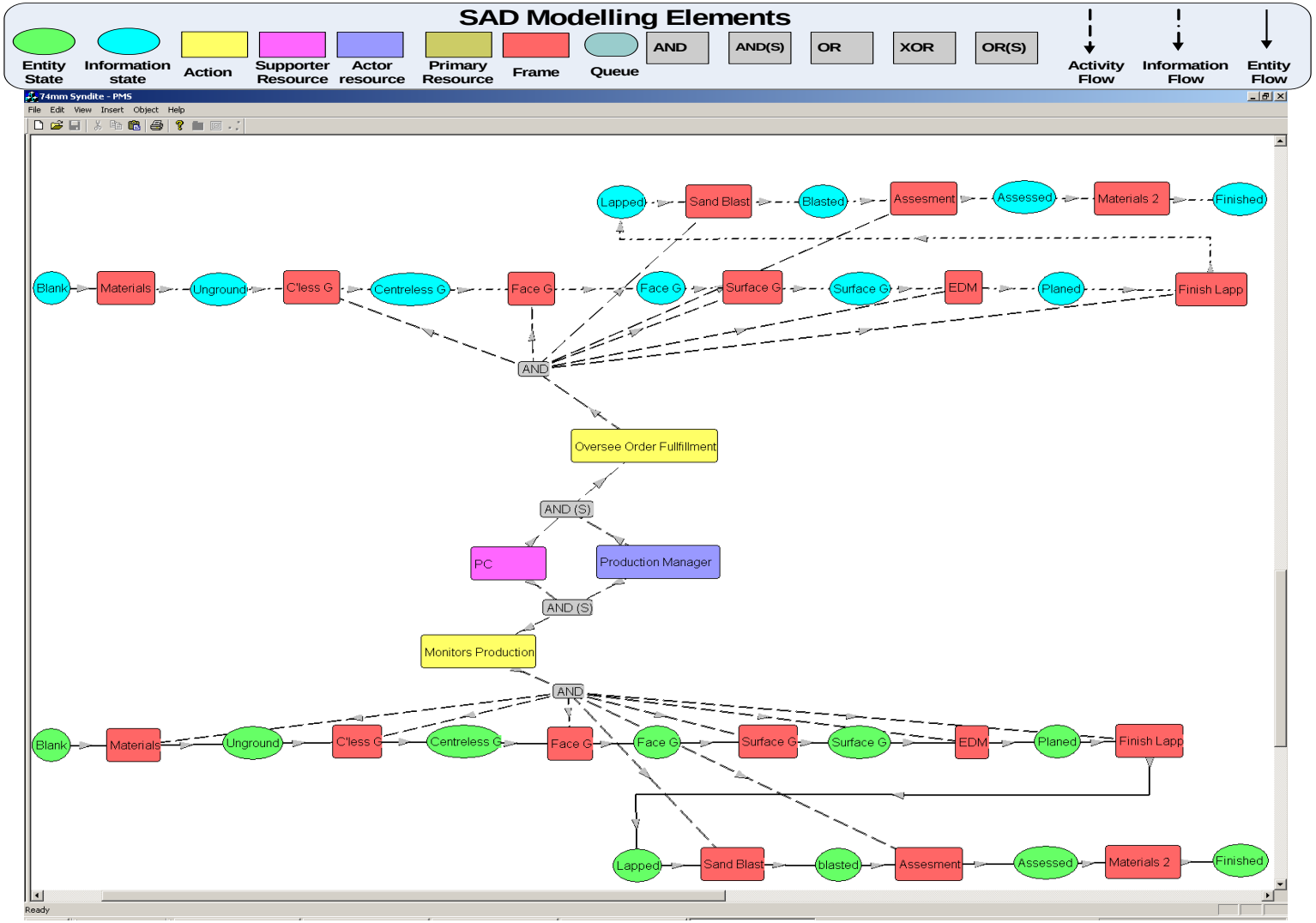

Figure 6.9 74mm High level SAD Diagram.

Elaboration of the Activity
Production Manager
USES
PC
TO
Monitors Production
AT
Materials
AND
C”less G
AND
Face G
AND
Surface G
AND
EDM
AND
Finish Lapp
AND
Sand Blast
AND
Assessment
AND
Materials 2
AND
Production Manager
USES
PC
TO
Oversee Order Fulfillment
AT
Materials
AND
C”less G
AND
Face G
AND
Surface G
AND
EDM
AND
Finish Lapp
AND
Sand Blast
AND
Assessment
AND
Materials 2
THEN
Blank entity state
TRANSITIONS TO
Finished entity state
AND
Blank information state
TRANSITIONS TO
Finished information state

Table 6.11 74mm High level SAD elaboration

6.4.2 Surface grinding

Surface grinding is one of the processes through which each part passes. The surface grinder grinds the carbide face of the disc to bring the disc to approximately 0.4mm above the height the material is going to end up as, when it has been completely processed by the 74mm Syndite line. There are two surface grinding machines (SSG13 & SSG14), which are used for the 74mm Syndite line. These two machines are located in a different area than the previous sets of machines. Each machine holds 10 discs per run. The discs are placed flat on 74mm washers, which are placed inside a frame on the table and spacers are placed between the discs to keep them apart. The spacers and the frame are thicker than the disc washers, in order to prevent lateral movement of the discs. Once the 10 discs have been placed on the machine, the table is magnetised holding the washers and the spacers in place. It is these in turn, which hold the discs in place. These machines are supported by a single operator, and are run over three shifts. The expected output from these machines is 50 parts per machine per shift (i.e. 5 runs of each machine per shift). When each machine has completed a run the operator removes one piece from the table and measures its thickness around the circumference. If it is within tolerance the rest of the discs are removed form the table and also measured. When all of these have been checked the next batch is measured before being loaded on the machine to determine the depth of material, which has to be removed. The parts are then loaded onto the machine and the cycle is started again. The operator waits for the second machine to complete its cycle. The SAD diagram for this area is shown in Figure 6.10, with the associated elaboration language being presented in Table 6.12.

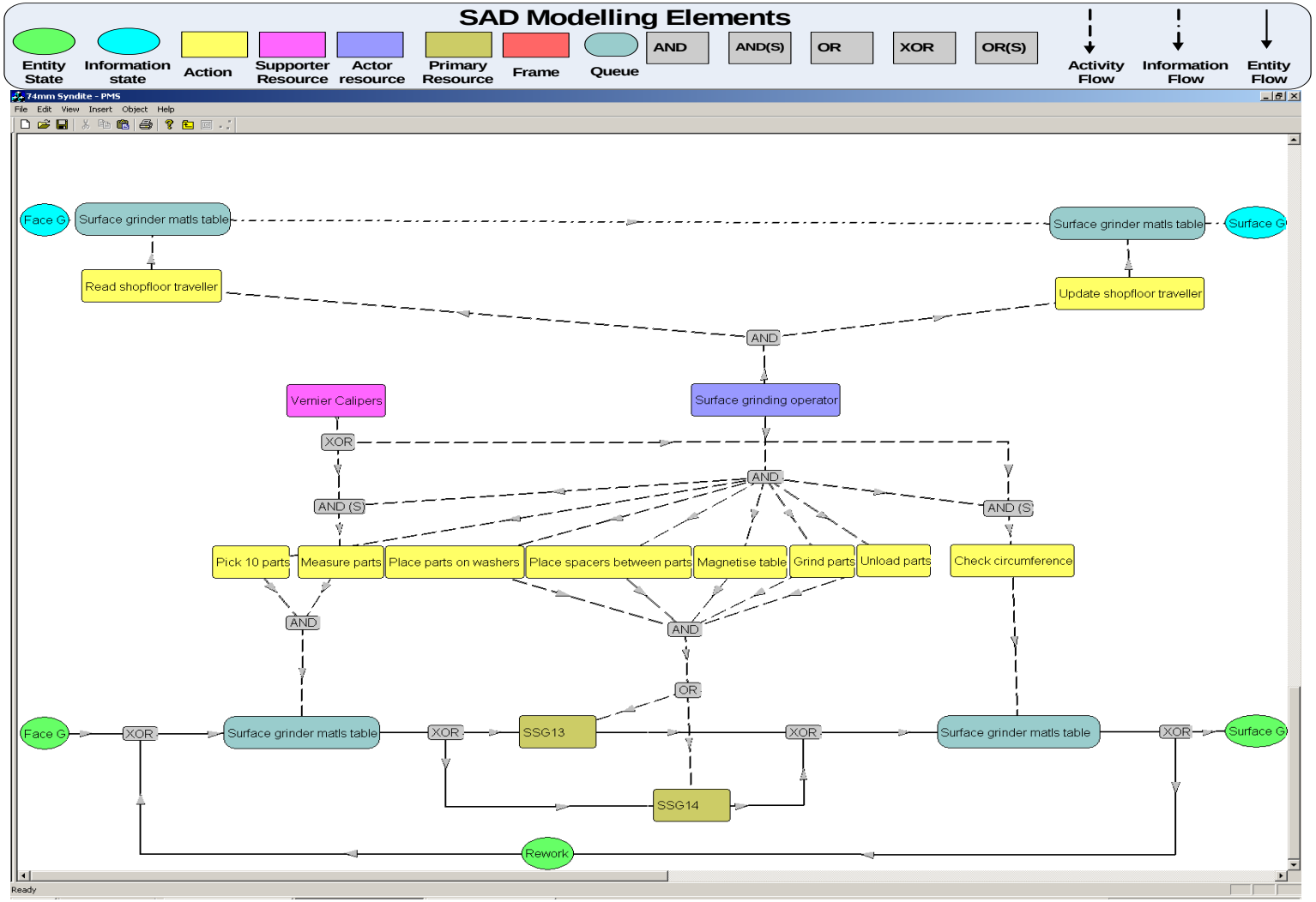

Figure 6.10 Surface Grinding

Elaboration of the Activity.
Surface grinding operator
Pick 10 parts
AND
USES
Vernier Calipers
TO
Measure parts
AT
Surface grinder matls table
AND
Place parts on washers
AND
Place spacers between parts
AND
Magnetise table
AND
Grind parts
AND
Unload parts
AT
SSG13
OR
SSG14
Each machine has a standard cycle time of 96 minutes.
AND
USES
Vernier Calipers
TO
Check circumference
AT
Surface grinder matls table
AND
Surface grinding operator
Read shopfloor traveller
AND
Update shopfloor traveller
AT
Surface grinder matls table
THEN
Face G entity state
TRANSITIONS TO
EITHER
Rework entity state
OR
Surface G entity state
AND
Face G information state
TRANSITIONS TO
Surface G information state

Table 6.12 Surface Grinding elaboration

6.5 Conclusions

This chapter presented a number of different discrete event systems modelled using the SAD technique. Each system modelled was used to demonstrate the ability of the SAD technique to model and communicate various aspects of discrete event systems. The first example illustrated the gaining of an initial overview of a discrete event system. In this instance this was gained through a series of interviews. From such interviews a number of SAD diagrams were developed. In this scenario, the SAD diagrams and associated elaborations that constitute the technique were used to represent the model developer's understanding of the system prior to gathering of detailed information. This avoids unnecessary data collection and misunderstandings of system functionality at too early a stage of a project.

In the second example, the SAD technique was used to model a theoretical production control system. This example was used to highlight the technique's capability to accurately model both a physical production system and an associated control system within the same model along with the interactions between both systems. Modern discrete event systems often consist of both physical transformations and associated with these, a means of controlling or regulating the physical system. Therefore a technique such as SADs has to be capable of representing such scenarios.

Thirdly, a discrete event system with a high degree of operator control was modelled. This example was used to represent the SAD technique's ability to accurately model a variety of operator/system based interactions and the representation of various system specific operational rules. Such information could not fully and accurately be represented in a graphical diagramming technique alone. The SAD elaboration language can be used in such an instance to support reasoning and the presentation of system logic in such a way as to eliminate any ambiguities. This manner of elaboration aids system communication and reasoning. To achieve this, the elaboration language allows a user to directly associate documents with descriptions of various system

aspects, which, when coupled with the elaboration language and SAD graphical representation seek to eliminate any ambiguities in understanding that may arise. This promotes the accurate communication of system issues and also supports simple reasoning with information.

The final example modelled a production line within a manufacturing facility. This example was used to represent the SAD technique's ability to accurately represent an entire production line and the interactions between the various areas of operation within such a line. The ability of the SAD technique to allow the grouping of a family of similar parts into a single representation was illustrated.

Each of the examples presented above highlighted particular aspects of discrete event systems and the ability of the SAD technique to model and represent such aspects in a manner that facilitates understanding and communication. As may be seen, many of the SAD diagrams contain details that may not be entirely necessary for the simulation of a system. For instance the sequence of actions, loading, machining and unloading a machine, would be normally grouped into a single time period. But, while the SAD technique is used to aid in the requirements gathering phase of a simulation project and such actions may indeed be grouped into a single time period for the purposes of modelling within a simulation tool, the technique is also used to promote communication and understanding among non-simulation personnel. Therefore, the inclusion of such actions, while perhaps not directly of benefit to a model developer in terms of model development, will aid in the communication with personnel as to how time periods used are arrived at. In this way such actions endeavour to promote understanding between model developer and system personnel.

Chapter 7: Conclusions

7.1 Thesis Summary

This thesis outlined the development of a process modelling technique specifically designed to aid a simulation model developer during the requirements gathering phase of a simulation project. The thesis highlighted the lack of techniques and tools available to specifically support the pre-simulation phases of a simulation project. While there are numerous process modelling tools available that can and have been used to support the requirements gathering phase of a simulation project, none fully support this phase of a simulation project. This area of pre-simulation coding was identified as important within the overall context of a simulation project and an important area in which to develop supports. This thesis therefore concerned itself with the development of a process modelling technique to overcome this shortfall. Initially a literature survey of process modelling techniques and tools capable of modelling discrete event systems was carried out to gain an insight into the various techniques and tools abilities to support the pre-simulation phases of a simulation project. While many of the techniques and tools examined were capable of being used to support the requirements gathering phases of a simulation project none were capable of capturing, representing and communicating the various aspects of discrete event systems. On completion of the literature review a design process was undertaken to develop a process modelling technique that was capable of modelling a discrete event system that satisfied the aforementioned criteria. This design process highlighted the various aspects of a discrete event system that need to be captured, represented and communicated to system personnel as outlined in the requirements, Chapter 1 page 7.

The outcome of this design process, named as Simulation Activity Diagrams (SADs) were presented in detail. These diagrams allow the encapsulation and visual representation of the various interactions between resources, information/control and physical systems within a discrete event environment.

A prototype software tool, PMS, was developed to support the representation of the SAD technique. This technique was then tried out on a number of actual and conceptual discrete event systems. Each system was chosen to validate the techniques ability to visually model and communicate different aspects of a discrete event system that may be encountered during the requirements gathering phases of a simulation project, including a full production system, interview information, complex resource interactions and information flows.

7.2 Reflection

As outlined in Chapter 1 this thesis focused on developing a process modelling technique to support the requirements gathering/conceptual modelling phases of a simulation project. To fully support this, the requirements outlined in Chapter 1 were introduced as goals. As discussed in Chapter 3 none of the techniques examined in Chapter 2 fully satisfied the requirements outlined in Chapter 1. However as a result of the development process undertaken the resultant SAD technique it is felt better fulfils the requirements outlined initially in Chapter 1 than any of the techniques examined previously, Figure 7.1.

A

Technique	Good Communication / Visualisation medium	User Perspective	State flow modelling	Information flow modelling	Resource modelling	Activity Modelling	Complex branching logic	Elaboration Language
Petri Nets	Medium	Low	High	Medium	Low	High	Medium	Low
ACDs	Medium	Low	High	Medium	Low	High	Low	Low
DEVS	Low	Low	High	Medium	Medium	Medium	Medium	Low
UML Activity Diagrams	High	Low	Low	Medium	Low	High	High	Low
UML Statecharts	High	Low	High	Medium	Low	Low	High	Low
RADs	High	High	Low	Medium	Medium	High	Medium	Low
GRAI	High	Low	Medium	High	Medium	High	Low	Low
IEM	High	Medium	High	Medium	High	High	Medium	Low
EDPCs	High	Low	Low	Medium	High	High	High	Low
IDEF0	High	Low	Low	Medium	Medium	High	Low	Low
IDEF3	High	Low	High	Low	Medium	High	High	Medium

B

SADs	High	High	High	High	High	High	High	High

Figure 7.1 (A) Techniques requirements satisfaction (B) Requirements claims for SAD

It is felt the SAD technique satisfies each of the requirements developed above in the following ways. The technique is highly visual and capable of communicating complex discrete event system logic through use of its various modeling elements and their interactions within a model. The perspective of the user is placed centrally within every SAD model by means of a specialization of an auxiliary resource known as an actor auxiliary resource. Both state, entity, and information occurrences are explicitly represented within each SAD model by means of entity and information state elements and their corresponding links. Resources are also central to each SAD model, with a distinction drawn between primary and auxiliary resources to distinguish between resources used to transform various state elements and those used to support such transformations. Auxiliary resources are also subdivided into actor and supporter resources to distinguish between a system user and other supporting resources. The activities that decide the progress of a discrete event system are also graphically represented by means of action elements. The division and grouping of various lines of execution within a SAD diagram, entity, information and activity are graphically represented by means of the SAD branching elements. The SAD technique also facilitates the decomposition of a model into more complex sub

models by means of a Frame element, thus allowing for the separation of varying levels of detail. Finally graphical models alone cannot always capture all aspects of a complex discrete event system. To account for this the SAD technique has a SAD elaboration language associated with it, which can be used to further explain any aspect of a SAD diagram.

To further explore the SAD techniques ability to support the modeling of discrete event systems a software implementation of the technique was developed. This Process Modelling for Simulation (PMS) software was used to further develop the concepts outlined as requirements in Chapter 1. The PMS tool allowed for the development of the graphical SAD models. The software was also used to develop a means of automatically generating the text based SAD elaboration language from the graphical SAD models. While this has been implemented within the PMS tool further developments are required to fully implement this functionality. It is hoped with some modifications to allow for the full linking between the actual graphical model and the elaboration text by means of a step through facility, which would lead a user simultaneously through both models by means of simple animation/highlighting.

The SAD technique is designed specifically to model discrete event systems and has not been developed with a view to modelling continuous simulation systems. While the SAD technique has been designed to model discrete event systems it has not to date been fully validated as being capable of representing all aspects of a complex discrete event system and further work will be necessary in this area. To date the technique has been tested on a number of aspects as outlined in Chapter 6. Initially the Precision component manufacturing model was used to validate the draft conceptual model of the SAD technique and its component interactions to ascertain the validity and communicative capabilities of the technique. Having iterated through a number of development phases using the expert opinions as outlined in Chapter 3 a number of subsequent systems were modelled using the technique to further explore the techniques ability to model certain aspects of a discrete event system. The Precision component manufacturing model was as previously discussed used as an initial model to

determine the techniques ability to model all general aspects of a system. Thereafter the Batch flow shop model was used to test more fully the SADs ability to accurately represent complex user/resource interactions within a discrete event system. A kanban system was modelled to examine whether or not the SAD technique was capable of accurately representing an information system in conjunction with a manufacturing system. Finally a full production line was modelled to examine the SADs ability to capture information on such a system. At each phase of development expert opinion was sought as to the ability of the modelling technique to communicate discrete event system issues in a manner conducive to the facilitation of understanding and communication to person not necessarily trained in the field of simulation modelling.

The SAD technique while not yet supplying a full and definitive support tool for the requirements gathering phases of a simulation project does it is felt by satisfying the initial requirements outlined in chapter 1 go some way towards acting as an initial solution space. The technique is not a definitive solution and as such will need further refinement, validation and development. A number of issues are still in need of addressing. Theses include the incorporation of multiple modelling views, this would allow a model developer to initially model the system requirements ‘as is’ model and from this develop a second system view or conceptual model. The facilitation of a process whereby both models could be developed in the same format and viewed simultaneously would it is felt further enhance communication and understanding. The full implementation of the step through facility discussed previously would also it is felt be advantageous. It is also felt that there is a need for the development of further techniques to support a simulation model developer in these pre coding phases of a simulation project. It is hoped that further research will be carried out in this area with a view to the development of such techniques. The advantages that such techniques may offer while being difficult to accurately predict may include a number of the following. The development of detailed, valid and visual process models of complex discrete event systems prior to the coding of simulation models may save time and ultimately money in the development of simulation models. The number of

project failures could be reduced as a result of access to correct information and the development of valid and understandable models earlier in a simulation project. Such models should also facilitate better understanding of the process of simulation among non-simulation experts. This communication should allow for the reduction in the time taken to complete simulation projects, as model developers should be able to retrieve the necessary information for the project at an earlier stage in the project life cycle. The information gathered should also be more accurate and focused in relation to the problem areas being examined thus reducing project iterations at a later stage or in more extreme cases project failures. Graphical and accurate models of a problem area may even negate the necessity of simulation model development in certain cases as a solution may become apparent through the initial process modelling phase of a project.

7.3 Conclusions

In Summary the main conclusions of this thesis are:

- From the literature is was apparent that there is a lack of specific support available to aid a simulation model developer in the pre-coding phases of a simulation project
- There are many process modelling techniques available that may be used to aid in the modelling of various aspects of a discrete event system. However there are currently no process modelling techniques available that were developed specifically to support the requirements gathering or conceptual model development phases of a discrete event simulation project.
- It is a hypothesis of this thesis that to fully support a simulation project a full range of pre-simulation modelling techniques should be provided to aid a simulation model developer in the pre-simulation phases of a simulation project.

- This thesis proposed the development of a process modelling technique, Simulation Activity Diagrams (SADs) in an attempt to specifically support the requirements gathering phase of a simulation project
- SADs attempt to graphically represent discrete event systems in a high level and user friendly manner by attempting to represent physical, control resource and action information in a single model.
- Available graphical process modelling techniques are not always capable of representing all complex discrete event system information and require a textual means of communicating such information. The PMS prototype attempts to support this by means of the SAD elaboration language.
- To attempt to demonstrate the ability of the SAD technique to model discrete event information a prototype process modelling tool Process Modelling for Simulation (PMS). This prototype was used to demonstrate the SADs ability to model different aspects of discrete event systems.
- The SAD technique requires further validation and development.
- From the survey there is a lack of research by the research community into the pre-coding area of simulation. More research is required into developing new tools/techniques in this area.

7.4 Future Work

The following are recommendations for future research work in this area:

- One are of future research is to continue examining the area of pre-simulation coding with a view to developing techniques and tools specifically for the purposes of aiding a simulation model developer in the pre-coding phases of a simulation project.

- The SAD technique requires further validation, to date the technique has not been used within the full cycle of a simulation project, this validation is vital to fully ascertain the techniques applicability to supporting the pre-simulation phases of a simulation project.
- Further research could also be undertaken into ways in which the SAD technique could further support the pre-coding phases of a simulation project. In its current format the SAD technique is primarily a requirements gathering tool. There are other pre-simulation phases such as conceptual modelling that may be supported by such a technique with further developments.
- Further research into the development of the PMS prototype software could be undertaken to improve and extend a number of aspects of the tool such as the user interface to allow for the easier development of models. Further development could also be undertaken into the step through capabilities of the software tool elaboration function. Such an improvement would allow for a better visual representation of the interactions between the key graphical elements within a SAD model and their representative elaboration language.
- As the boundaries between problem formulation, model development, and coding are not generally well defined the development of a method in which the language that the conceptual problem is defined could also serve to outline the simulation model may be beneficial to a model developer. For such a method to be practical the language in which the conceptual model is defined would have to be transferable to a neutral representation format. This neutral representation format then can be used to transfer the information to the simulation engine or other software of choice.

References:

1. Forgionne, G.A. (1983) "Corporate Management Science Activities: An Update", in ***Interfaces***, Vol. 13(3) pp 20-23.

2. Harpell, J.L., Lane, M.S. and Mansour, A.H. (1989) "Operations research in practice: A longitudinal study", in ***Interfaces***, Vol. 19(3) pp 65-74.

3. Keller, L., Harrell, C. and Leavey J. (1991) "The three reasons why simulation fails", in ***Industrial Engineering***, Vol 23 pp 27-31.

4. Pegden, C.D., Sadowski, R.P. and Shannon R.E. (1995) ***Introduction to Simulation using Siman***, McGraw-Hill Education.

5. EMPlant http://www.tecnomatix.com. Date Accessed 12/09/2003

6. Kelton, D.W., Sadowski, R.P. and Sadowski, D.A. (1998) ***Simulation with Arena***, McGraw-Hill.

7. ED, T., http://www.taylorii.com. Date Accessed 15/09/2003

8. Balci, O. (1986) "Credibility Assessment of Simulation Results", in ***Proceedings of the 1986 Winter Simulation Conference,*** Piscataway, NJ, pp 38-43

9. Schlenoff, C., Knutilla A., and Ray S. (1996) ***Unified Process Specification Language: Requirements for Modelling Process***, National Institute of Standards and Technology, Gaithersburg.

10. Johansson, H.J., McHugh, P., Pendlebury, A.J., and Wheeler, W.A. (1993) ***Business Process Reengineering Breakpoint strategies for market dominance***, John Wiley & Sons Ltd.

11. Kawalek, P. and Kueng, P. (1997) "The Usefulness of Process Models: A Lifecycle Description of how Process Models are used in Modern Organisations", paper presented at ***International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design***. Barcelona, Spain.

12. Brian W. Hollocks: (2001) "Discrete-event simulation: an inquiry into user practice" in ***Simulation. Practice and Theory***, Vol. 8(6-7): pp 451-471

13. Robert G. Sargent: (1999) "Validation and verification of simulation models" In ***proceedings of the 1999 Winter Simulation Conference***: pp 39-48

14. Conwell, C.L., R. Enright, and M.A. Stutzman. (2000) "Capability Maturity Models Support of Modeling and Simulation Verification, Validation, and Accreditation" In ***Proceedings of the 2000 Winter Simulation Conference***, Orlando, Society for Computer Simulation International.

15. Nethe, A., and Stahlmann H.D. (1999) "Survey of a General Theory of Process Modelling", in ***International Conference on Process Modelling***, Cottbus, 22-24. February 1999.

16. Silvia T. Acuña, Xavier Ferré (2001***)*** "Software Process Modelling" paper presented at ***World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001)*** Orlando, FL (USA).

17. Kettinger, W.J., Teng, J.T.C. and Guha, S. (1997) "Business process change: A study of methodologies, techniques and tools", ***MIS Quarterly***, Vol. 21(1) pp 55-80.

18. Robinson, S. (2004) ***Simulation: The Practice of Model Development and Use***, Wiley.

19. Petri, C.A., (1962) ***Kommunikation mit Automaten***, University of Bonn, West Germany.

20. Desrochers, A.A. and AL-Jarr, R.Y. (1995) ***Applications of Petri Nets in Manufacturing Systems***, IEEE Press.

21 Vojnar, T. (1997) "Various Kinds of Petri Nets in Simulation and Modelling", in ***Proceedings of 31st Spring International Conference on Modelling and Simulation of Systems MOSIS"97***, Hradec and Moravici, Czech Republic.

22 Ou-Yang, C. and Shieh, C.M. (1999) "Developing a Petri-net-based simulation model for a modified hierarchical shop floor control framework", in ***International Journal of Production Research***, Vol. 37(14) pp 3139-3167.

23 Balduzzi, F.; Giua, A.; Seatzu, C. (2001) "Modelling and simulation of manufacturing systems with first order hybrid petri nets", in: ***International Journal of Production Research***, Vol. 39(2) pp 255-282.

24. Koriem, S.M. (2000) "A fuzzy Petri net tool for modelling and verification of knowledge-based systems" in ***The Computer Journal***, Vol. 43(3) pp 206-223.

25. Shih, M.H. and Leung, H.K.C. (1997) "Management Petri net - a modelling tool for management systems", in ***International Journal of Production Research***, Vol. 35(6) pp 1665-1680.

26. Evans, J.B. (1988) ***Structures of Discrete Event Simulation: An Introduction to the Engagement Strategy***, Ellis Horwood.

27. Ceric, V. and Paul, R. (1992) "Diagrammatic representations of the conceptual simulation model for discrete event systems", in ***Mathematics and Computers in Simulation***, Vol. 34(3-4) pp 317-324.

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

29. Chwif, L., Paul, R.J. and Barreto, M.R.P. (1999) ***Combining the best of the two: An activity cycle diagram / condition specification approach***, paper presented at UKSim'99 at St. Catherine's College, Cambridge.

30. Pidd, M. (1992) ***Computer simulation in management science***, 3rd Ed, Wiley.

31. Martinez, C.J. and Ioannou, G.P. (1995) "Advantages of the activity scanning approach in the modelling of complex construction processes", in ***Proceedings of the 27th Winter simulation conference***, ACM Press, pp 1024-1031.

32. Shi, J. (1997) "A conceptual activity cycle-based simulation modelling method", in ***Proceedings of the 29th Winter simulation conference***, ACM Press, pp 1127-1133.

33. Zeigler, B.P. (1984) ***Multifacetted Modelling and Discrete Event Simulation***, Academic Press.

34. Zeigler, B.P. (1990) ***Object-Oriented Simulation with Hierarchical, Modular Models***, Academic Press.

35. Rozenblit, J.W., Hu, J., Kim, T.G. and Zeigler B.P. (1990) "Knowledge based design and simulation environment: foundational concepts and implementation", in ***Journal of Operational Research Society***, Vol. 41 pp 475-489.

36. Thomasma, T. and Ulgen, O.M. (1988) "Hierarchical, Modular simulation modelling in icon-based simulation program generators for manufacturing", in ***Proceedings of the 20thWinter simulation conference***, ACM Press pp 254-262.

37. UML, (1997) ***UML Summary***, Rational Software Corporation.

38. Richter, H. and Marz, L. (2000) "Toward a standard process: the use of UML for designing simulation models" in ***Proceedings of the 32nd Winter simulation conference***, ACM Press, pp 394 - 398.

39. Rodrigues, R.S.W. (2000) ***Formalising UML activity diagrams using finite state processes***, paper presented at 3rd International Conference on the Unified Modelling Language. 2-6 October, 2000, York, UK.

40. Harel, D. (1987) "Statecharts: A visual formalism for complex systems" in ***Science of Computer Programming***, Vol. 8(3) pp 231-274.

41. Muller, P.A. (1997) ***Instant UML***, Wrox Press.

42. Barjis, J. and Shishkov, B. (2001) "UML Based Business Systems Modelling and Simulation", in ***Proceedings of Eurosim 2001 – Shaping Future with Simulation***, 4th International Eurosim Congress, Delft, The Netherlands.

43. Niere, J. and Zundorf, A. (1999/2000) "Testing and simulating production control systems using the Fujaba Environment" in ***Proceedings of International Workshop, AGTIVE'99 (Applications of Graph Transformations with Industrial Relevance), September 1-3, 1999***, Lecture Notes in Computer Science Series, Volume 1779, Springer Publications.

44. Borger, E., Cavarra, A. and Riccobene, E. (2000) "Modelling the Dynamics of UML State Machines", in ***Proceedings of the International Workshop on Abstract State Machines, Theory and Applications (ASM"2000)***, pp223-241, Springer-Verlag.

45. Pap, Zs. , Majzik, I., Pataricza, A. and Szegi, A. (2001) "Completeness and Consistency Analysis of UML Statechart Specifications", in ***Proceedings of the Design and Diagnostics of Electronic Circuits and Systems (DDECS) Workshop, April 2001, Győr, Hungary***, pp. 83-90.

46. Hu, Z. and Shatz, S.M. (2004) "Mapping UML Diagrams to a Petri Net Notation for System Simulation", in ***International Conference on Software Engineering and Knowledge Engineering (SEKE), 20-24 June 2004***. Banff, Canada.

47. Ould, M.A., (1995) ***Business Processes: Modelling and analysis for re-engineering and improvement***, Wiley.

48. Murdoch, J. and McDermid, J.A. (2000) "Modelling engineering design processes with role activity diagrams", in ***Transactions of the Society for Design and Process Science (SDPS)***, Vol. 4(2) p. 45-65.

49. Dawkins, S. (1998) "Role Activity Diagrams for Safety Process Definition" in ***16th International System Safety Conference***, Seattle, WA, USA.

50. Abeysinghe, G., Henderson, P., Phalp, K.T., and Walters, R.J. (1998) "RoleEnact: Enactable Models of Business processes", in ***Information and Software Technology***, Vol. 40 pp 123-133.

51. Heavey, C. and Ryan, J. (2002) ***Process Modelling for Simulation***, paper presented at 19th International Manufacturing Conference 28th-30th August Queens University Belfast.

52. Doumeingts, G. (1985) "How to decentralise decisions through GRAI Modelling production management" in ***Computers in Industry***, Vol. 6 pp 501-514.

53. Tucker, D. and Leonard, R. (2001) "An Innovative approach for using the GRAI methodology for reengineering the new product introduction process", in ***The International Journal of Flexible Manufacturing Systems***, Vol. 13 pp 177-193.

54. Chen, D., Vallespir, B. and Doumeingts, G., (1997) "GRAI integrated methodology and its mapping onto generic enterprise reference architecture and methodology", in ***Computers in Industry***, Vol. 33 pp 387-394.

55. Doumeingts, G. (1989) "GRAI approach to designing and controlling advanced manufacturing systems in a CIM environment", in Nof, S.Y. and Moodie, C.Y. (Eds) ***Advanced Information Technology for Industrial Material Flow Systems,*** Springer-Verlag.

56. Doumeingts, G. and Ducq, Y. (2001) "Enterprise modelling techniques to improve efficency of enterprises" in ***Production Planning and Control***, Vol. 12(2) pp 146-163.

57. Zulch, G., Rinn, A. and Strate, O. (2001) "Dynamic analysis of changes in decisional structures of production systems" in ***International Journal of Production Economics***, Vol. 69 pp 239-252.

58. Doumeingts, G., Ducq, Y., Vallespir, B. and Kleinhans, S. (2000) "Production management and enterprise modelling" in ***Computers in Industry***, Vol. 42 pp 245-263.

59. Ducq, Y., Vallespir, B. and Doumeingts, G. (2001) "Coherence analysis methods for production systems by performance aggregation", in ***International Journal of Production Economics***, Vol. 69 pp 23-37.

60. Mertins, K. and Jochem, R. (1999) ***Quality-oriented Design of Business Processes***, Kluwer Academic.

61. Mertins, K., Jochem, R. and Jakel, F.W. (1997) "A tool for object-oriented modelling and analysis of business processes" in ***Computers in Industry***, Vol. 33 pp 345-356.

62. Keller, G., Nuttgens, M. and Scheer A.W. (1992) "Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK)". in ***Veroffentlichungen des Instituts fur Wirtschaftsinformatik***, University of Saarland, Saarbrucken.

63. Tardieu, H., Rochfeld, A. and Colletti, R. (1983) ***La Methode Merise, Principes et Outil,*** Les editions d'organisation.

64. Nuttgens, M., Feld, T. and Zimmermann, V. (1998) "Business process modelling with EPC and UML: Transformation or integration" in Schader, M., Korthaus, A. and Groom-Workshop on the Unified Modelling Language ***The unified modelling language - technical aspects and applications***, Springer-Verlag.

65. van der Aalst, W.M.P. (1999) "Formalization and Verification of Event-Driven Process Chains" in ***Information and Software Technology***, Vol. 41(10) pp 639-650.

66. Markus, U. (1998) "Process oriented entrepreneurship support with internet solutions". in ***Internationalizing Entrepreneurship education and training***. (Proceedings of the IntEnt-Conference European Business School, Schloß Reichartshausen, Germany, July 27-29) Schloss Reichartshausen, Oestrich-Winkel, Germany.

67. Scheer, A.W., Nuttgens, M. and Zimmermann, V. (1997) "Objektorientierte Ereignisgesteuerte Prozesskette (oEPK) - Methode und Anwendung" in ***Veroffentlichungen des Instituts fur Wirtschatfsinformatik***, Saarbrucken.

68. Knowledge Based Systems Inc. (KBSI) (1994) ***IDEF0 Function Modelling Manual***.

69. Mayer, R.J. (Ed) (1994) ***IDEF1 Information Modelling***, Knowledge Based Systems, Inc

70. National Institute of Standards and Technology (1993) ***Integration Definition for Information Modelling (IDEF1X)***, National Institute of Standards and Technology.

71. http://www.idef.com/idef3.html Date Accessed 20/11/2003

72. Mayer, R.J., Menzel, C.P., Painter, M.K., Blinn, T and deWitte, P.S. (1995) ***Information Integration for concurrent engineering (IICE) IDEF3 process description capture method report***, Knowledge Based Systems Inc.

73. deWitte, P.S. and Mayer, R.J. (1995) ***IDEF4 Object-Oriented Design Method Report***, Knowledge Based Systems Inc.

74. Benjamin, P.C., Menzel, C.P., Mayer, R.J., Fillion, F., Futrell, M., deWitte, P.S. and Lingineni, M. (1994) ***Information Integration for Concurrent Engineering (IICE) IDEF5 Method Report***, Knowledge Based Systems Inc.

75. Jeong, K.Y. (2000) "Conceptual frame for development of optimised simulation-based scheduling systems", ***Expert Systems with Applications***, 18(4): pp 299-306.

76. Perera, T. and Liyanage, K. (2000) "Methodology for rapid identification and collection of input data in the simulation of manufacturing systems" in ***Simulation Practice and Theory***, Vol. 7(7) pp 645-656.

77. van Rensburg, A. and Zwemstra, N. (1995) "Implementing IDEF techniques as simulation modelling specifications" in ***Computers & Industrial Engineering***, Vol. 29(1-4) pp 467-471.

78. Al-Ahmari, A.M.A. and Ridgway, K. (1999) "An integrated modelling method to support manufacturing systems analysis and design" in ***Computers in Industry***, Vol. 28(3) pp 225-238.

79. Whitman, L., Huff, B. and Presley, A. (1997) "Structured models and dynamic systems analysis : The integration of the IDEF0/IDEF3 modelling methods and

discrete event simulation" in ***Proceedings of the 29th conference on Winter simulation***, ACM Press.

80. Scheer, A.W. (1998) "ARIS", in Mertins, K. and Schmidt, G. (Eds) ***Handbook on Architectures of Information Systems***, Springer-Verlag.

81. Carrie, A. and Plaia, A. "Application and assessment of IDEF3 - process flow description capture method" in ***International Journal of Operations and Production Management***, Vol. 15(1) pp 63-73.

82. Law, A M. and Kelton, W.D. (2000). ***Simulation Modeling and Analysis***. McGrawHill, 3rd edition.

83. Sagasti, F. R. & Mitroff, I. I. 1973. Operations research from the viewpoint of general systems theory. ***OMEGA***, 1:695-709

84. Landry, M., J-L. Malouin, M. Oral, (1983), "Model Validation in Operations Research," ***European Journal of Operational Research***, Vol.14, pp.207-220 (Invited Paper).

85. Schlenoff, C., Knutilla, A., Ray, S., Unified Process Specification Language: Requirements for Modeling Process, NIST Interagency Report 5910, Gaithersburg, MD, September 1996. PSL

86. Beeckman, Dirk, "*CIMOSA: Computer Integrated Manufacturing --- Open System Architecture*", ***International Journal of Computer Integrated Manufacturing***, Vol. 2, pp. 94--105, 1989.

87. Fox, M.S., Chionglo, J.F. and Fadel, F.G., 1993. "A Common-Sense Model of the Enterprise" In ***Proceedings of the Second Industrial Engineers Research Conference (IERC)*** Norcross, GA, Institute for Industrial Engineers.

88. Williams, T. J. (Editor), A Reference Model for Computer Integrated Manufacturing (CIM), ISA Publications, Research Triangle Park, NC (1989)

89. Geraghty, J. and Heavey, C. 2004. A Review and Comparison of Hybrid and Pull-Type Production Control Strategies Accepted for publication ***OR Spectrum***, October 2004.

90. Geraghty, J. and Heavey, C. 2004. A Comparison of Hybrid Push/Pull and CONWIP/Pull Production Inventory Control Policies. ***International Journal of Production Economics***. Volume 91 Issue 1, pp75-91.

91. Geraghty, J. and Heavey, C. 2001. An Application of Order Release Mechanisms in a Batch Production Flow Shop. ***International Journal of Industrial Engineering - Theory Applications and Practice***, 2001. Volume 8 Issue 3, pp251-261.

92. Crawford, J.W. and Gallwey, T.J. "Bias and variance reduction in computer simulation studies" ***European Journal of Operational Research***, v 124, n 3, 1 Aug. 2000, p 571-90

93. Khanian, S. and C. Heavey. 2001. "Development of a Manufacturing Simulation Model Using Object Oriented Principles". ***Third Aegean International Conference on Design and Analysis of Manufacturing Systems***, Tinos Island, Greece, May 2001 . Third Aegean International Conference on Design and Analysis of Manufacturing Systems, Tinos Island, Greece, May 2001.

94. Khanian, S. and C. Heavey. 2000. "A Process Modelling System to Support Operations Management". In: ***9th ASIM Dedicated Conference on Simulation in Production and Logistics***, Berlin, Germany, March, pp. 185-195 . In: 9th ASIM Dedicated Conference on Simulation in Production and Logistics, Berlin, Germany, March, pp. 185-195..

95. McNally, P. and C. Heavey. Simulation as a Desktop Resource: The Limitations and Future Requirements of Simulation Technologies. in ***19th International Manufacturing Conference***. 2002. Belfast, Northern Ireland

96. McNally, P. and C. Heavey. Development of an Integrated Simulation Model to Support Decision Making Within an SME. in ***17th International Manufacturing Conference***. 2000. Galway, Ireland.

97. Khanian, S. and C. Heavey. 1999. "Development of a Manufacturing Model to Facilitate Maintenance and Reuse". In: Proceedings of ***15th International Conference on Production Research,*** M. T. Hillery and H. I. Lewis (eds.). University of Limerick, August 9-12. In: Proceedings of ***15th International Conference on Production Research***, M. T. Hillery and H. I. Lewis (eds.). University of Limerick, August 9-12.

98. Heavey, C. and S.M.S. Khanian, 2002. "Development of a process model simulator". In: ***Simulation Modelling Practice and Theory***, 2002. 10(1-2): p. 13-33.

Appendix A: Example of SADs representing a precision component manufacturing system

A.1 Introduction

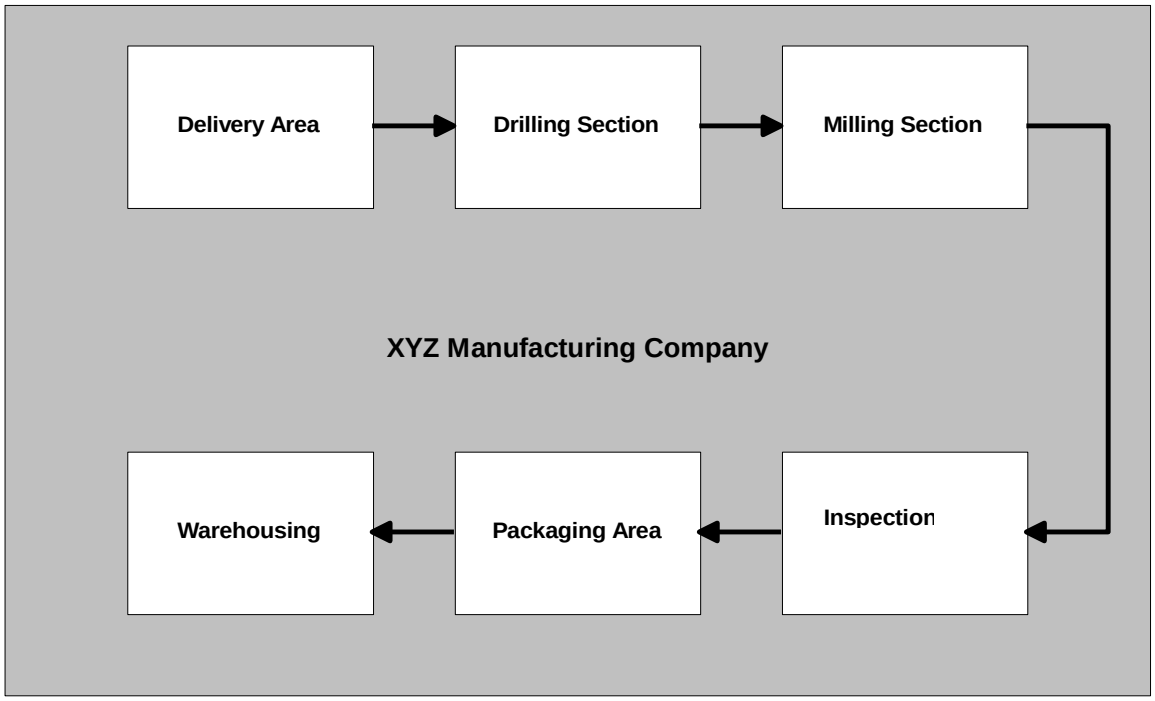

Figure A.1 Shop floor layout

The system outlined in this section is based on the results of a series of system interviews conducted with a number of workers in a precision component manufacturing facility in Galway, Ireland. In the early stages of any simulation project, indeed any project, it is necessary to gain a detailed understanding of the operation of the system being studied. The shop-floor layout of the manufacturing facility is shown in Figure. A.1 and consists of six separate areas of processing. Each of these areas is modelled using the SAD modelling technique. Figure A.2 shows the highest level of the system modelled in this case. Here the various actions carried out by the production manager are shown as are the various flows of information and entities through the manufacturing facility. An elaboration language description of this highest level diagram is shown in Table A.1.

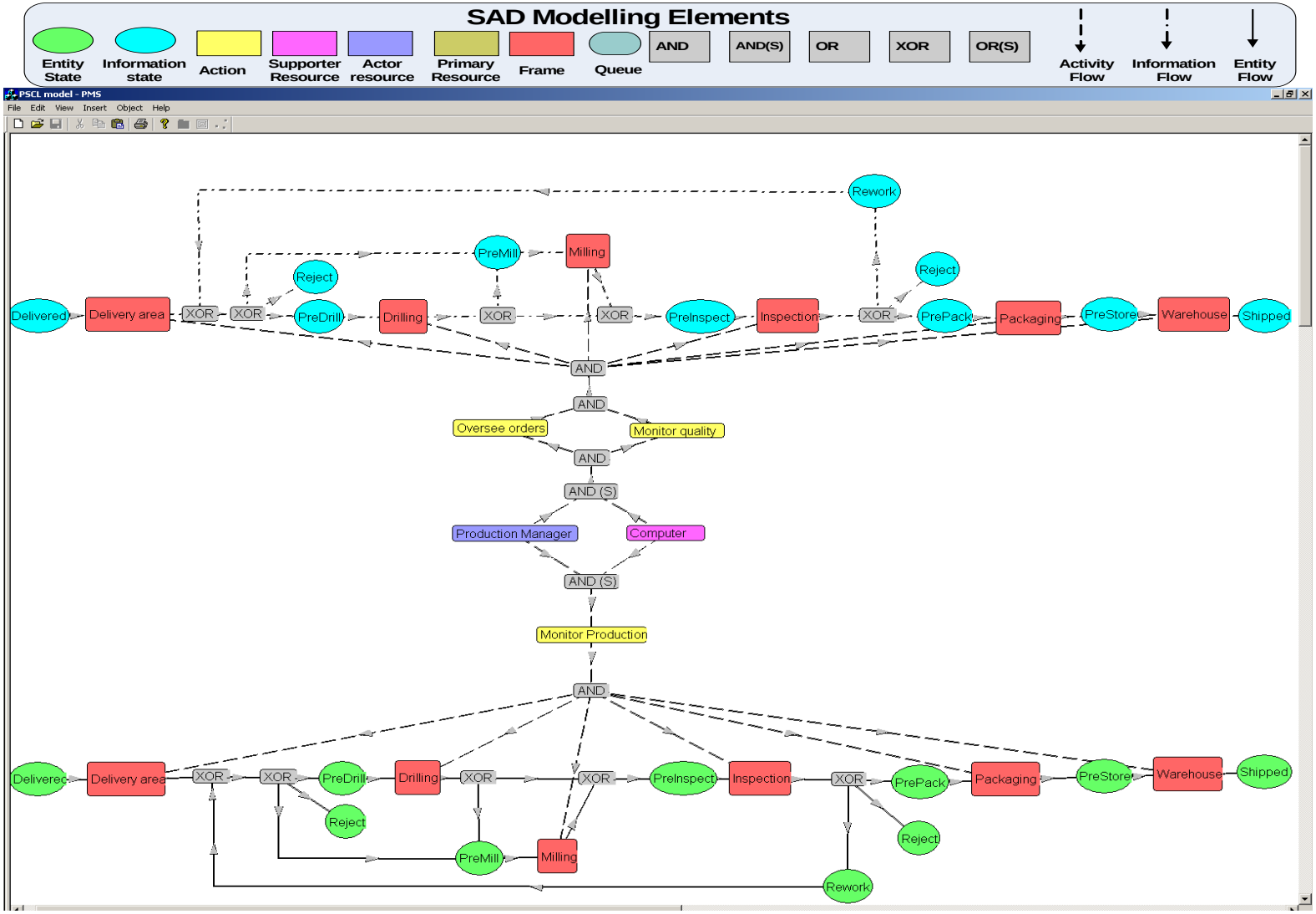

Figure A.2 Highest level of the system.

Elaboration of the Activity
Production Manager
USES
Computer
TO
Monitor Production
AT
Delivery area
AND
Drilling
AND
Milling
AND
Inspection
AND
Packaging
AND
Warehouse
AND
Production Manager
USES
Computer
TO
Oversee orders
AND
Monitor quality
AT
Delivery area
AND
Drilling
AND
Milling
AND
Inspection
AND
Packaging
AND
Warehouse
THEN
Delivered entity state
TRANSITIONS TO
Shipped entity state
AND
Delivered information state
TRANSITIONS TO
Shipped information state

Table A.1 Elaboration description the Highest level SAD diagram

A.2 Delivery

In this section of the facility, parts are delivered in pallets of 100 parts. The company deals in the repair and upgrade of three types of component. These

components follow the same general route through the facility but are graded on the type of repair work which has to be carried out. To decide the exact route taken by each part through the facility the goods inwards inspector carries out an inspection on the parts and fills out their routing on an operations card. This routing may vary between the drilling and milling sections, with routings through these stations being dependant on the condition of the part. If a part needs either a drilling, or a milling operation or both carried out, it is recorded on the operations card before the parts are passed to the necessary holding areas.

The following is a description of how the goods inwards inspector describes his job;

> ***"Parts arrive once a day. When a consignment of parts arrive I initially carry out a visual inspection to ensure the proper quantities of parts are present. If so I sign for the parts. Having returned the documentation to the deliverer I carry out a detailed inspection of the parts present. These parts are then separated into sections according to the types of rework, which have to be carried out, milling, drilling or both. Having separated the parts into pallets, I fill out an operations sheet for each pallet and place it on each respective pallet, denoting what operations are to be carried out on each pallet. After this I deliver the pallets to the respective holding areas, these being the drilling and milling holding areas. This entire operation generally takes half an hour to complete."***

Figure A.3 shows the SAD model for the delivery area, with Table A.2 outlining the elaboration language description of the model.

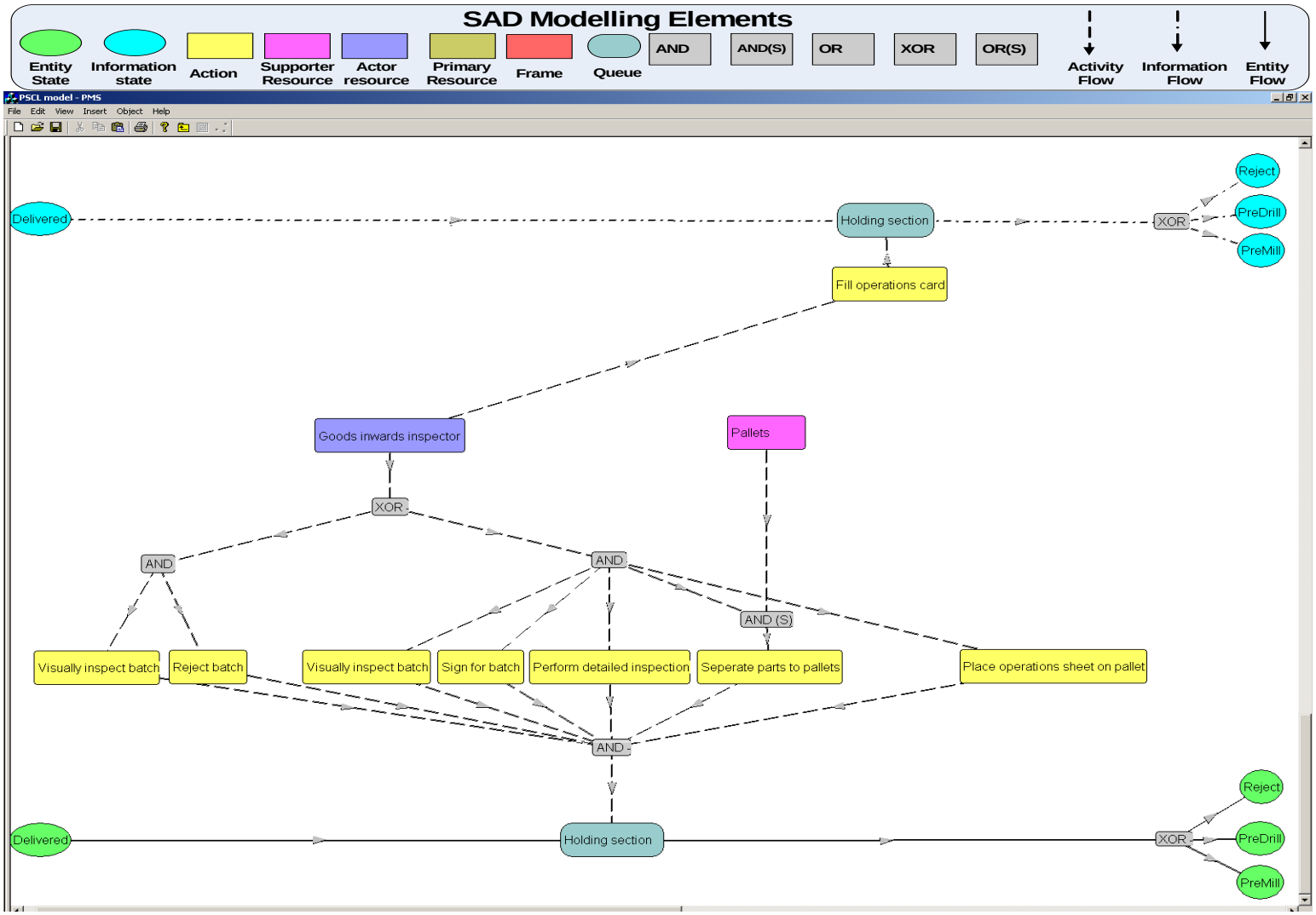

Figure A.3 Delivery Area

Elaboration of the Activity
Goods Inwards Inspector
EITHER
Visually inspect batch
AND
Reject batch
AT
Holding section
OR
Visually inspect batch
AND
Sign for batch
AND
Perform detailed inspection
AND
USES
Pallets
TO
Separate parts to pallets
There are three different types of pallets for parts milling, drilling or both.
AND
Place operations sheet on pallet
AT
Holding section
AND
Goods Inwards Inspector
Fills operations card
The details of all operations that are to be carried out are recorded on this operations card
AT
Holding section
THEN
Delivered entity state
TRANSITIONS TO
EITHER
Reject entity state
OR
PreDrill entity state
OR
PreMill entity state
This transition is physically executed by the Goods Inwards Inspector who delivers each pallet of parts to the respective holding areas, which is dependant on the details of operations entered on the operations sheet. Whether or not a mill or drill operation or both has to be performed
AND
Delivered information state
TRANSITIONS TO
EITHER
Reject information state

	OR
	PreDrill information state
	OR
	PreMill information state
	This transition is physically executed by the Goods Inwards Inspector who delivers each pallet of parts to the respective holding areas, which is dependant on the details of operations entered on the operations sheet. Whether or not a mill or drill operation or both has to be performed

Table A.2 Elaboration description for the Delivery Area

A.3 Drilling Station

The drilling station consists of an index-drilling machine, which is operated by a single operator. The operator initially takes parts from the holding area between the delivery and machining areas. Only parts which are held in the drilling section of the holding area and which have a valid operations card specifying that a drilling operation has to be carried out can have a drilling operation carried out on them. Each of the pallets in the holding area consists of 100 parts. The operation carried out consists of loading the parts onto an index-drilling machine and then allowing the machine to undertake the full cycle before unloading the part, as shown in Figure A.4. Table A.3 includes an elaboration description of the Drilling Area.

The following is how the operator of this machine describes his job.

"I initially collect a pallet of parts from the drilling holding are,; there is no particular order to the picking of the parts. I generally pick from the largest section of the queue however the section for rework at both sections takes precedence over the rework at my section alone. On return to the machine I load each part onto an indexing head with five loading stations. At the end of each cycle I unload a part and place it on a pallet while at the same time reloading the next part to be processed this generally takes about 10 minutes to complete. Having completed each pallet I either bring the pallet to the milling holding area for milling or pass it onto the inspection holding area if my drilling operation was all that was necessary. I also inspect the drill bits at the end of each cycle and if necessary replace the bits. If one bit is out of size I replace all bits at the same time as per regulations."

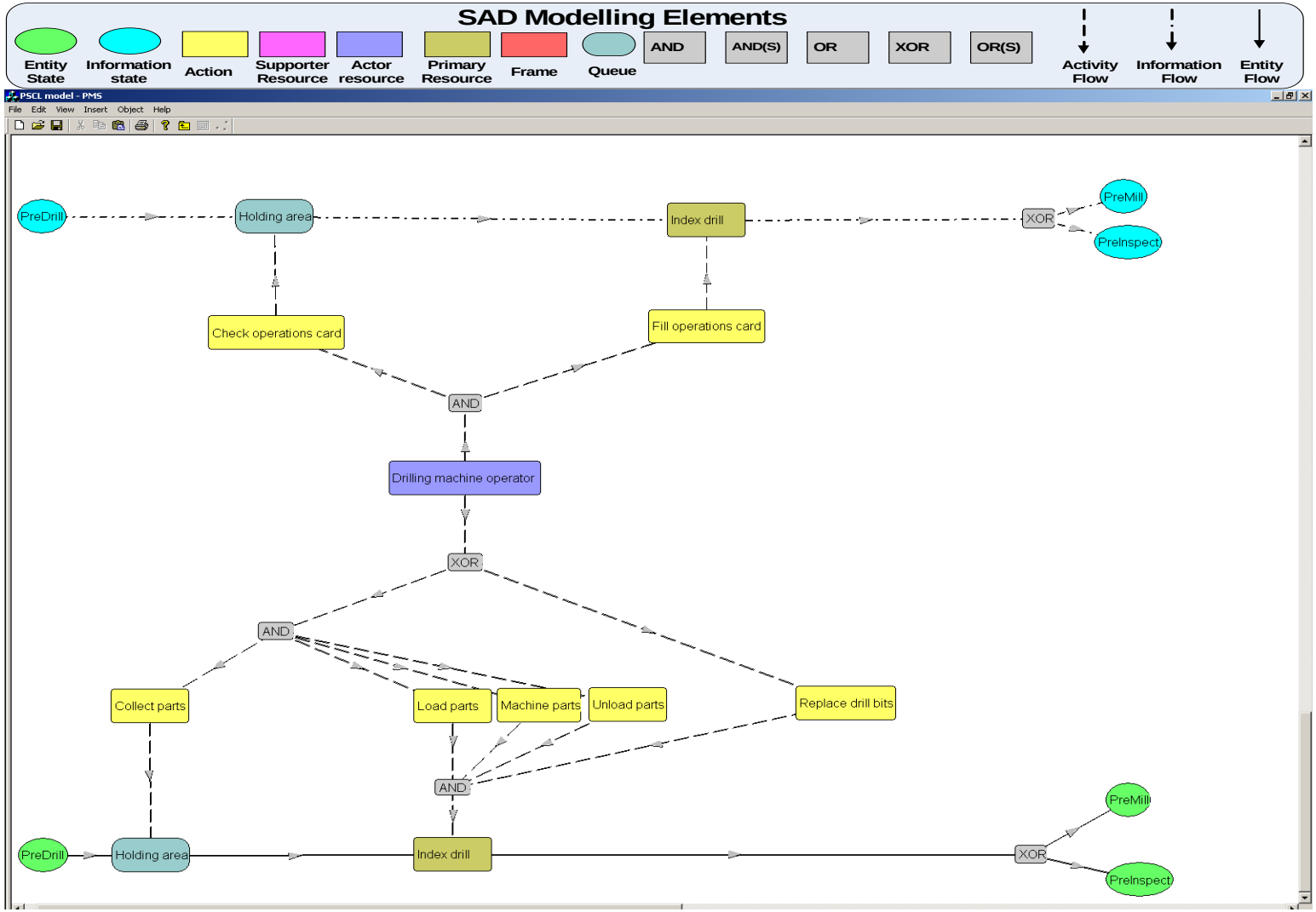

Figure A.4 Drilling Area

Elaboration of the Activity
Drilling machine operator
EITHER
Collect parts
AT
Holding area
AND
Load parts
AND
Machine parts
AND
Unload parts
AT
Index drill
OR
Replace drill bits
AT
Index drill
Drill bits are inspected at the end of each cycle and if any drill bit is outside the limits all are replaced. On examination this replacement happens on average every 500 parts or 5 batches. There is not a standard time given for this operation, however on observation of the process over a number of days the start, end, process times and overall averages were recorded as were any outlier recordings. These results are recorded in the replace drill bits excel spreadsheet. As a result of this process the average time to execute this process was found to be 5.6 minutes.
AND
Drilling machine operator
Check operations card
AT
Holding area
AND
Fill operations card
AT
Index drill
AND
THEN
PreDrill entity state
TRANSITIONS TO
EITHER
PreMill entity state
OR
PreInspect entity state
This transition is physically executed by the drilling operator who delivers each pallet of parts to the respective holding areas, which is dependant on the details of operations entered on the operations sheet. Whether or not a mill has to be performed
AND
PreDrill information state
TRANSITIONS TO
EITHER
PreMill information state

OR
Pre inspect information state
An information state"s transition is dependant on the information that is contained on the operations cards that accompany each batch of parts.

Table A.3. Elaboration Description of the Drill Area

A.4 Milling Machines

The milling machine section consists of two vertical head-milling machines, which are used to carry out a milling operation on the parts. Two operators who each operate a single milling machine operate the section. Parts are collected from the milling work section with rework batches taking precedence over standard parts. In this case parts are collected from the milling section of the rework holding area and processed on each milling machine, as shown in Figure A.5. Table A.4 contains the elaboration language description for the Milling area.

The following is the description given by one of the two section operators;

> ***"The parts to be processed are collected from the milling holding area and then loaded one at a time onto the milling machine. To set the part I firstly load it onto a four-jaw chuck and then using a dial gauge I adjust the positioning of the part for off-centre milling. This set-up operation is difficult and as a result the time for processing each part varies quiet a lot, therefore I would not be able to give you even a near ball park figure for the average processing time. However, having set the parts up on the milling machine the rest of the operation is quite standard. Having finished machining I unload the part and place it on a pallet. Having processed a hundred parts (a pallet) I transfer the parts to the inspection holding area. I also carry out a cleaning operation on the milling machine, this consists of cleaning the swarf from the machine, topping up the cutting fluid, and replacing the cutter bit."***

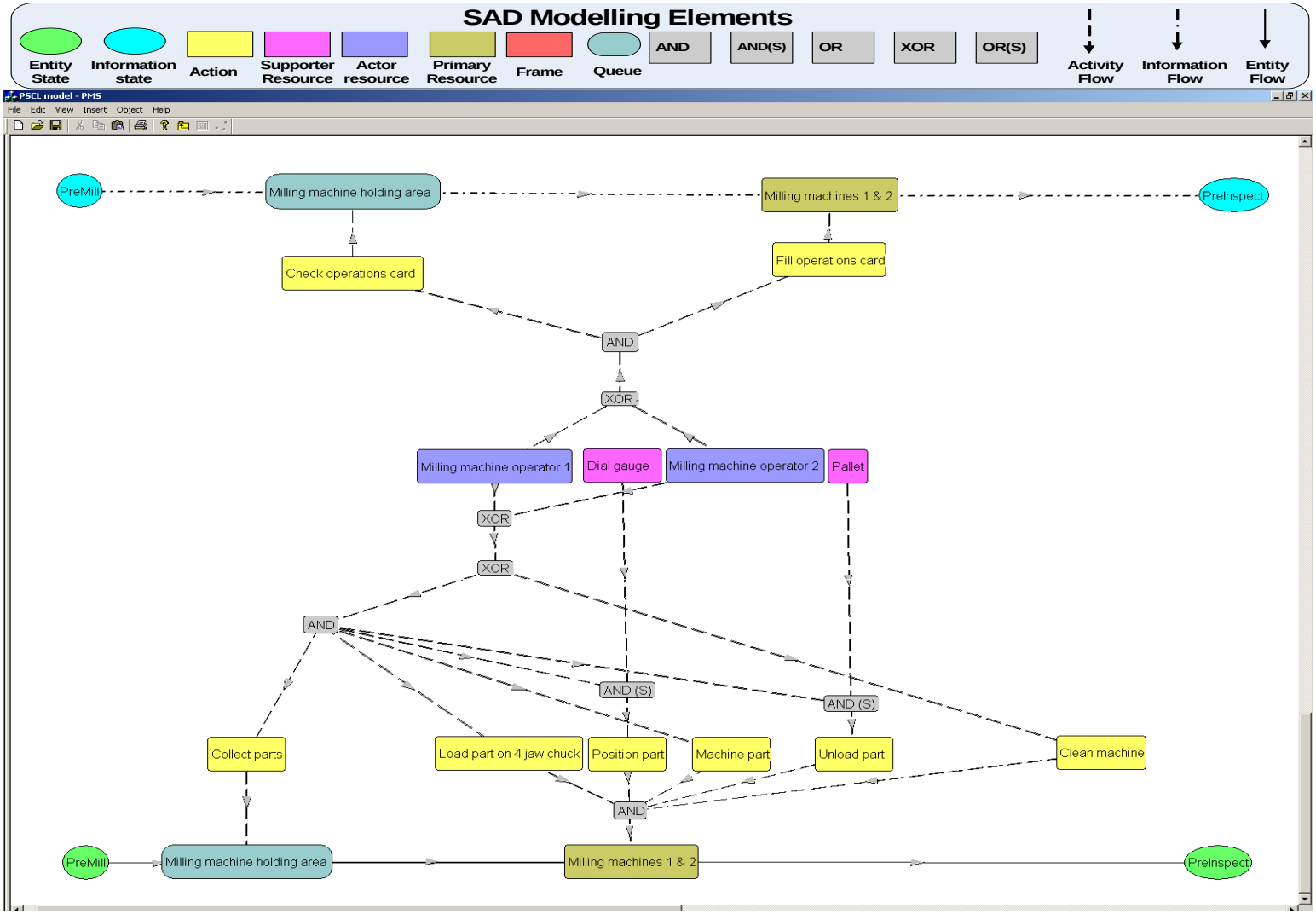

Figure A.5 Milling Area

Elaboration of the Activity
EITHER
Milling machine operator 1
OR
Milling machine operator 2
EITHER
Collect parts
AT
Milling machine holding area
AND
Load part on 4 jaw chuck
AND
USES
Dial gauge
TO
Position part
Time for this vary and are recorded in an "Position part times" excel spreadsheet attached to the position part action. On observation this operation was seen to vary significantly with an average setup time of 9.4 minutes. Refer to "Position part times" for further details.
AND
Machine part
Average for this operation is 9 minutes as shown in an excel attachment
AND
USES
Pallet
TO
Unload part
AT
Milling machines 1 & 2
OR
Clean machine
AT
Milling machines 1 & 2
AND
EITHER
Milling machine operator 1
OR
Milling machine operator 2
Check operations card
AT
Milling machine holding area
AND
Fill operations card
AT
Milling machines 1 & 2
AND
THEN
PreMill entity state
TRANSITIONS TO
PreInspect entity state

AND	
PreMill information state	
TRANSITIONS TO	
PreInspect information state	
	The milling machine operators deliver each completed batch of parts to the inspection and rework area. This results in both the entity and information states transitioning to states of pre inspect.

Table A.4. Elaboration language description of the Milling Area

A.5 Inspection

The inspection area consists of an inspection table where one operator inspects every part passing through the station. If the parts pass the inspection of the operator they are placed directly on a pallet for transfer to the packaging area. If the parts are found to be oversized they are placed on a pallet for disposal. If the parts are found to be under sized they are placed on pallets for transfer to the rework section of the delivery holding area. The inspection area is modelled as shown in Figure A.6, with elaboration language description of this area being contained in Table A.5.

The following is the description given by the inspection operator;

> ***"Parts are placed into the inspection buffer from there I pick and inspect all parts. The inspection is a simple operation where I check the critical dimensions of each piece using a height gauge and a vernier calliper, the quality of the surface finish is also tested using an electronic surface tester. On the basis of these two tests I decide if a part needs to be reworked or not. If the part does not need to be reworked it is placed on a pallet for transportation directly to the packaging area. Where the part needs rework it is placed on a either a pallet for milling rework operations, drilling rework operations or both, for transport to the necessary holding section on completion of a batch of 100 parts. Oversized parts are also placed on a pallet for dumping."***

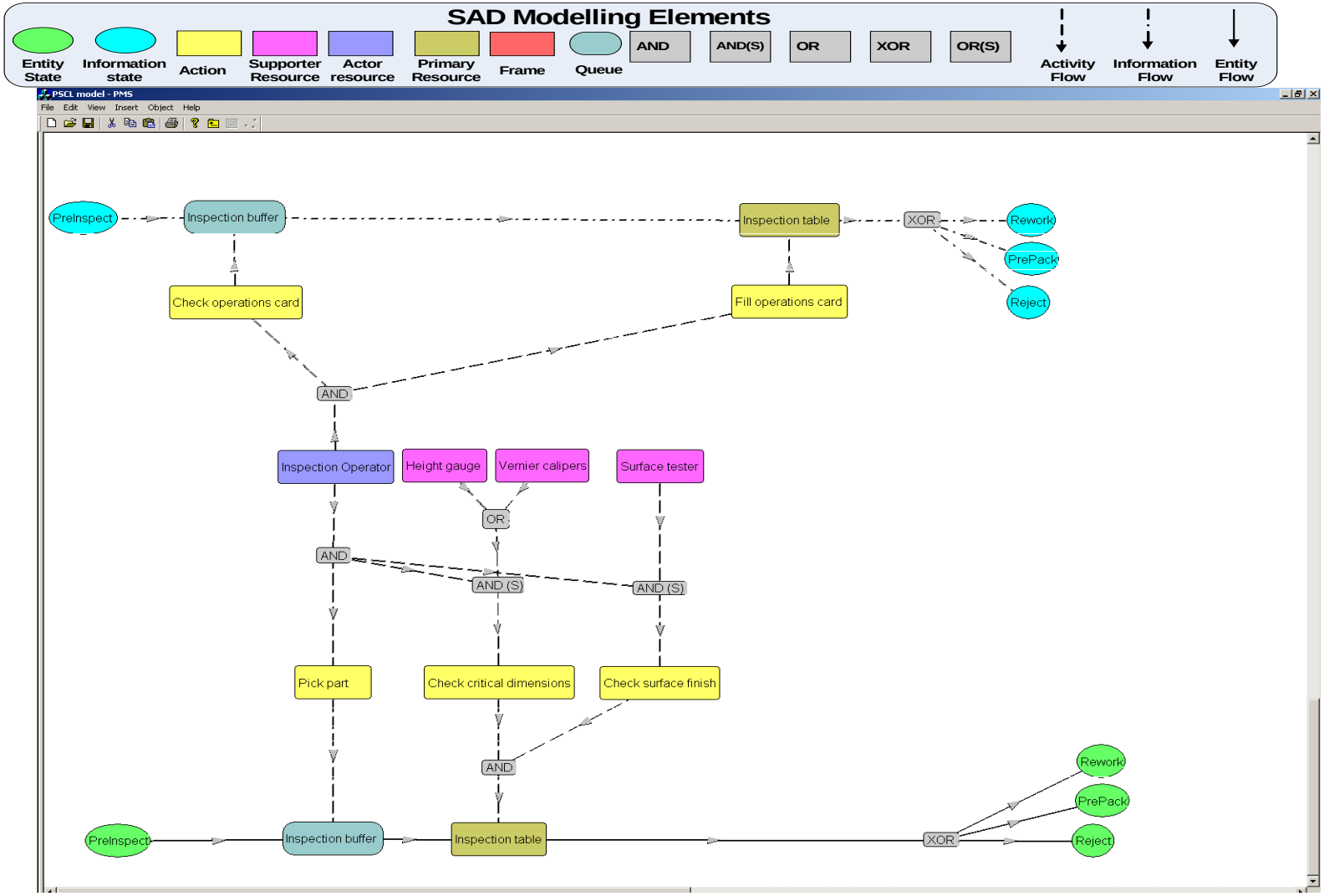

Figure A.6 Inspection Area

Elaboration of the Activity
Inspection Operator
Pick part
AT
Inspection buffer
The Inspection buffer treats parts in a First In First Out (FIFO) manner
AND
USES
Height gauge
OR
Vernier calipers
The details of the critical dimension tests performed on the parts in the Inspection area are contained in the attached document (Dimension_tests.doc)
TO
Check critical dimensions
The setup times for this operation average 1.36 mins and the details of this are recorded in the attached document (Dimension_test_setup.xls) The average time taken for this operation is 5.8 mins, with the details contained in the attached document (Dimension_Op_Times.xls)
AND
USES
Surface tester
The details of the Surface finish tests performed on the parts in the Inspection area are contained in the attached document (Surface_tests.doc)
TO
Check surface finish
The setup times for this operation average 2.56 mins and the details of this are recorded in the attached document (Surface_test_setup.xls) The average time taken for this operation is 3.2 mins, with the details contained in the attached document (surface_Test_Times.xls) The Mean Time to Failure (MTF) and the Mean Time to Repair (MTR) for this operation are attached in the following documents respectively

(Surface_test_MTF.xls) (Surface_test_MTR.xls)
AT
Inspection table
AND
Inspection Operator
Check operations card
AT
Inspection buffer
AND
Fill operations card
AT
Inspection table
THEN
PreInspect entity state
TRANSITIONS TO
EITHER
Rework entity state
OR
Prepack entity state
OR
Reject entity state
This transition is based on the results of the tests carried out on the parts by the inspection operator.
AND
PreInspect information state
TRANSITIONS TO
EITHER
Rework information state
OR
PrePack information state
OR
Reject information state
The transition here represents the transition of the operations card, which details each operation and in the case of the inspection operation, the outcome of the operation, which accompanies each batch of parts through the system.

Table A.5. Elaboration language description for the inspection area

A.6 Packaging Area

This section consists of two automatic packaging machines in sequence. The first machine wraps the finished parts while the second seals them in an airtight vacuum pack. The machines automatically pack the parts and are manned by a single operator who keeps both machines operating. The graphical representation for this is shown in Figure A.7. With the accompanying elaboration language description being included in Table A.6. This operator now describes his job;

> ***"The parts are picked from the Packaging holding area and placed on the first packaging machine. As soon as this machine finishes processing the part is transferred immediately to the second machine which is set in motion. At this stage I reload the first packaging machine, this operation continues. The machines do not need to be cleaned and are capable of being replenished without stopping. As the part is finished processing on the second machine it is placed on a pallet. When the pallet is finished it is sent directly to shipping."***

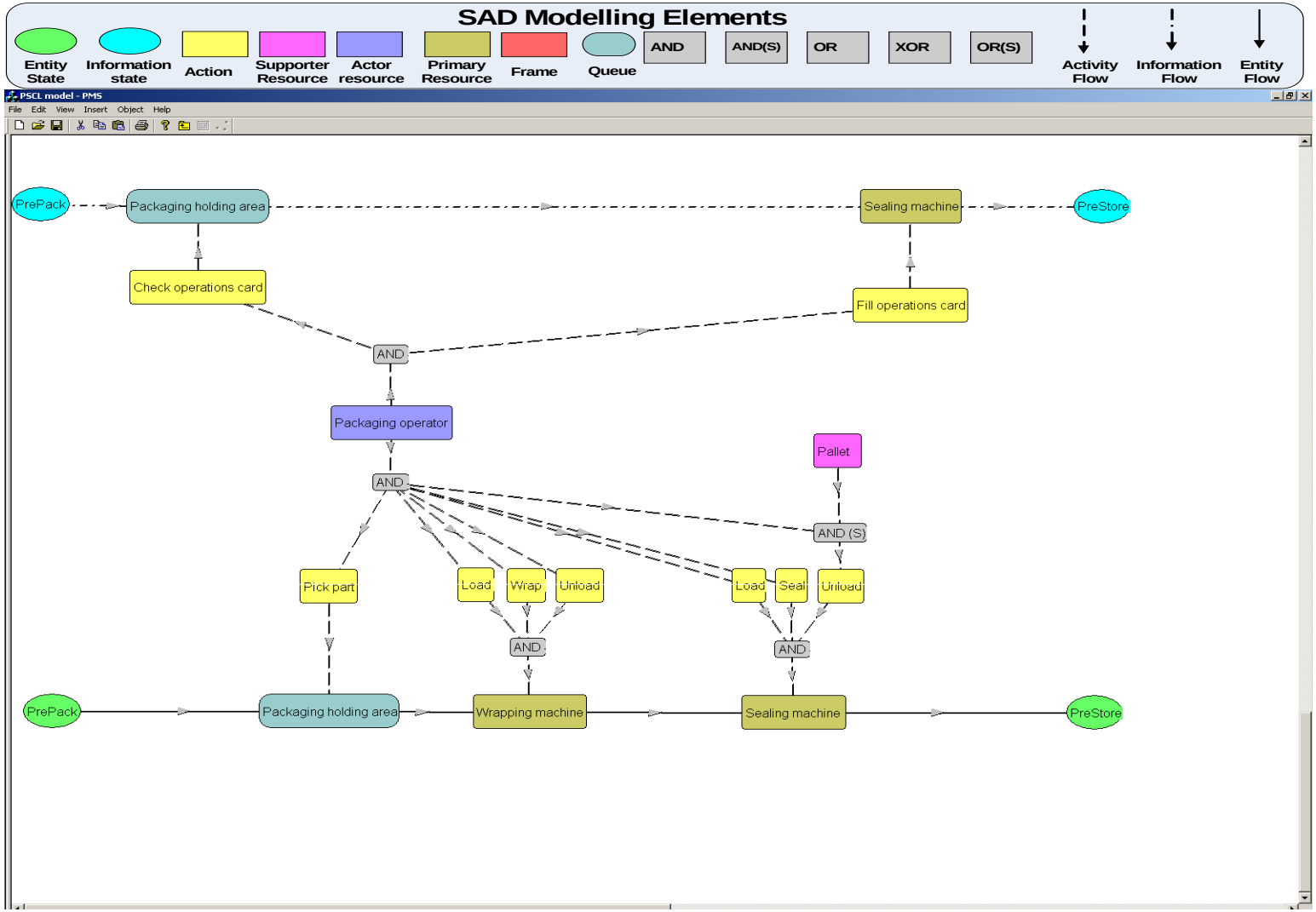

Figure A.7 Packaging Area

Elaboration of the Activity
Packaging operator
Pick part
AT
Packaging holding area
AND
Load
AND
Wrap
AND
Unload
AT
Wrapping machine
AND
Load
AND
Seal
AND
USES
Pallet
TO
Unload
AT
Sealing machine
AND
Packaging operator
Check operations card
AT
Packaging holding area
AND
Fill operations card
AT
Sealing machine
THEN
PrePack entity state
TRANSITIONS TO
PreStore entity state
AND
PrePack information state
TRANSITIONS TO
PreStore information state.

Table A.6 Elaboration language for the Packaging area

A.7 Warehousing

This area consists of a small warehouse where parts are held until being shipped. Parts are stored by the warehouse operator who fills orders as per the order slips which arrive daily, notifying him of the quantities of each part type which are required each day. The graphical representation of this situation is shown in Figure A.8, with the elaboration language description included in Table A.7. In the following section he outlines his work;

> ***"As the packed parts arrive from the packaging area I place them on the required racks in their designated storage areas. Each morning I receive the orders for the day, which I prepare for shipping from the stock in hand and dispatch as required. For this I have a forklift truck to allow me to load the orders onto the trucks for dispatch."***

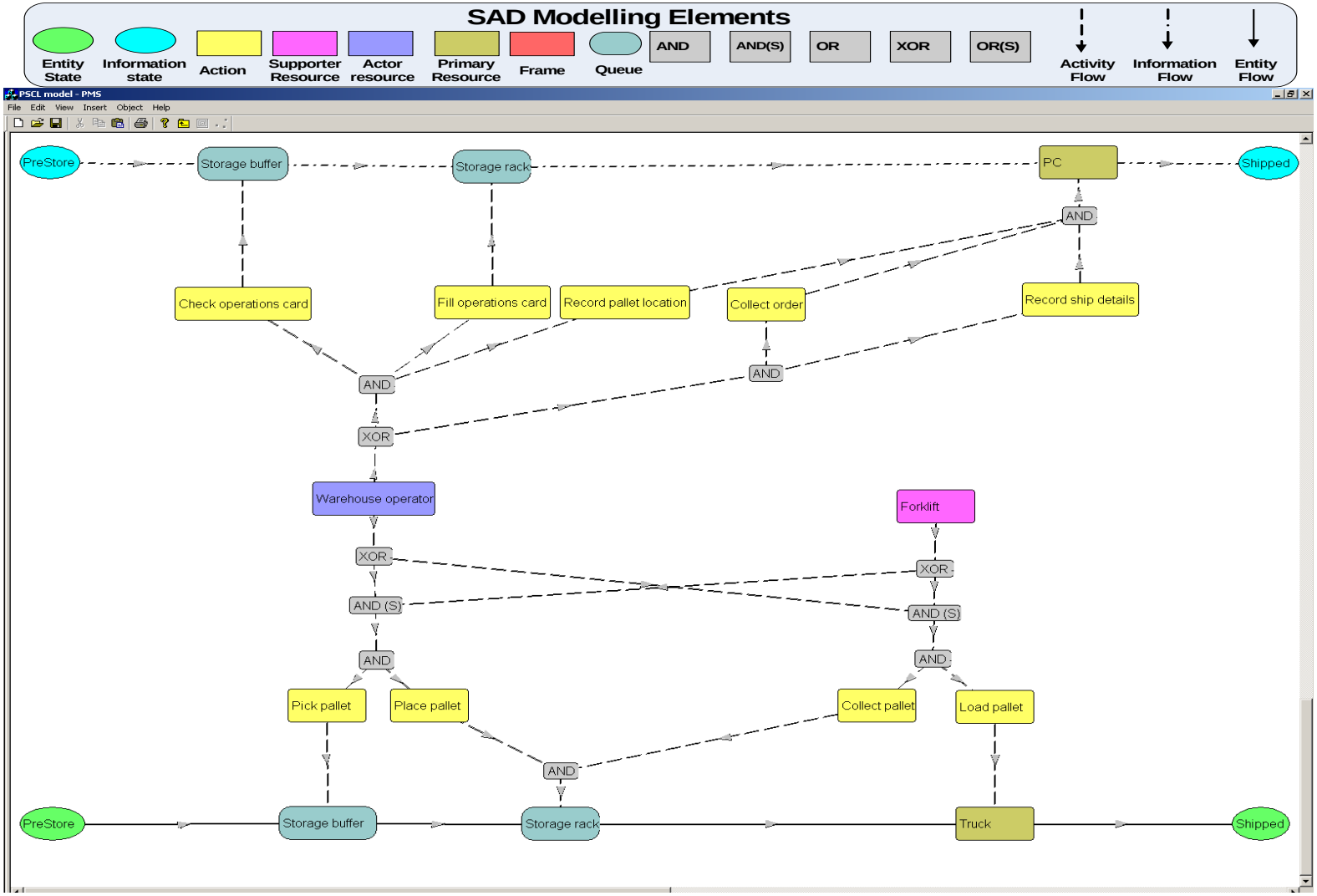

Figure A.8 Warehousing Area

Elaboration of the Activity
Warehouse operator
EITHER
USES
Forklift
TO
Pick pallet
AT
Storage buffer
AND
Place pallet
AT
Storage rack
OR
USES
Forklift
TO
Collect pallet
AT
Storage rack
AND
Load pallet
AT
Truck
AND
Warehouse operator
EITHER
Check operations card
AT
Storage buffer
AND
Fill operations card
AT
Storage rack
AND
Record pallet location
AT
PC
OR
Collect order
AND
Record ship details
AT
PC
THEN
PreStore entity state
TRANSITIONS TO
Shipped entity state
AND
PreStore information state
TRANSITIONS TO
Shipped information state.

Table A.7 Elaboration Language Description for the Warehousing area

Appendix B: Boart SAD Model

The work region modelled in this example relates to the carburising or induction-hardening phase of the production process within a company producing mining consumables. It is on modelling the operator control and decision making processes within this area that the following SAD example will concentrate.

B 1 Carburising Area

Rods that require carburising are staged in the carburising area, until a sufficient quantity of rods required for the specific carburising setting are ready to be loaded onto a carburising jig for placement in the carburising furnace.

Before the rods are carburised certain preparatory operations are performed, e.g. inserting a carburising rope. To enter the furnace the rods are manually loaded onto a carburising jig. The carburising jig consists of a column, attached to which, at varying intervals is a six sectioned “spider”. Placed within each section of this “spider” is a honeycomb tray, which allows the rods to be hung vertically in each section. The spider, honeycomb trays and rods contained therein are collectively known as a “tier”. The length of the rods being carburised determines the number of tiers on the jig. For very long rods only 1 tier is useable, for very short rods four tiers can be used. The diameter and shape of the rods determine the type of honeycomb tray that is used.

When the jig has been filled to capacity or near capacity, the operators use a crane to place the loaded jig into the furnace. The carburising furnace operates on a number of different carburising settings depending on the type of rods to be carburised. After the jig containing the rods is carburised, it must be transferred immediately to the cooling tower to be cooled under controlled conditions to ensure the required hardness is achieved by the carburising process. After the cooling tower the operators allow the jig to air cool until the rods are cool enough to be unloaded. The unloading operation is a manual operation where the parts are unloaded and passed to the next work region. The following section presents a SAD diagram developed to communicate the various interactions between the operators and the manufacturing system. In

this system, parts arrive into the furnace area and wait until all operations such as roping, application of anti-carburising paint and stamping of the batch number have been performed. At this point the parts are split-up into separate sections based on the carburising setting. Within each of these carburising setting sections there are four further holding sections based on the product length. It is from these areas that a jig is built. A jig is made up of tiers of rods, of which there are a maximum of four on each jig. Each tier has six trays containing honeycombs into which rods are slotted. There are four types of trays:

- **Type A** Can hold a maximum of 16 rods;
- **Type B** Can hold a maximum of 12 rods;
- **Type C** Can hold a maximum of 9 rods;
- **Type D** Can hold a maximum of 3 rods.

It is also possible to build a jig containing trays of more than one type. A jig has a maximum length of 20 feet (6.1m) and can be placed in the furnace on completion of building, providing the furnace is free. If parts are in the holding area for more than eight hours and there are not enough parts available of the particular type to build an entire jig, then partially built jigs may be used.

The maximum numbers of rods that can be arranged on a jig are detailed in Tables B 4 to B 6. Table B 4 assumes that all rods on the jig are the same. This does not have to apply in reality. Provided that the rods all have the same carburising cycle code, a jig can contain rods of varying types, lengths, diameters and shapes. There can even be different tray types on a single tier.

There are two furnace operators who are required to carry out the following operations:

- Load/unload the furnace;
- Load/unload the air cooling tower;
- Build/dismantle a jig;
- Pre-jig building operations.

Pre-jig building operations consist of inserting rayon ropes, applying anti carburising paint and stamping the batch number on parts. The unloading of a jig takes thirty minutes and is taken as the highest priority or most important job within the furnace area, the operation descriptions and their priorities are shown in Table B 1.

Priority	Description
1	Unloading the furnace
2	Building a jig (To ensure there is always a jig available)
3	Dismantle a jig
4	Load/Unload the air cooling tower
5	Pre-jig building operations

Table B 1 Furnace operation priorities

Unloading the furnace occupies the operators for 30 minutes. This task is assigned the highest priority in the model and therefore, whenever it occurs the operators stop working on all other tasks and are pulled to the furnace. Building or dismantling of jigs is given the next highest priority. All other tasks have very low priority and cannot be started unles the aforementioned operations are not possible. Operators will attempt to build a jig before dismantling one so as to ensure that a jig will be available when the furnace requires one. However, jigs are a limited resource in that there are only three jigs in the furnace area. Also, jig building may not be complete when the furnace next becomes empty. The manning requirements to load and unload a jig are shown in Tables B7 to B10.

s

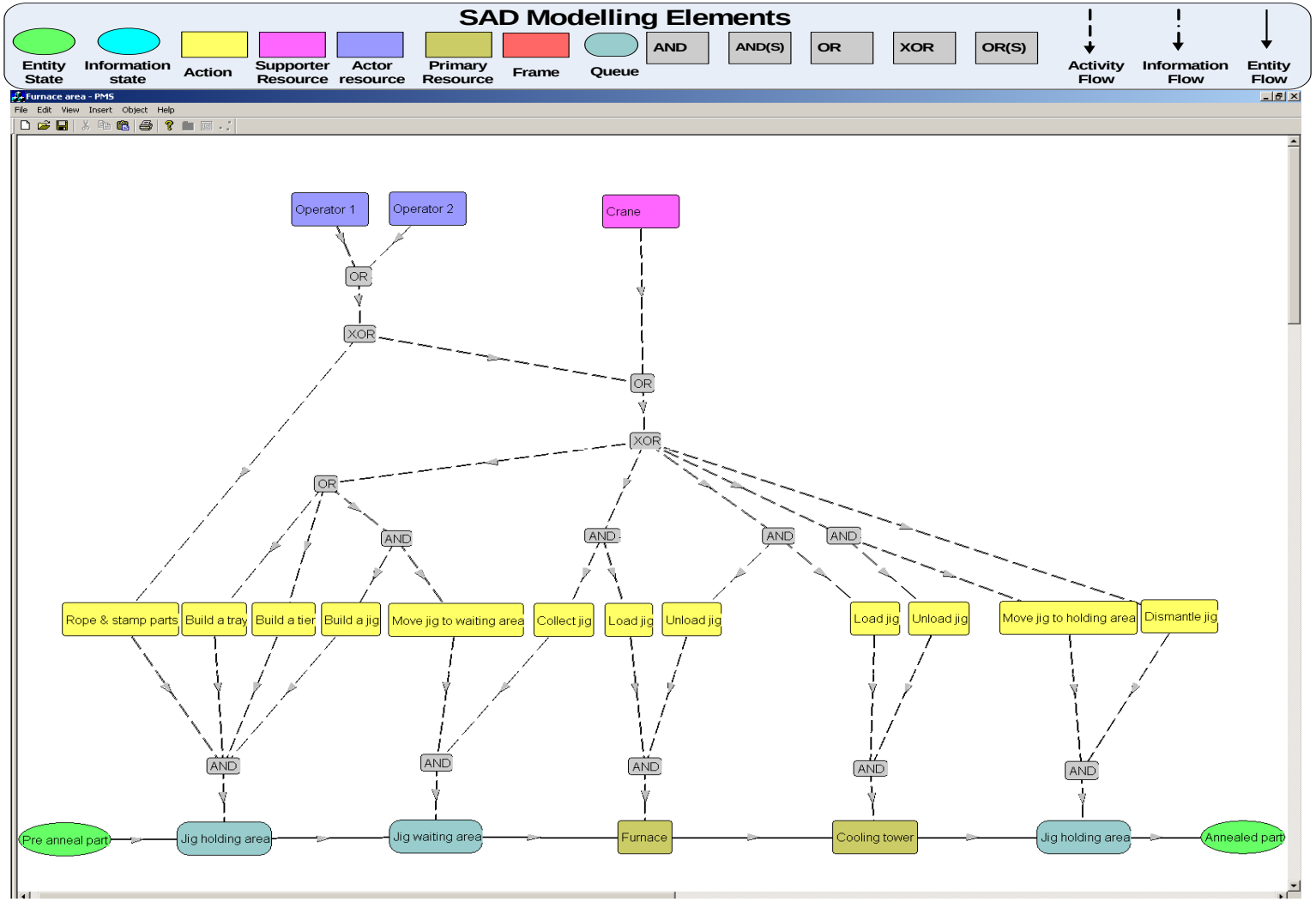

Figure B.1 Furnace area SAD

Elaboration of the Activity
Operator 1
OR The number of operators and need for a crane is dependant on the size of parts being placed on the tray/tier or jig. Details are contained in the following four attached documents. (Load-requirements-hex-rods.xls) (Load-requirements-round-rods.xls) (Unload-requirements-hex-rods.xls) (Unload-requirements-round-rods.xls)
Operator 2
EITHER The operations are outlined here in the sequence of execution to produce a part, however priority rules apply to the sequence of operations within the area and these priority rules are contained in an attached document (Furnace-operation- priorities.doc)
Rope & stamp parts
OR
OR
USES
Crane
The number of operators and need for a crane is dependant on the size of parts being placed on the tray/tier or jig. Details are contained in the following four attached documents. (Load-requirements-hex-rods.xls) (Load-requirements-round-rods.xls) (Unload-requirements-hex-rods.xls) (Unload-requirements-round-rods.xls)
TO
EITHER
Build a tray
There are four types of tray the details of which are contained in the attached document (tray-types.xls)
OR
Build a tier
A tier consists of six of trays
OR
Build a jig
A jig is made up of a maximum of four tiers and each tier is made up of a number of trays. The number of tiers and trays used and the number of parts is dependant on the size and weight of parts with maximum limits on each. The details for this are contained within the following attached documents. (Max-Furnace-utilisation.xls) (Round-rod-weights.xls) (Hex-Rod-weights.xls) While fully built jigs are preferred, parts in the holding section for longer than eight hours may be used on partially built jigs.
AT
Jig holding area

	AND
	Move jig to waiting area
	AT
	Jig waiting area
OR	
	Collect jig
	AT
	Jig waiting area
	AND
	Load jig
	AT
	Furnace
	The furnace cycle times vary with the details contained in the attached document (Furnace-cycle-times.xls)see table B 3
OR	
	Unload jig
	AT
	Furnace
	AND
	Load jig
	AT
	Cooling tower
OR	
	Unload jig
	AT
	Cooling tower
	AND
	Move jig to holding area
	AT
	Jig holding area
OR	
	Dismantle jig
	AT
	Jig holding area
AND	
THEN	
Pre anneal part entity state	
TRANSITIONS TO	
Annealed part entity state	

Table B 2 Furnace area elaboration

Frequently Used	***Setting***	***Cycle Time (Hrs.)***
	2	8.5
	7	10.5
	8	4.5
	10	8.5
	12	6.5
	14	6.5
Occasionally Used		
	3	4.5
	6	6
	11	4.5
	13	8.5
	17	10.5

Table B 3 Carburising Furnace Cycle Times

Rod Length (M)	Rod Length (Ft)	No. of Tiers	3/4" Hex collar	7/8" Hex collar	1" Hex collar	1 1/8" Hex collar	1 3/8" Hex	1 1/4" Hex collar	1 ½" Hex	1 1/4" Round	1 1/2" Round	1 3/4" Round	2" Round	44mm Tubes	Tube Rods
300mm/1.22	1,2,3,4	4	384	384	384	384	288	336	216	336	336	216	216	216	72
1.5/1.8	5,6	3	288	288	288	288	216	252	162	252	228	162	162	162	54
2.1/2.7	7,8,9	2	192	192	192	192	144	168	108	168	140	108	108	108	36
3.05	10	2	192	192	192	96	72	96	54	168	140	108	108	108	36
3.35	11	1	96	96	96	96	72	96	54	96	96	54	54	54	18
3.66	12	1	96	96	96	96	72	96	54	96	96	54	54	54	18
3.96	13	1	96	96	96	96	72	96	54	96	96	54	54	54	18
4.27	14	1	96	96	96	96	72	96	54	96	96	54	54	54	18
4.89	16	1	96	96	96	96	72	96	54	96	96	54	54	54	18
5.49	18	1	96	96	96	96	72	96	54	96	76	40	40	40	18
6.1	20	1	96	96	96	96	72	96	54	96	76	40	40	40	18

UTILISATION CHART FIGURES BASED ON: 1. Maximum payload weight equal to 8,000 lbs. **2.** Maximum number of rods on a single tier jig equals 96.

Table B 4 Maximum carburising jig utilisation chart

Rod Size	*2ft*	*4ft*	*6ft*	*8ft*	*10ft*	*12ft*	*14ft*	*16ft*	*18ft*	*20ft*
7/8"	4	8	12	16	20	25	29	33	37	41
1"	6	11	17	22	28	33	39	44	50	55
1"	6	11	17	22	28	33	39	44	50	55
1 1/8"	7	17	21	28	35	42	49	56	63	70
1 1/8"	7	17	21	28	35	42	49	56	63	70
1 1/4"	8	17	25	34	42	51	59	68	76	85
1 1/4"	8	17	25	34	42	51	59	68	76	85
1 3/8"	10	20	30	40	50	60	70	80	90	100
1 1/2"	11	23	34	46	57	69	80	92	103	115

Note: 1m =>1 man required to load/unload, 2m =>2 men required to load/unload, & c => furnace area crane required for loading/unloading.

Table B 5: Hexagonal Rod Weights in lbs. (rounded up).

Rod Size	*2ft*	*4ft*	*6ft*	*8ft*	*10ft*	*12ft*	*14ft*	*16ft*	*18ft*	*20ft*
1 1/4"	8	15	23	30	38	45	53	60	68	76
1 1/2"	11	23	34	45	57	68	79	90	102	113
1 3/4"			46		77	92	107	123	138	153
2"					100	120	140	160	180	200

Note: 1m =>1 man required to load/unload, 2m =>2 men required to load/unload, & c => furnace area crane required for loading/unloading.

Table B 6 Round Rod Weights in lbs. (rounded up).

Rod Size	*2ft*	*4ft*	*6ft*	*8ft*	*10ft*	*12ft*	*14ft*	*16ft*	*18ft*	*20ft*
7/8"	1m	1m	1m	1m	1m	1m	1m	1m	1m	1m
1"	1m	1m	1m	1m	1m	1m	1m	1m	1m	1m
1"	1m	1m	1m	1m	1m	1m	1m	1m	1m	1m
1 1/8"	1m	1m	1m	1m	1m	1m	1mc	1mc	1mc	1mc
1 1/8"	1m	1m	1m	1m	1m	1m	1mc	1mc	1mc	1mc
1 1/4"	1m	1m	1m	1m	1m	1m	1mc	1mc	1mc	1mc
1 1/4"	1m	1m	1m	1m	1m	1m	1mc	1mc	1mc	1mc
1 3/8"	1m	1m	1m	1m	1m	1mc	1mc	1mc	1mc	1mc
1 ½"	1m	1m	1m	2m	2m	1mc	1mc	1mc	1mc	1mc

Note: 1m =>1 man required to load/unload, 2m =>2 men required to load/unload, & c => furnace area crane required for loading/unloading.

Table B 7 Manning Requirements for Loading the Carburising Furnace Jig with Hexagonal Rods.

Rod Size	*2ft*	*4ft*	*6ft*	*8ft*	*10ft*	*12ft*	*14ft*	*16ft*	*18ft*	*20ft*
1 ¼"	1m	1m	1m	2m	1m	1mc	1mc	1mc	1mc	1mc
1 ½"	1m	1m	2m	2m	1m	1mc	1mc	1mc	1mc	1mc
1 ¾"			2m		1mc	1mc	1mc	1mc	1mc	1mc
2"				1mc	1mc	1mc	1mc	1mc	1mc	1mc

Note: 1m =>1 man required to load/unload, 2m =>2 men required to load/unload, & c => furnace area crane required for loading/unloading.

Table B 8 Manning Requirements for Loading the Carburising Furnace Jig with Round Rods.

Rod Size	*2ft*	*4ft*	*6ft*	*8ft*	*10ft*	*12ft*	*14ft*	*16ft*	*18ft*	*20ft*
7/8"	1m	1m	1m	1m	2m	2m	2m	2m	2m	2m
1"	1m	1m	1m	1m	2m	2m	2m	2m	2m	2m
1"	1m	1m	1m	1m	2m	2m	2m	2m	2m	2m
1 1/8"	1m	1m	1m	1m	2m	2m	2mc	2mc	2mc	2mc
1 1/8"	1m	1m	1m	1m	2m	2m	2mc	2mc	2mc	2mc
1 1/4"	1m	1m	1m	1m	2m	2m	2mc	2mc	2mc	2mc
1 1/4"	1m	1m	1m	1m	2m	2m	2mc	2mc	2mc	2mc
1 3/8"	1m	1m	1m	1m	2m	2mc	2mc	2mc	2mc	2mc
1 1/2"	1m	1m	1m	1m	2m	2m	2m	2m	2m	2m

Note: 1m =>1 man required to load/unload, 2m =>2 men required to load/unload, & c => furnace area crane required for loading/unloading.

Table B 9 Manning Requirements for Unloading the Carburising Furnace Jig with Hexagonal Rods.

Rod Size	*2ft*	*4ft*	*6ft*	*8ft*	*10ft*	*12ft*	*14ft*	*16ft*	*18ft*	*20ft*
1 ¼"	1m	1m	1m	1m	2mc	2mc	2mc	2mc	2mc	2mc
1 1/2"	1m	1m	2m	2m	2mc	2mc	2mc	2mc	2mc	2mc
1 3/4"			2m		2mc	2mc	2mc	2mc	2mc	2mc
2"				2mc	2mc	2mc	2mc	2mc	2mc	2mc

Note: 1m =>1 man required to load/unload, 2m =>2 men required to load/unload, & c => furnace area crane required for loading/unloading.

Table B 10 Manning Requirements for Unloading the Carburising Furnace Jig with Round Rods.

Tray type	*Capacity*
Type A	16 rods
Type B	12 rods
Type C	9 rods
Type D	3 rods

Table B 11 Tray types and maximum capacity

Appendix C: SAD model of a production line

C.1 Product Description

The production line modelled here is used for the manufacture of diamond cutter discs. The SAD technique is used to model the overall production line, by giving an overview of the line and then providing more detailed information on the various production areas, represented by their own individual SAD diagrams. There are 17 products, which are processed on the 74mm Syndite line, the product codes for these are can be seen in Table C.1.

Name	Product Code	Item No.
1	USYR7416 – 36005 002	HC000519
2	USYR7416 – 36005 010	HC000122
3	USYR7416 – 36005 025	HC000500
4	USYR7419 – 36005 002	HC000520
5	USYR7419 – 36005 010	HC000510
6	USYR7419 – 36005 025	HC000502
7	USYR7420 – 36005 002	HC000521
8	USYR7420 – 36005 010	HC000123
9	USYR7420 – 36005 025	HC000501
10	USYR7432 – 36005 002	HC000522
11	USYR7432 – 36005 010	HC000124
12	USYR7432 – 36005 025	HC000503
13	USHR7416 – 36005 025	HC000512
14	USHR7419 – 36005 025	HC000511
15	USHR7420 – 36005 025	HC000513
16	USHR7432 – 36005 025	HC000514
17	USQR7480 – 36007 025	HC000518

Table C.1 74mm Syndite Line Products and Item Numbers.

The product code gives the details of the part. For example:

- U SY R 74 16 – 360 05 002;
- U signifies that the part is for buffer stock;
- SY signifies the type of part (SY = Syndite type CTB, SH = Syndite type CTH, SC = Syndite type CTC and SQ = Syndril);
- R signifies that the part is a round part (i.e. a whole disc);
- 74 signifies the usable area of the disc (i.e. 74mm);
- 16 signifies the thickness of the disc (i.e. 1.6mm);
- 360 signifies that the part is a complete 360°;
- 05 signifies the diamond layer thickness (i.e. .05mm);
- 002 signifies the diamond grade.

Every part which flows through the 74mm Syndite line follows an identical route. Table C 2 gives an outline of the routing for the line, the actual

processes used, the number of machines in each area, the number of operators per shift, the number of shifts per process and the number of discs, which are processed on each machine in one run.

As every part modelled in this particular production area follows the same routing it is possible to model all parts with one part family used to represent this scenario. This approach is supported in the PMS modelling software, which allows for the modelling of part families as introduced in chapter 4.

Op. No.	Process	Machines	Operators	Shifts	Process batch per m/c
10	Centreless Grind	1	1	2	10
20	Face Grinding	2	1	3	1
30	Surface Grinding	2	1	3	10
40	EDM Planning	11	2	3	9
50	Finish Lapping	4	1	3	40
60	Sandblasting	1	1	1	1
70	Assessment	2	6	1	36

Table C 2 74mm Syndite Process.

The high level SAD for this line is presented in Figure C.1, in this diagram the various areas within the production line are graphically represented by frame elements. The elaboration language associated with this diagram is presented in Table C.3.

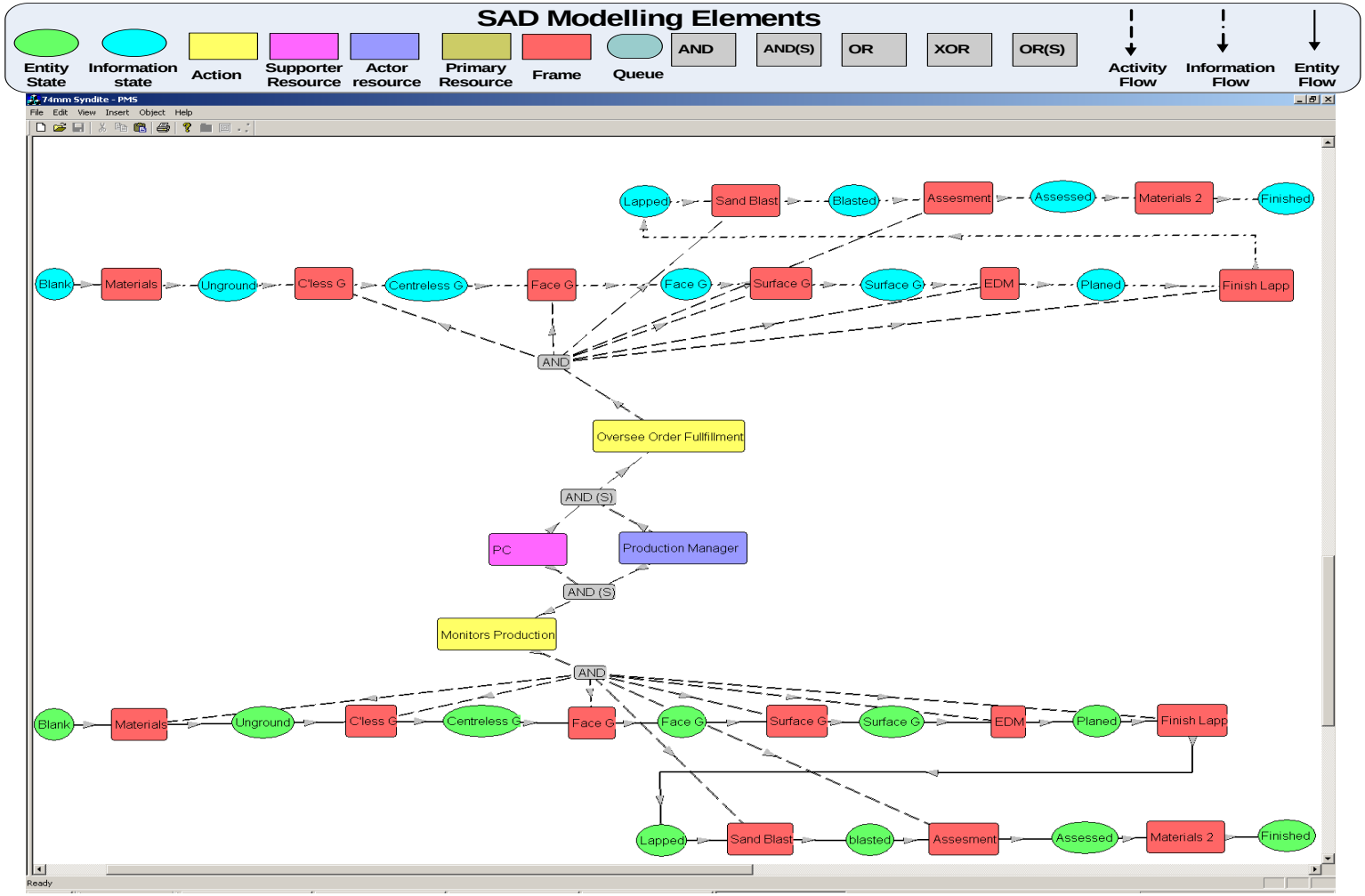

Figure C.1 74mm syndite top level SAD

Elaboration of the Activity
Production Manager
USES
PC
TO
Monitors Production
AT
Materials
AND
C”less G
AND
Face G
AND
Surface G
AND
EDM
AND
Finish Lapp
AND
Sand Blast
AND
Assessment
AND
Materials 2
AND
Production Manager
USES
PC
TO
Oversee Order Fullfillment
AT
Materials
AND
C”less G
AND
Face G
AND
Surface G
AND
EDM
AND
Finish Lapp
AND
Sand Blast
AND
Assessment
AND
Materials 2
THEN
Blank entity state
TRANSITIONS TO
Finished entity state
AND
Blank information state
TRANSITIONS TO
Finished information state

Table C.3 74mm syndite top level Elaboration

C.2 Materials control

Batches travel through the line in multiples of 40. These batches are stored in colour-coded bins, with different bins representing different diamond grades. Each batch has a shop floor traveller, which also follows the part through the line. The raw material for the line is held in Materials Control as rough discs from the diamond synthesis process. The line manager brings the material to the first operation and places it on a table awaiting processing. The batches then pass through their appropriate processes in relation to its routing. Table C.4 and Figure C.2 show the elaboration and SAD diagram for this area.

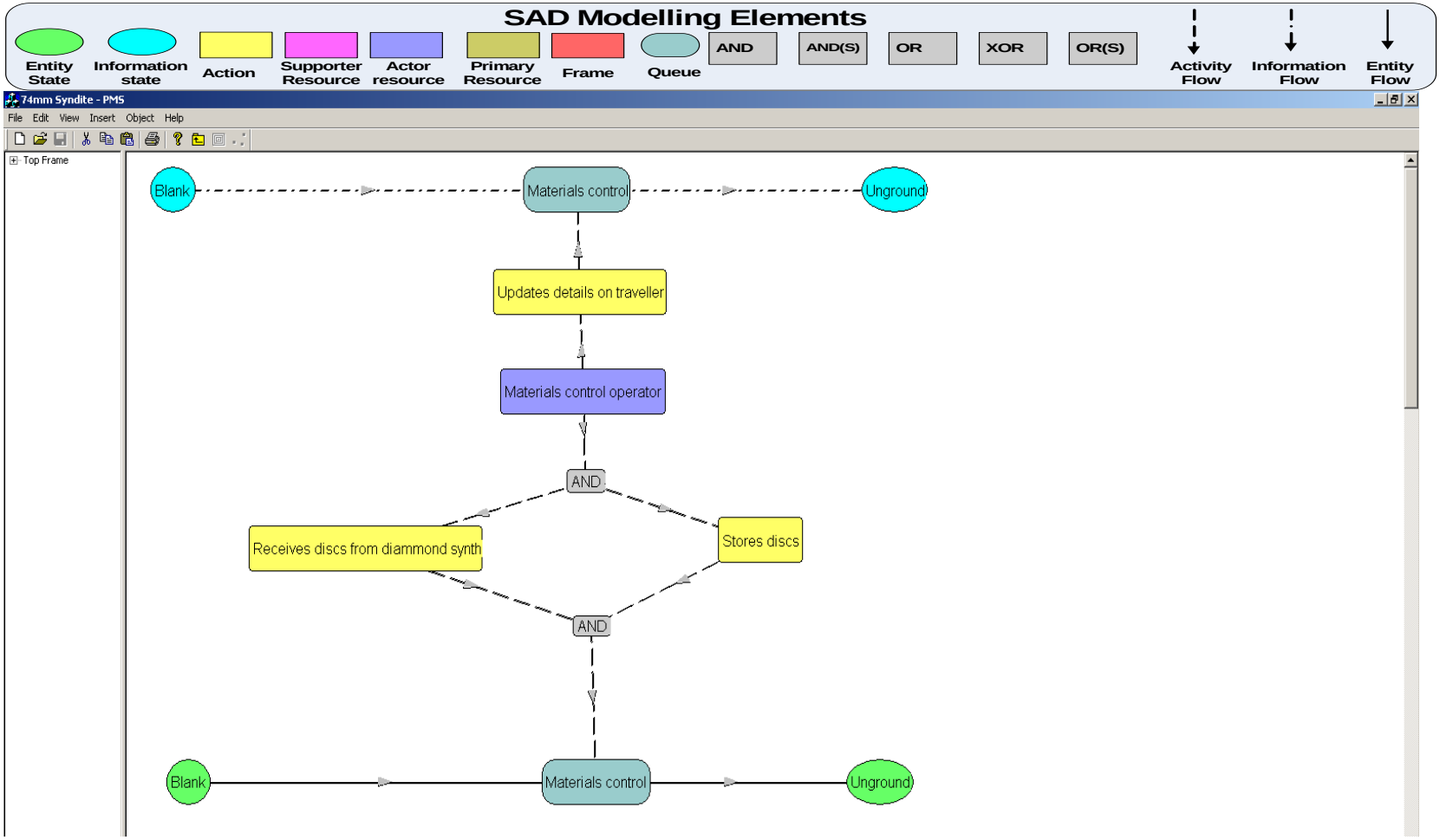

Figure C.2 Materials Control SAD

Elaboration of the Activity
Materials control operator
Receives discs from diamond synth
AND
Stores discs
At
Materials control
AND
Materials control operator
Updates details on traveller
AT
Materials control
THEN
Blank entity state
TRANSITIONS TO
Unground entity state
AND
Blank information state
TRANSITIONS TO
Unground information state

Table C.4 Materials Control Elaboration

C.3 Centreless grinding

The first operation, operation 10 is Centreless Grinding, where the edge around the material is ground down to a set dimension. There is one dedicated machine for this task, and a single operator who is shared between this machine and a similar machine, which is used for the 57mm Syndite line. The number of discs processed on this machine in one run, depends on the thickness of the discs. This is because there is a width capacity for holding discs. The machine can hold 10 of the 1.6mm discs. So it will take four runs of the machine to finish these batches. This machine is run over two shifts and the expected output per shift is 120 discs per shift or a cycle time of 40 minutes per run of the centreless grinder. The elaboration and SAD for this area are shown in Table C.5 and Figure C.3 respectively.

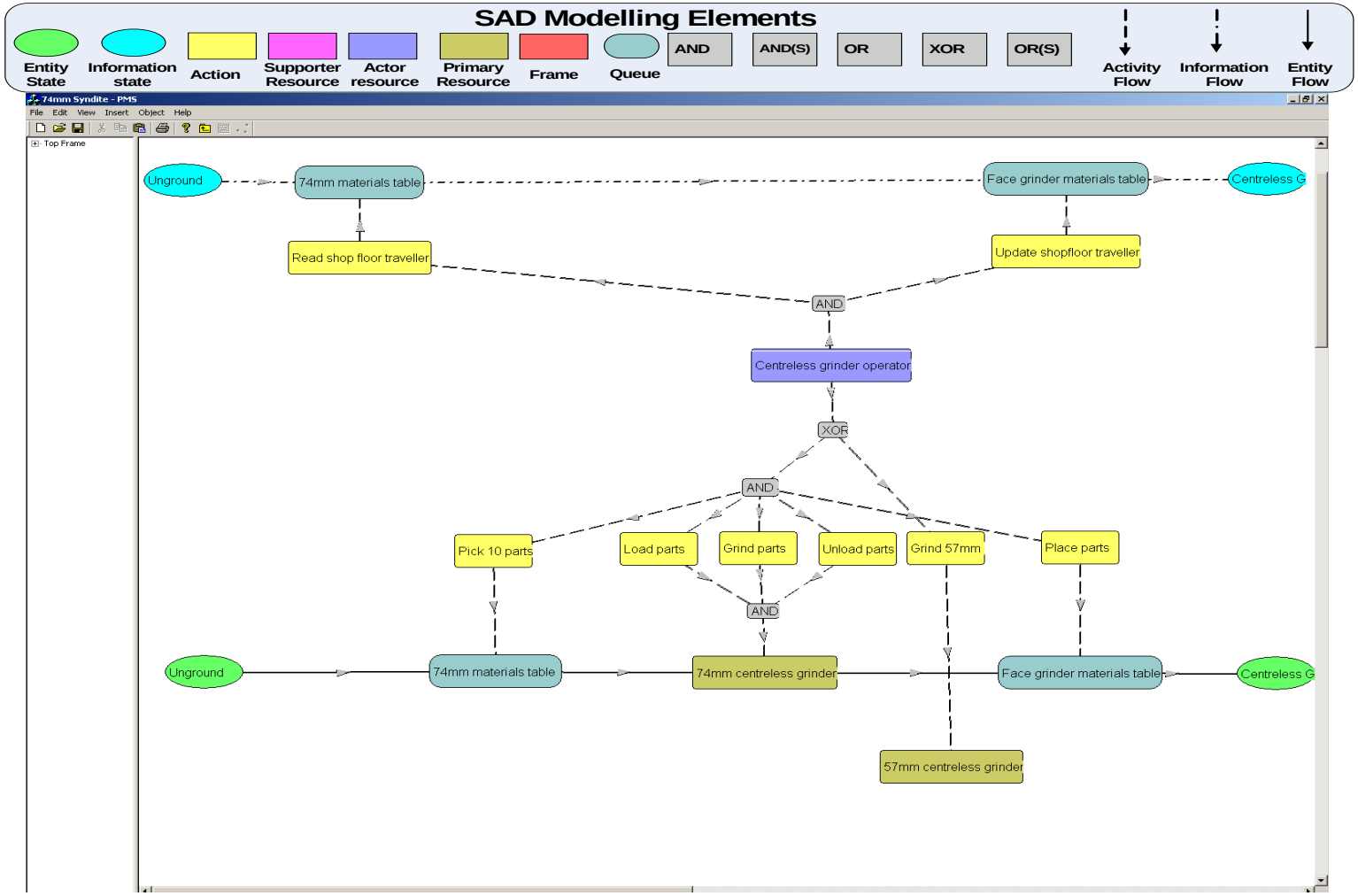

Figure C.3 Centreless Grinding

Elaboration of the Activity.
Centreless grinder operator
EITHER
Pick 10 parts
AT
74mm materials table
AND
Load parts
AND
Grind parts
AND
Unload parts
AT
74mm centreless grinder
This machine has a standard cycle time of 40 minutes.
AND
Place parts
AT
Face grinder materials table
OR
Grind 57mm parts
ON
57mm centreless grinding machine
AND
Centreless grinder operator
Read shop floor traveller
AT
74mm materials table
AND
Update shopfloor traveller
AT
Face grinder materials table
THEN
Unground entity state
TRANSITIONS TO
Centreless G entity state
AND
Unground information state
TRANSITIONS TO
Centreless G information state

Table C.5 Centreless Grinding Elaboration

C.4 Face Grinding

The next operation which is operation 20, which is called “Face Grinding”. When the disc has been removed from the die in the synthesis plant, an operator marks the PCD (Poly Crystalline Diamond) side of it, to make it easier to distinguish at this operation. The disc is held in a three-jaw chuck with the PCD side facing out. The metal is then removed from the face of the disc, exposing the PCD layer. There are two machines, (SAGC8 & SAGC 9) which are used for this operation. These two machines are located adjacent to the centreless grinding machine used in operation 1. The operator working at operation 1 places each batch of material on the material awaiting processing table for operator 2. Each disc has to be processed individually on these machines. These machines are supported by a single operator, and are run over three shifts. The expected output from these machines is 40 per machine per shift. After each disc has been removed the operator places them to one side and when the batch is done, if on a visual inspection any require further finishing the operator places them back on the machine to complete them. The SAD for this operation is shown in Figure C 4 with the associated elaboration shown in Table C 6.

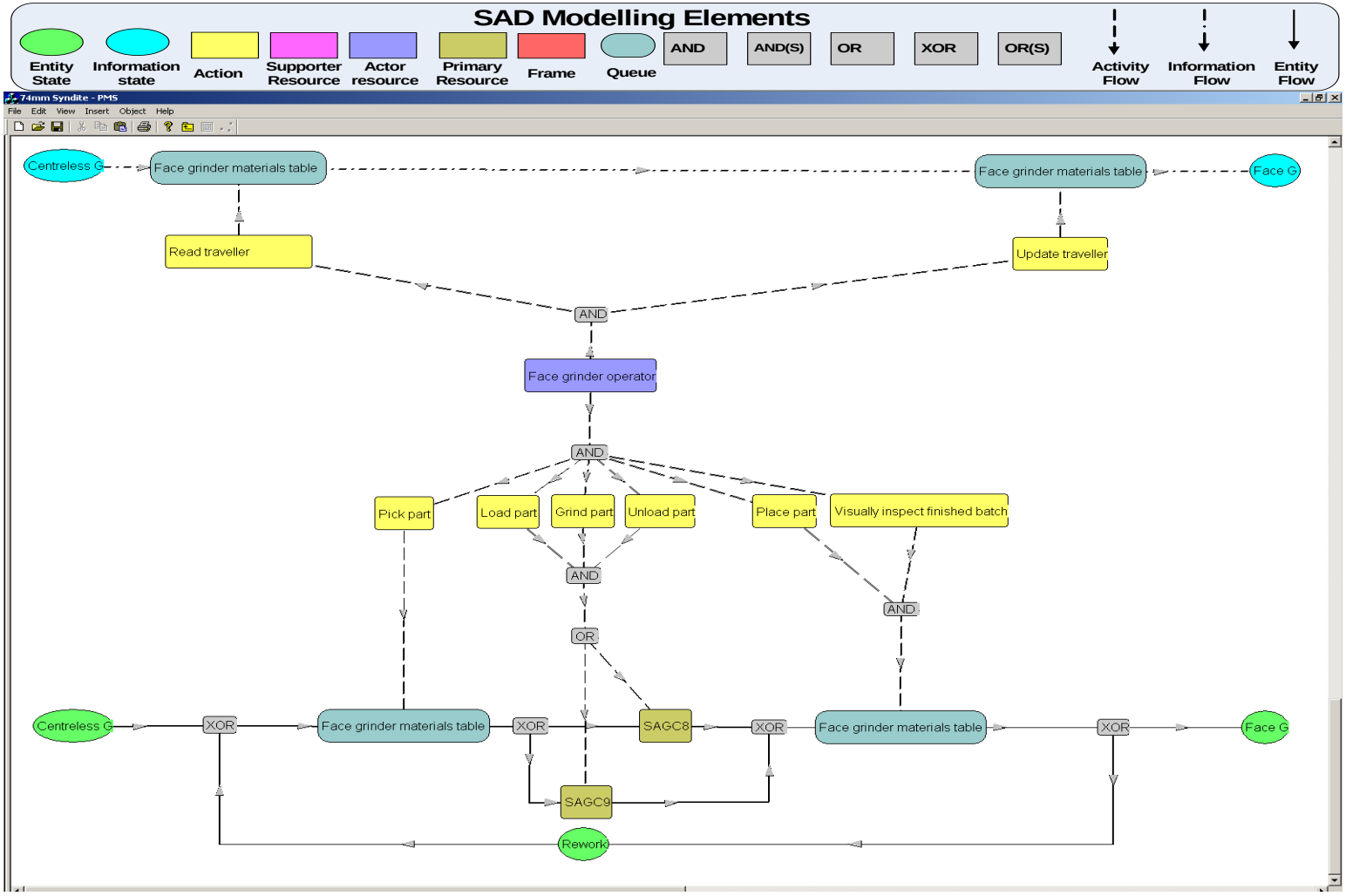

Figure C.4 Face Grinding

Elaboration of the Activity.
Face grinder operator
Pick part
AT
Face grinder materials table
AND
Load part
AND
Grind part
AND
Unload part
ON
SAGC8
OR
SAGC9
This machine has a standard cycle time of 12 minutes per disc.
AND
Place part
AND
Visually inspect finished batch
AT
Face grinder materials table
AND
Face grinder operator
Read traveller
AND
Update traveller
AT
Face grinder materials table
THEN
Centreless G entity state
TRANSITIONS TO
EITHER
Rework entity state
OR
Face G entity state
AND
Centreless G information state
TRANSITIONS TO
Face G information state

Table C.6 Face Grinding Elaboration

C.5 Surface grinding

The next step in the process is Surface Grinding.The surface grinder grinds the carbide face of the disc to bring the disc to approximately 0.4mm above the height the material is going to end up as, when it has been completely processed by the 74mm Syndite line. There are two surface grinding machines (SSG13 & SSG14), which are used for the 74mm Syndite line. These two machines are located in a different area than the previous sets of machines. Each machine holds 10 discs per run. The discs are placed flat on 74mm washers, which are placed inside a frame on the table and spacers are placed between the discs to keep them apart. The spacers and the frame are thicker than the disc washers, in order to prevent lateral movement of the discs. Once the ten discs have been placed on the machine, the table is magnetised, holding the washers and the spacers in place. It is these in turn, which hold the discs in place. These machines are supported by a single operator, and are run over three shifts. The expected output from these machines is 50 per machine per shift (i.e. 5 runs of each machine per shift). When each machine has completed a run the operator removes one piece from the table and measures its thickness around the circumference. If it is within tolerance, the rest of the discs are removed from the table and also measured. When all of these have been checked the next batch is measured, before being loaded on the machine, to determine the depth of material, which has to be removed. The parts are then loaded onto the machine and the cycle is started again, and the operator waits for the second machine to complete its cycle. Table C.7 shows the elaboration for this area while Figure C.5 details the SAD diagram.

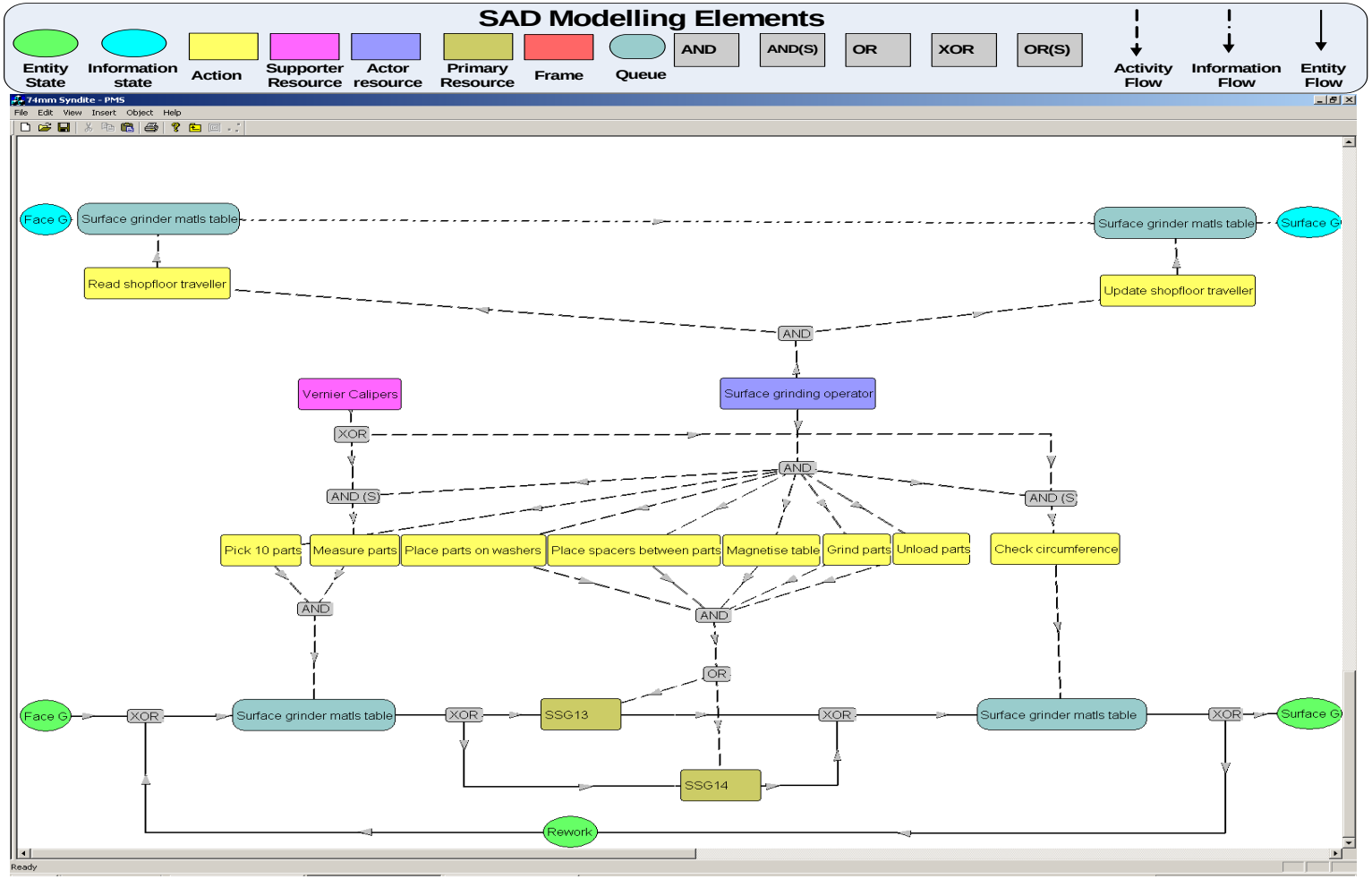

Figure C.5 Surface Grinding

Elaboration of the Activity.
Surface grinding operator
Pick 10 parts
AND
USES
Vernier Calipers
TO
Measure parts
AT
Surface grinder matls table
AND
Place parts on washers
AND
Place spacers between parts
AND
Magnetise table
AND
Grind parts
AND
Unload parts
AT
SSG13
OR
SSG14
Each machine has a standard cycle time of 96 minutes.
AND
USES
Vernier Calipers
TO
Check circumference
AT
Surface grinder matls table
AND
Surface grinding operator
Read shopfloor traveller
AND
Update shopfloor traveller
AT
Surface grinder matls table
THEN
Face G entity state
TRANSITIONS TO
EITHER
Rework entity state
OR
Surface G entity state
AND
Face G information state
TRANSITIONS TO
Surface G information state

Table C.7 Surface Grinding Elaboration

C.6 EDM Planing

The next operation in the routing is EDM planing. Material is brought across to this area once a day by the EDM shop supervisor. When the supervisor is

collecting material for the day he also brings the material completed the previous day to the next operation. The area consists of 15 EDM machines (SD3 – SD17), but only 11 of these are commissioned. There are 13 Charmiles machines and two Ingersol machines. Each machine has a fixture (palette), which holds 9 discs at a time, but the machines only operate on one disc at a time. The cycle time to process one disc in the palette is 2 hours, so it will take the machine 18 hours to complete the palette. Each machine is also equipped with a robotic arm, which can load and unload palettes for machining, and currently each machine can hold up to 12 palettes, but no more than four are used. The machines operate 3 shifts per day and have two people manning the machines on the day and evening shifts. On the night shift the machines are left to run unattended. The day shift is manned by one full time person manning the machines, and a supervisor who mans the machines and also looks after maintenance of the fixtures etc. On the evening shift there are two full time operators manning the machines and preparing the machines to operate unattended throughout the night shift. The elaboration and SAD diagram for this area are shown in Table C.8 and Figure C.6 respectively.

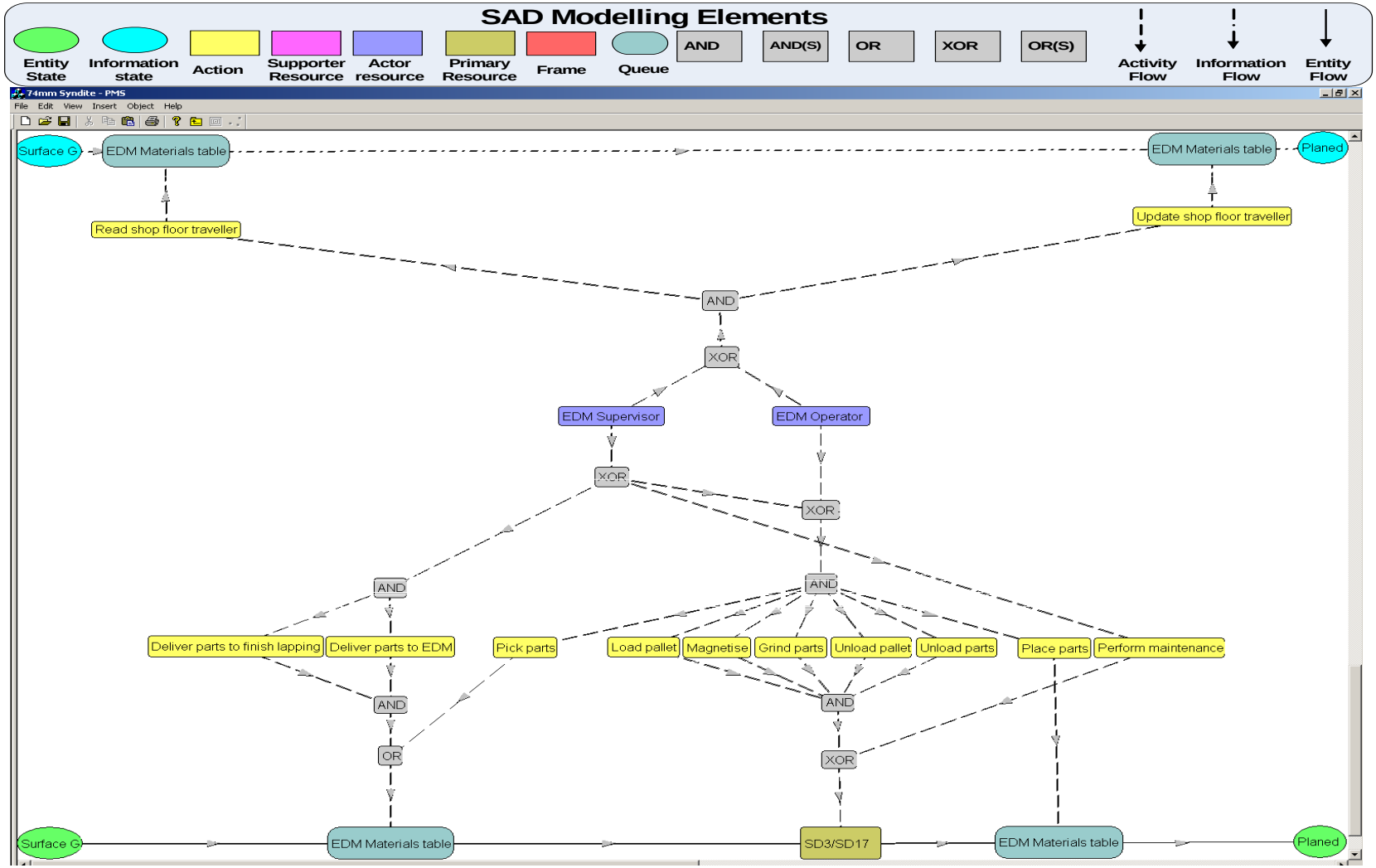

Figure C.6. EDM Planing

Elaboration of the Activity.
EDM Supervisor
EITHER
Deliver parts to finish lapping
AND
Deliver parts to EDM
OR
EITHER
EDM Supervisor
OR
EDM Operator
Pick parts
AT
EDM Materials table
AND
Load pallet
AND
Magnetise
AND
Grind Parts
AND
Unload pallet
AND
Unload parts
AT
SD3/SD17
These machines each have a standard cycle time of 18 hours.
AND
Place parts
AT
EDM Materials table
OR
Perform maintenance
AT
SD3/SD17
AND
EITHER
EDM Supervisor
OR
EDM Operator
Read shop floor traveller
AND
Update shop floor traveller
AT
EDM Materials table
THEN
Surface G entity state
TRANSITIONS TO
Planed entity state
AND
Surface G information state
TRANSITIONS TO
Planed information state

Table C.8 EDM Planing elaboration

C.7 Finish Lapping

The next operation is the Finish Lapping process. The material is brought to this area by the EDM supervisor once a day. The area consists of 4 lapping machines, and a number of pieces of testing equipment. There are two 32" machines, which are used for preparing and finishing the discs, and two 48" machines, which carry out the majority of the work. There are two cells in this area, each of which contain a 32" machine and a 48" machine. The two operators work as a team between the two cells. The area works three shifts per day and the target is to get 40 discs completed in each cell per shift. This area works to the final specifications of the relevant disc. The first operation carried out on the discs in this area is to visually examine them and determine which discs need preparation. These discs are run on the 32" machine, which has a capacity of 28 discs. When they are ready they are then removed from this machine and loaded onto the 48" machines for processing. The 48" machine has a capacity of 40 discs per run, and it is from this machine the overall batch size of 40 is determined. The process time for this machine is 4.5 hours. The discs are then loaded back on the 32" machine where a finishing operation is carried out to level the carbide. The process time for this operation is determined by the actual discs and the condition of them. When they are finished on this machine they are brought to the inspection table in the same area. Here they are initially checked under the microscope for metal, scratches and shadows (some discs may need to be reworked after this stage). They are then measured by a scanner, followed by a flatness check on a three point level gauge. After the batch has been checked it is moved to the sand blasting machine by one of the lapping area operators. Table C.9 shows the elaboration for this area while the SAD diagram for this area is shown in Figure C.7.

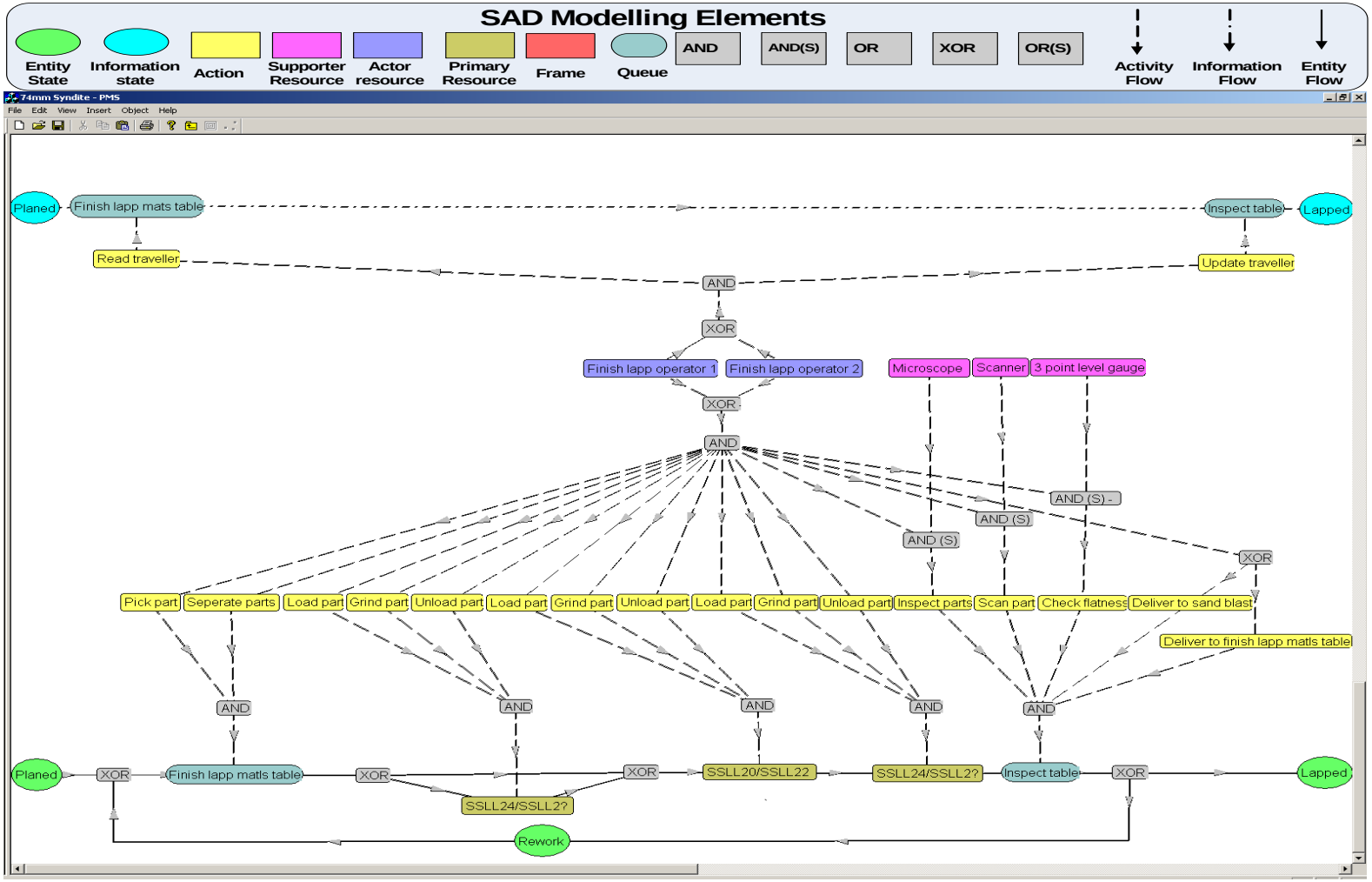

Figure C.7 Finish Lapping

Elaboration of the Activity.
EITHER
Finish lapp operator 1
OR
Finish lapp operator 2
Pick part
AND
Separate parts
AT
Finish lapp matls table
AND
Load part
AND
Grind part
AND
Unload part
AT
SSLL24/ SSLL23
This is an operation that is only performed on parts that required a preparatory operation. The cycle time for this machine is variable with the amount of preparation needed.
AND
Load part
AND
Grind part
AND
Unload part
AT
SSLL20/SSLL22
This machine has a standard cycle time of 4.5 hours.
AND
Load part
AND
Grind part
AND
Unload part
AT
SSLL24/ SSLL23
The cycle time for this machine is variable with the amount of preparation needed.
AND
USES
Microscope
TO
Inspect parts
AND
USES
Scanner
TO
Scan part
AND
USES
3 point level gauge
TO
Check flatness

AND
EITHER
Deliver to sand blast
OR
Deliver to finish lapp matls table
AT
Inspect table
AND
EITHER
Finish lapp operator 1
OR
Finish lapp operator 2
Read traveler
AT
Finish lapp matls table
AND
Update traveler
AT
Inspect table
THEN
Planed entity state
TRANSITIONS TO
EITHER
Lapped entity state
OR
Rework entity state
AND
Planed information state
TRANSITIONS TO
Lapped information state

Table C.9 Finish Lapping elaboration

C.8 Sandblasting

The next operation in the routing is sandblasting. There is only one machine for this operation and the machine is shared with the 57mm Syndite line. This operation is carried out individually on discs and is a blast of sand onto the disc to remove smears, dirt etc. to aid final visual inspection. The machine has designated operators from other operations who carry out this operation. It takes approximately 0.5 hours to complete 90 discs on this machine. Table C.10 and Figure C.8 show the elaboration and SAD diagram for this area.

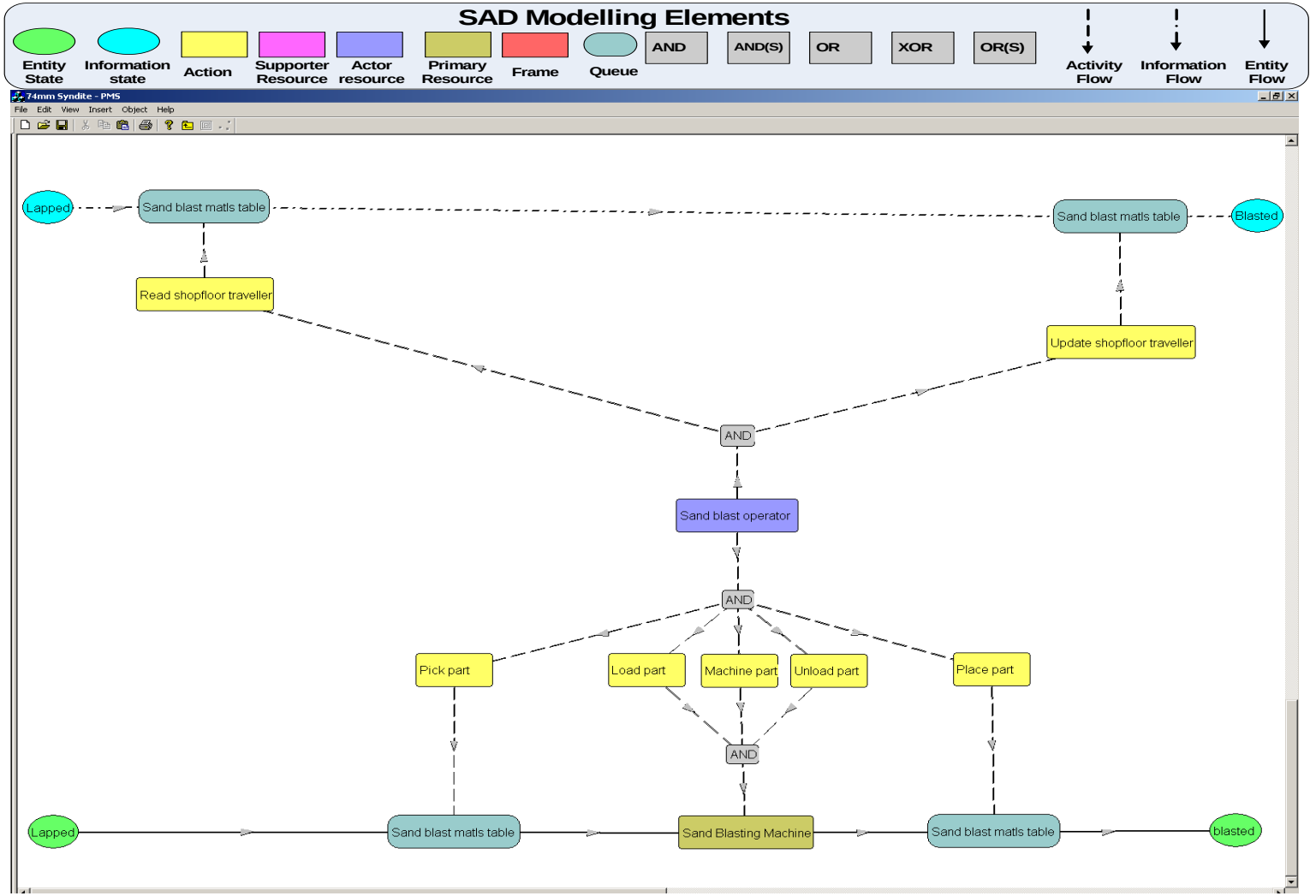

Figure C.8 Sandblasting

Elaboration of the Activity
Sand blast operator
There is no dedicated sand blast operator. The sand blast operator consists of other dedicated operators with idle time or with broken machines from other sections.
Pick part
AT
Sand blast matls table
AND
Load part
AND
Machine part
AND
Unload part
AT
Sand Blasting Machine
This machine has a standard cycle time of .333 mins per disc.
AND
Place part
AT
Sand blast matls table
AND
Sand blast operator
Read shopfloor traveler
AND
Update shopfloor traveler
AT
Sand blast matls table
THEN
Lapped entity state
TRANSITIONS TO
blasted entity state
AND
Lapped information state
TRANSITIONS TO
Blasted information state

Table C.10 Sandblasting elaboration

C.9 Assessment

The next operation in the sequence is assessment, where the discs are checked against their final specifications. Material is visually inspected under a microscope and ultrasonically tested to inspect the PCD layer for internal cracking. It is from this that the material is categorised. The actual outside diameter of the discs on the 74mm Syndite line are 76.3mm.

Material from all of the Bulk Processing lines arrives into this area for assessment. The material is placed in a physical queue. The first operation carried out on the material is a visual inspection, which has four operators. The material is visually inspected under microscopes, to check that the physical dimensions are according to the spec laid out for the relevant product. When the material has been visually inspected, an assessment sheet of the material is filled in which goes with the material to the ultrasonic testing area, which contains the category of each disc in the batch. The material then moves to one of the ultrasonic testing machines where they are tested more stringently. The machine can hold a batch size of 36 discs of diameter 74mm and 64 discs of diameter 57mm. It is usual to get between 7 and 8 scans on each of these machines per shift, and this area only operates for one shift per day. Different batches of material are not mixed on this machine. The ultrasonic testing machines are linked to a statistical package, which determines the quality of the discs. At this point the assessment sheet, which accompanied the material to the ultrasonic machine, can be altered if necessary (i.e. change the categories of certain discs). Table C.11 shows the elaboration for this area while Figure C.9 shows the SAD diagram for the same area.

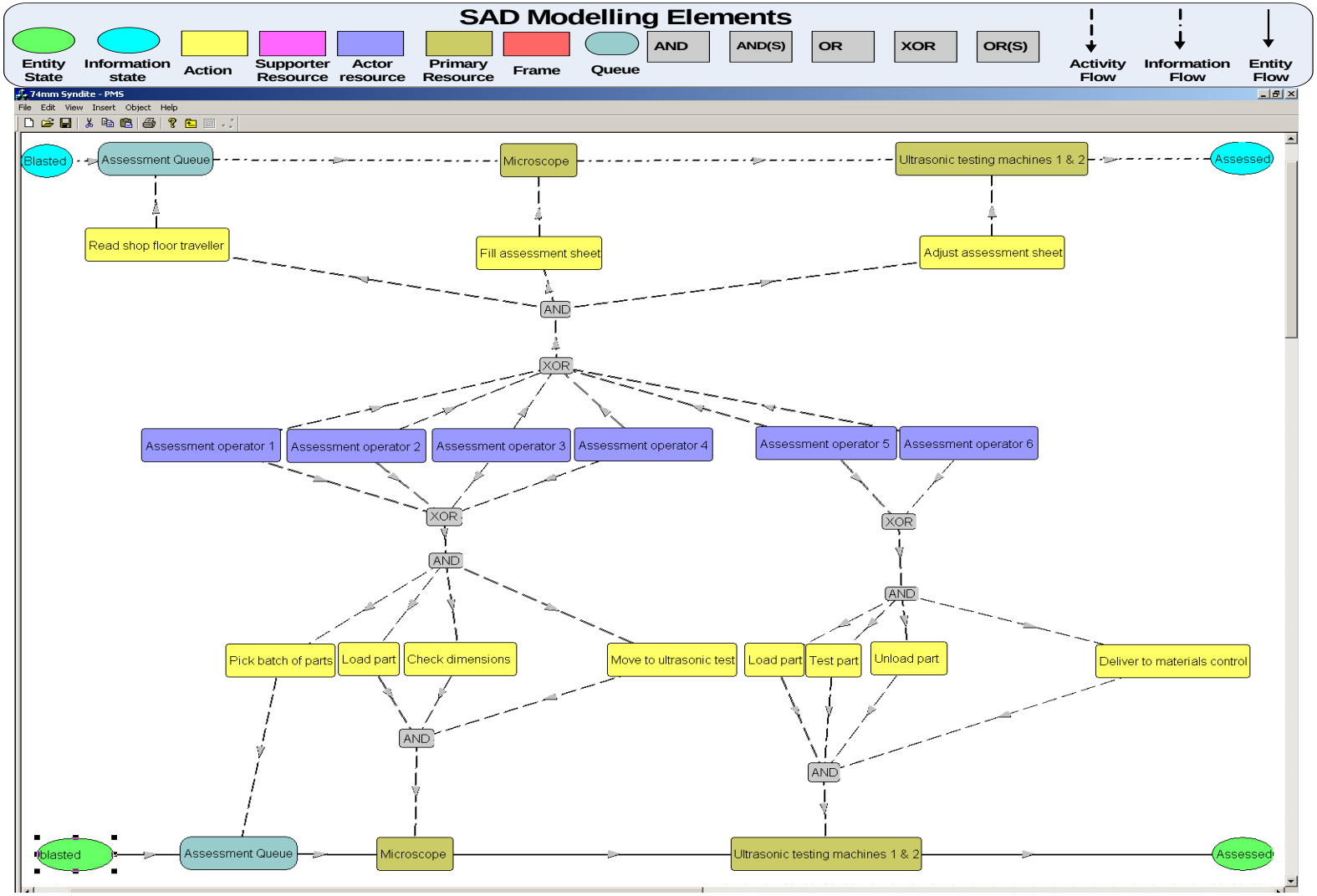

Figure C.9 Assessment

Elaboration of the Activity
EITHER
Assessment operator 1
OR
Assessment operator 2
OR
Assessment operator 3
OR
Assessment operator 4
Pick batch of parts
AT
Assessment Queue
AND
Load part
AND
Check dimensions
AND
Move to ultrasonic test
AT
Microscope
AND
EITHER
Assessment operator 5
OR
Assessment operator 6
Load part
AND
Test part
AND
Unload part
AND
Deliver to materials control
AT
Ultrasonic testing machines 1 & 2
Each machine has a standard cycle time of between 60 and 69 minutes.
AND
EITHER
Assessment operator 1
OR
Assessment operator 2
OR
Assessment operator 3
OR
Assessment operator 4
OR
Assessment operator 5
OR
Assessment operator 6
Read shop floor traveler
AT
Assessment Queue
AND
Fill assessment sheet
AT
Microscope
AND
Adjust assessment sheet

AT
Ultrasonic testing machines 1 & 2
THEN
blasted entity state
TRANSITIONS TO
Assessed entity state
AND
Blasted information state
TRANSITIONS TO
Assessed information state

Table C.11 Assessment elaboration

C.10 Materials control

The batch of material is then brought to material controls, by an operator from the assessment area, with its appropriate assessment sheet. Materials controls then enters the data on the assessment sheet into the ERP system for each disc.

The discs are categorised into “perfect” discs, “cutting” discs or “other” discs, in the assessment. Perfect discs are classified as category 11a and 11. The “a” signifies that the disc may be polished i.e. it has a height above a certain level (i.e. it is at the top of its tolerance limit for height), which means that after polishing (i.e. more material removal) it will still fall within the tolerance limits for that part. Category 11 means that the part has fallen on the low end of the tolerance limit thus it cannot be polished. A category 11a and 11 also guarantees a usable overall diameter of 74mm on the disc.

Cutting discs are categorised as per Table C.12. For example there is a category 10a disc and a category 10 disc, which guarantees a usable diameter of 70mm on the 74mm disc and where the “a” category disc can be polished. A disc that has been categorised as a 5 or lower is referred to as an “other” disc, which means it is a reject and will not be used in production. The elaboration for this area is shown in Table C.13. While the SAD diagram for the area is shown in Figure C.10.

Category	10(a)	9(a)	8(a)	7(a)	6(a)	5(a)	5b or lower
Guaranteed usable dia.	70	65	60	55	53	50	Other
“a” is a disc that can be polished.							

Table C.12 Usable disc area for cutting discs.

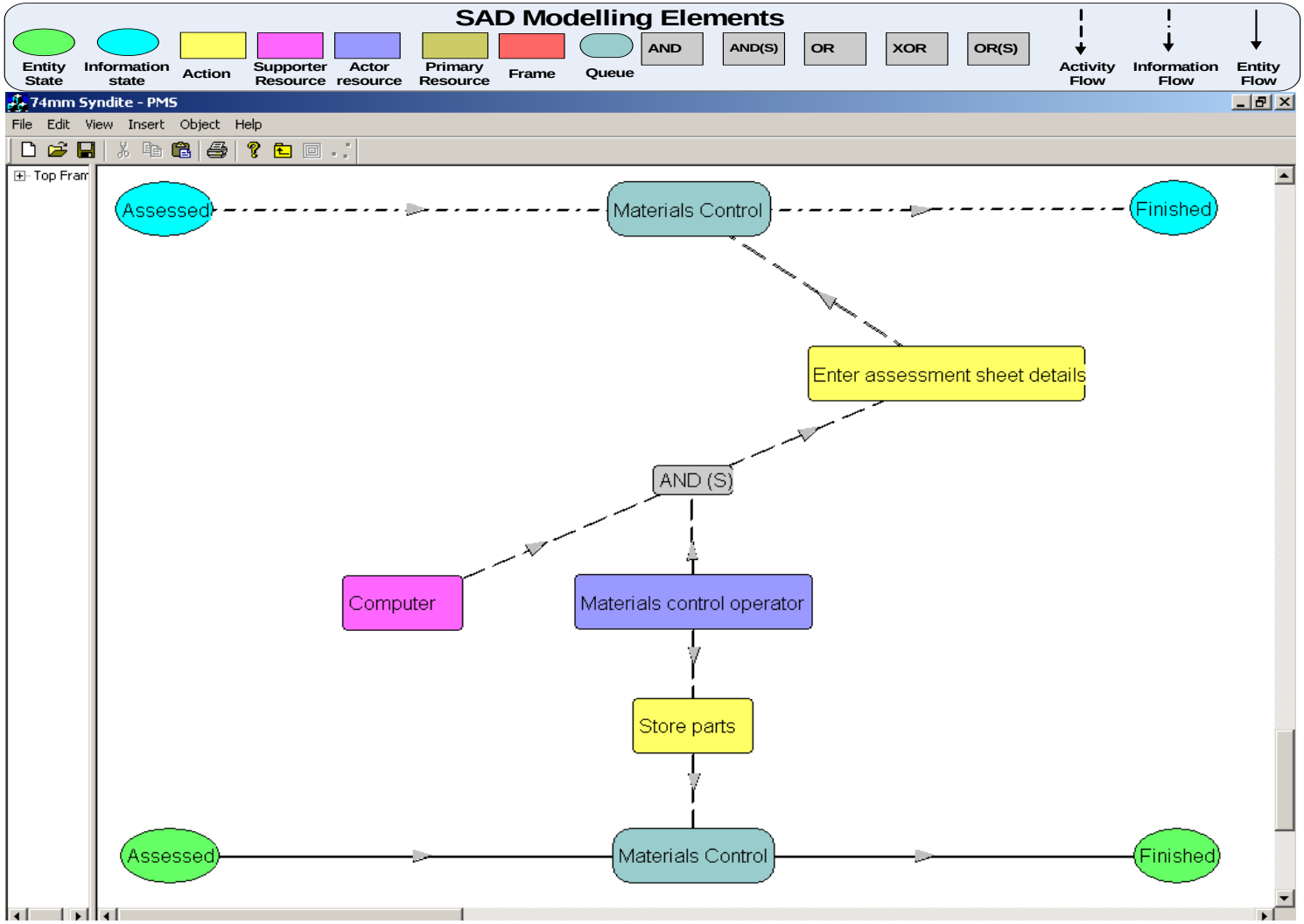

Figure C.10 Materials 2

Elaboration of the Activity.
Materials control operator
Store parts
AT
Materials Control
AND
Materials control operator
USES
Computer
TO
Enter assessment sheet details
The discs are categorised into "perfect" discs, "cutting" discs or "other" discs, in the assessment. Perfect discs are classified as category 11a and 11. The "a" signifies that the disc may be polished i.e. it has a height above a certain level (i.e. it is at the top of its tolerance limit for height), which means that after polishing (i.e. more material removal) it will still fall within the tolerance limits for that part. Category 11 means that the part has fallen on the low end of the tolerance limit thus it cannot be polished. A category 11a and 11 also guarantees a usable overall diameter of 74mm on the disc. The categorisation of cutting discs is shown in the table contained in cutting disc diameters.doc.
AT
Materials Control
THEN
Assessed entity state
TRANSITIONS TO
Finished entity state
AND
Assessed information state
TRANSITIONS TO
Finished information state

Table C.13 Materials 2 elaboration