A USER-ORIENTED ENTERPRISE PROCESS MODELING LANGUAGE

By

AMIT A. CHAUGULE

Bachelor of Engineering

Bombay University

Bombay, India

1995

Submitted to Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
August, 2001

A USER-ORIENTED ENTERPRISE PROCESS MODELING LANGUAGE

Thesis Approved:

Manjunath Kamath

Thesis Adviser

Alfred Carlozzi

Dean of the Graduate College

PREFACE

Enterprise process modeling has been an emerging topic of interest since the early nineties. The research in this area has been driven by the vision of process improvement. There are two key steps in applying process modeling tools and techniques to support process improvement initiatives. These are (i) the correct representation of the processes in the form of a process model, and (ii) the analysis of the processes to identify improvement opportunities.

Process modeling is representing processes and the relevant details usually in a graphical language. These details are the inputs to and the outputs from a process, the description of the resources used or consumed by a process, and the relationship of the subjects involved in the process with respect to each other. The literature contains many process modeling tools and techniques. A technique typically involves graphical symbols with their semantics and syntax to capture process details. This thesis presents a brief review of several enterprise process modeling languages that have been developed so far. The strengths and the limitations of these languages are also presented. These form the basis for the requirements of a new enterprise process modeling language.

The proposed enterprise process modeling language exploits the strengths of existing process modeling languages. The proposed language is user friendly, yet rigorous in the definition of its constructs. It emphasizes control flow, which is an essential aspect of any process model. Emphasis on control flow is useful for analyzing a process description by using formal tools such as Petri nets. A comprehensive example is represented in the existing languages and in the proposed language to illustrate the advantages of the proposed language.

ACKNOWLEDGEMENTS

Success is a sequel to hard work, and hard work is a result of motivation. It was a privilege to have been motivated to pursue an independent research and to discover an immense capacity for learning, and to enjoy it. This would have been impossible, but for the generous support and encouragement of my advisor Dr. Manjunath Kamath, who was always there to guide me professionally and personally. I extend my sincere appreciation to my committee members Drs. William J. Kolarik and Nikunj P. Dalal, whose guidance and suggestions were invaluable to my work.

The preparation of this document was a time consuming task and I thank all the committee members, particularly Dr. Kamath, who invested their time and offered their critique. I am indebted to them for nominating me for the Graduate Research Excellence Award for outstanding research contributions for a Master's thesis. Their constructive guidance enabled me to win this award. This thesis was a part of a NSF funded project at Oklahoma State University and it was worthwhile to work with the project team.

It was a pleasure to work with my colleagues in the Center for Computer Integrated Manufacturing (CCIM), especially Eswar and Baskar. The work atmosphere was a learning place with stimulating discussions on contemporary research and learning avenues, and the exciting gossips are definitely worth mention. Many thanks to my friend Srikant, and to Baskar whose timely help made me complete this document on time.

My sincere thanks to my parents and my sister for the support and encouragement that I have been receiving for my endeavors.

I appreciate the support that I received from the School of Industrial Engineering and Management at Oklahoma State University. The past two years of my graduate study have been a rewarding experience and I hope that they hold promise for a bright future.

"I dedicate this thesis in fond memory of my beloved grandmother who would have loved to see me excel. She was, and will continue to be a source of motivation for me."

TABLE OF CONTENTS

Chapter **Page**

1. Introduction 1

1.1. Terminology 1

1.2. Available Aids for Enterprise Modeling 2

1.3. The Problem Description 3

1.4. A Solution 3

1.5. The Research Area 4

1.6. Outline of the Thesis 4

2. Research Statement and Methodology 6

2.1. Research Objectives 6

2.2. Scope of the Research 7

2.3. Research Methodology 7

3. Literature Review 9

3.1. Terminology 9

3.2. Data Flow Diagram (DFD) 11

3.3. Integration Definition for Function Modeling (IDEF0) 13

3.4. IDEF3 – A Process Description Capture Method 15

3.5. SAP's Event-driven Process Chain (EPC) Method 17

3.6. CIMOSA – CIM Open System Architecture 20

3.7. Integrated Enterprise Modeling (IEM) 22

3.8. The Purdue Enterprise Reference Architecture (PERA) 24

3.9. GRAI-GIM (GRAI Integrated Model) 26

3.10. TOronto Virtual Enterprise (TOVE) Method 28

3.11. Baan's Dynamic Enterprise Modeling (DEM) Technique 30

3.12. Generalized Enterprise Reference Architecture and Methodology 31

3.13. Architecture for Integrated Information Systems 33

3.14. Unified Modeling Language 35

3.15. Summary 37

4. Characteristics of an Enterprise Process Modeling Language 38

4.1. Characteristics 38

4.2. Elements of a Process Model 41

5. Enterprise Process Modeling Language (EPML) 42

5.1. Activity 42

5.2. Control Flow 45

5.3. Data Flow 46

5.4. Electronic Data Flow 46

5.5. Material Flow 47

5.6. Connectors 47

5.7. Time Trigger 49

5.8. Logical Operators 49

5.9. Binary Decision 55

5.10. Constraint, Rules, and Guidelines 56

5.11. Rules for Representing Control Flow to an Activity 56

5.12. Feedback Representation 57

5.13. Data Flows and Logical Constructs 58

5.14. EPML Implementation within a Software Environment 58

6. Comparison of EPML with Existing Process Modeling Techniques 61

6.1. The Scenario 61

6.2. Modeling using IDEF Techniques 62

6.3. Modeling using CIMOSA 67

6.4. Modeling using IEM 67

6.5. Modeling using TOVE 71

6.6. Modeling using SAP's EPC 74

6.7. Modeling using Data Flow Diagram 74

6.8. Modeling using BAAN's DEM 78

6.9. Modeling using EPML.......... 81

6.10. Comparison Criteria.......... 81

6.11. Summary.......... 84

7. Conclusions and Future Work.......... 87

7.1. Research Summary 87

7.2. Contributions 88

7.3. Future work.......... 88

Bibliography 91

LIST OF TABLES

Table **Page**

Table 1: Summary of the Comparison Study.......... 86

TABLE OF FIGURES

Figure **Page**

Figure 3.1: DFD for a Shipping Process 12

Figure 3.2 Basic Building Block of IDEF0 14

Figure 3.3: Example of Functional Decomposition in IDEF0 14

Figure 3.4: IDEF3 Process Description Diagram 16

Figure 3.5: IDEF3 Object State Transition Diagram 16

Figure 3.6: Constructs for the EPC Method 19

Figure 3.7: Example of an Event-driven Process Chain 19

Figure 3.8: CIMOSA Business Modeling Constructs 21

Figure 3.9: CIMOSA Domains, Business Processes and Enterprise Activities 21

Figure 3.10: Basic Building Block of IEM 23

Figure 3.11: IEM General Activity Model 23

Figure 3.12: PERA – Different Task Modules 26

Figure 3.13: GRAI-GIM Reference Model 27

Figure 3.14: GRAI-GIM 2-D Matrix 27

Figure 3.15: GIM Modeling Framework 27

Figure 3.16: TOVE Activity State Model 29

Figure 3.17: DEM Modeling Constructs 30

Figure 3.18: Manufacturing Example in DEM 31

Figure 3.19: GERAM Framework Components 32

Figure 3.20: Basic Building Blocks of an ARIS Process Model 34

Figure 3.21: UML Activity Diagram Constructs 36

Figure 3.22: UML Activity Diagram Example 36

Figure 5.1: Activity block 43

Figure 5.2: Control Flow 45

Figure 5.3: External Trigger 45

Figure 5.4: Data Flow 46

Figure 5.5: Electronic Data Flow 47

Figure 5.6: Material Flow 47

Figure 5.7: Use of Connectors 48

Figure 5.8: Connector 48

Figure 5.9: Time Trigger 49

Figure 5.10: AND Operator – Parallel Split 50

Figure 5.11: AND Operator – Parallel Merge 51

Figure 5.12: XOR Operator – (One to one-out-of-many) 52

Figure 5.13: XOR Operator – (One-out-of-many to One) 53

Figure 5.14: IOR Operator – One to Many 53

Figure 5.15: IOR Operator – Many to One 54

Figure 5.16: An Example of a Binary Decision 55

Figure 5.17: Constraint Construct 56

Figure 5.18: Rules for Representing Control Flow 57

Figure 5.19: Feedback Representation 58

Figure 5.20: Software Implementation Details of EPML Constructs 60

Figure 6.1: IDEF0 Representation of HAL's Ordering Process 63

Figure 6.2: IDEF0 Representation of HAL's Ordering Process, - Sub process # 2.4 64

Figure 6.3: IDEF3 Representation of HAL's Ordering Process 65

Figure 6.4: IDEF3 Representation of HAL's Ordering Process - Sub process # 2.4 66

Figure 6.5: CIMOSA Representation of HAL's Ordering Process 68

Figure 6.6: IEM Function Chain for Hal's Ordering Process 69

Figure 6.7: IEM Partially Autonomous Unit 70

Figure 6.8: TOVE – Activity Abstraction Diagram – HAL's Ordering Process 72

Figure 6.9: TOVE – Activity Cluster Diagram 73

Figure 6.10: SAP's EPC Representation of HAL's Ordering Process 75

Figure 6.11: SAP's EPC Representation of HAL's Ordering Process, (contd.) 76

Figure 6.12: DFD Representation of HAL's Ordering Process 77

Figure 6.13: Baan's DEM Representation of HAL's Ordering Process 79

Figure 6.14: Baan's DEM Representation – Sub Process-HAL's Loan Payment System 80

Figure 6.15: EPML Representation of HAL's Ordering system 82

ABBREVIATIONS

ARIS	Architecture for Integrated Information Systems
CIM	Computer Integrated Manufacturing
CIMOSA	CIM Open System Architecture
DEM	Dynamic Enterprise Modeling
DFD	Data Flow Diagram
EPC	Event-driven Process Chain
EPML	Enterprise Process Modeling Language
GERAM	Generalized Enterprise Reference Architecture
GRAI-GIM	GRAI Integrated Methodology
IDEF	Integration Definition for Function Modeling
IEM	Integrated Enterprise Modeling
PERA	Purdue Enterprise Reference Architecture
TOVE	Toronto Virtual Enterprise
UML	Unified Modeling Language

1. Introduction

An enterprise is a group of organizations that have a common objective to offer products and services. Enterprise modeling is the process of representing an enterprise, by explicitly describing its operations. Enterprise models help us in understanding proposed or existing business processes, in our quest to improve the effectiveness and efficiency of the enterprise as a whole. The representation of an enterprise could be in the form of a mathematical model, a symbolic representation, or a textual description. The purpose of a model is to allow the modeler to gain a thorough understanding of the enterprise, analyze its processes, and suggest changes to improve the processes.

An analogy to the product development process is immediate. While developing a new product, or modifying the design of a product, the designer needs to know the exact function of the product. Then the designer formulates a set of requirements for the product and designs the product using scientific principles. The output is usually in the form of a drawing, which is then translated to a physical product. In this process, well-established techniques, tools and performance measures support the design process. For example, the designer can categorize product requirements into mechanical design requirements, expected fatigue life, thermal requirements, etc. These requirements are then translated into design specifications using design tools such as IDEAS (www.sdrc.com), which are readily available in the form of commercial software packages. Specific performance measures can help verify the design after product testing. In the present context, the "product" is usually a process or a set of processes to achieve certain objectives.

1.1. Terminology

Before proceeding further, it is necessary to clarify the meaning of some terms that are frequently mentioned in the enterprise modeling literature (ISO/FDIS 15704, 1999). Additional details are contained in Section 3.1.

Activity: A unit of functionality; it might be all or a part of functionality.

Business processes: A set of activities in an order designed to achieve the goals of an enterprise.

Framework: A structural diagram that relates component parts of an entity to each other.

Architecture: A description of the basic arrangement and connectivity of the different parts of a system.

Methodology: A set of instructions, which serve as a step-by-step guide to the user.

Modeling Language: A set of constructs, and their syntax and semantics by which a system can be modeled.

1.2. Available Aids for Enterprise Modeling

Enterprise modeling has been an active area of research for the last several years. The efforts are evident in the various forms of modeling techniques, methodologies and frameworks that exist. Examples are CIMOSA, PERA, IEM, GERAM, and GRAI-GIM. These efforts have produced reference architectures to support the organizing of enterprise knowledge, and to serve as a guide in enterprise integration programs. Each of these compilations has been developed with a specific purpose in mind. CIMOSA provides a framework to guide the users in modeling business requirements, aid engineers in enterprise design and implementation, and support vendors in system component development (Zelm *et al.* 1995; Kosanke 1995). PERA provides the capability of modeling the human component as well as the manufacturing or customer service component of any enterprise, in addition to the information and control system components (Williams 1994). PERA describes a unique method for defining the place of the human in a computer-integrated enterprise. IEM claims that its approach to model an enterprise can provide a functional base to create a unified model from a user's point of view (Mertins *et al.* 1992). GRAI-GIM supports an integrated manufacturing environment and recommends different techniques to be used for different purposes, e.g., IDEF0 for functional modeling and Entity Relationship Diagram for data modeling (Chen and Doumeingts 1996). A unique feature of GRAI-GIM is its decision control model. GERAM defines a tool-kit of concepts for designing and maintaining enterprises for their entire life cycle (GERAM 1999). Examples of modeling techniques mentioned in GERAM are IDEF0 for functional modeling, IDEF1X for data modeling and data flow diagrams for process modeling.

1.3. The Problem Description

Given the task of modeling an enterprise for analyzing and improving its performance, the user has many choices as described in the previous section. But the success of this effort depends to a great extent on the modeling language chosen, and its associated methodology and framework, if any. For example, one might choose data flow diagrams (Dao and Rodjak 1991) or the IDEF0 technique (Colquhoun *et al.* 1993; Bravovo and Yadav 1985) to start the modeling effort because the constructs are easy to understand and use. However, these languages lack a theory base and are not capable of supporting any formal performance analysis. They might be useful in capturing the enterprise processes from the top to the bottom. But the user has to deal with a bewildering array of diagrams that result, and could find that it is very difficult to analyze the processes without additional support from the modeling technique.

On the other hand, suppose an "intelligent user" chooses a theory-based technique to model an enterprise. Let us also assume that the user will be successful in describing the enterprise as well as analyzing and improving its processes. However, a common user may find the model difficult to comprehend because of the complexity inherent in a formal theory base supporting the model. So, we have an apparent conflict between an easy to use or informal modeling language and a formal, theory-based language. Each has its own advantages and disadvantages.

1.4. A Solution

An ongoing project in the School of Industrial Engineering and Management at the Oklahoma State University, funded by the National Science Foundation through grant # DMI-0075588, is focused on creating a user-oriented process and performance modeling framework (Kamath *et al.* 2001). This effort is aimed at creating a user-oriented front-end language with Petri net theory (Zurawski and Zhou 1994) as the underlying mathematical engine. The idea is to create a graphical front end that would let the user model enterprise processes graphically. The front-end description is linked to a Petri net-based back-end tool, which will then be used to analyze the processes. The front-end and the back-end will be integrated through a two-way mapping scheme such that any changes in the front end will be reflected in the back end and vice versa.

1.5. The Research Area

The creation of a front-end graphical modeling language as described in the previous section is the focus of this research. Not all of the enterprise modeling frameworks described in Section 1.2 have a modeling language of their own. For example, GRAI-GIM framework recommends IDEF0 for process modeling (Chen and Doumeingts 1996). GERAM does not recommend any specific modeling language, but suggests that it should be integrated with data modeling and other tools used to model the entire enterprise (GERAM 1999). CIMOSA has its own set of constructs, which allow a detailed and hierarchical description of the enterprise processes (Zelm *et al.* 1995). However, it is not as strong in depicting control flow as the SAP-EPC model (Keller and Detering 1993). Each language has its own strengths and limitations.

This research involves the development of a front-end graphical modeling language for the NSF-funded research effort. The focus of this thesis was limited to developing a set of constructs with clearly defined semantics and syntax, which will help a modeler to capture details needed for process modeling in a clear and unambiguous manner. Any associated framework, architecture and methodology were beyond the scope of this thesis.

1.6. Outline of the Thesis

The remainder of this thesis is presented in six chapters. Chapter 2 presents the research objectives, scope, and deliverables, and focuses on the steps that were taken during development of the proposed language. Chapter 3 includes a review of the literature on enterprise modeling languages. It also elucidates their strong and weak points. Chapter 4 describes the characteristics of a process modeling language. The constructs, semantics and syntax for the language developed are described in Chapter 5. Chapter 6 presents a comparative study and qualitative arguments on limitations of the existing process modeling techniques based on a business process example. Chapter 7 concludes this thesis, and identifies areas for future work.

2. Research Statement and Methodology

Existing enterprise process modeling languages have their own strengths and limitations. There is a need to develop a graphical process modeling language that potentially addresses the shortcomings of the existing languages while retaining their strengths. As mentioned in Section 1.4, the research work underway at the Oklahoma State University addresses the following limitations of the current enterprise modeling languages (Kamath *et al.* 2001).

- Need for a theory base
- Need for modeling and implementing distributed computing
- Need for integrating activity based management
- Need for linkages between business and engineering processes

This research was an integral part of a larger process modeling framework and partially addresses the above limitations. Petri nets will serve as the theory base for the modeling approach and will be the back-end representation. Petri net representation will be linked to a graphical process modeling tool at the front end (Sivaraman 2001). Modern concepts such as activity based management, linkages between the engineering and business approaches, and the distributed computing concept of the Internet are incorporated into the proposed framework.

2.1. Research Objectives

The purpose of this thesis was to develop enterprise process modeling constructs with clearly defined semantics and syntax that take into account the limitations of the existing process modeling languages. The objectives of the research were:

1. To conduct a thorough evaluation of existing enterprise process modeling techniques.

2. To identify constructs and theory from existing modeling techniques that can be incorporated into a new enterprise process modeling language.
3. To design a new set of graphical constructs for enterprise process modeling with clearly defined syntax and semantics.
4. To evaluate the enterprise process modeling language developed.

2.2. Scope of the Research

The purpose of this research was to develop a set of graphical constructs with well-defined semantics and syntax, which will not only blend the advantages of the existing process modeling techniques, but also allow the user to model processes as observed in reality. The central idea was to provide flexibility and also avoid syntactic constraints and semantic ambiguities while modeling a process. Henceforth, we will call the language developed as enterprise process modeling language or EPML. The architecture, which will support implementation of the proposed EPML, is a logical next stage in the research. As mentioned in Section 1.4, mapping with the back-end representation (Petri nets) was a parallel activity as a part of the NSF project. The EPML is structured in such a way that the process description can be readily transformed into a Petri net. Mapping between EPML and Petri nets was a part of the NSF project, but not a part of this thesis effort.

The NSF-funded project at Oklahoma State University also addressed the issue of scalability and prescriptive ability with the modeling framework. Efforts to address the issues of scalability and prescriptive ability were not a part of this thesis. Also, a Web-based software implementation of EPML was a part of the NSF project, but not a part of this thesis.

2.3. Research Methodology

In order to accomplish the objectives stated in Section 2.1, the effort was divided into six distinct stages as outlined below.

Stage 1: The existing modeling techniques were studied and explored in detail. The purpose, strengths and limitations of these techniques were evaluated in the light of the

NSF project and its objectives. Constructs common to most of the process modeling languages were identified.

Stage 2: A list of all the elements that could be captured in a process model was developed (Section 4.2). This list was useful as a checklist for verifying that details necessary to describe a process were being captured by the EPML constructs.

Stage 3: In this stage, concepts like distributed computing and linkages between business and technical processes were studied in the context of defining constructs or features of constructs in the proposed modeling language.

Stage 4: In this stage, the information gathered in the earlier stages was used to develop a new set of constructs, and their syntax and semantics. Apart from designing a new set of constructs for correctly and completely describing the processes in an enterprise, a secondary objective was to facilitate Petri net representations.

Stage 5: The nature of this thesis is qualitative. Hence, verification and validation by running computer simulations or experimentation was not possible in order to judge the effectiveness of EPML. An ideal testing environment would involve modeling by a chosen group of knowledgeable users in the industrial world. However, this task would require substantial time and effort, and was clearly beyond the scope of this thesis. Hence, for the fulfillment of the objectives of the thesis, an example business process was modeled in EPML and in the existing process modeling languages. The comparison and evaluation was done with qualitative arguments, and subsequently summarized in a tabular form.

Stage 6: The final stage involved the identification of potential enhancements and extensions to the EPML. Future work related to the use of EPML within a modeling framework under development was also documented.

3. Literature Review

In this chapter, the terminology that is frequently used in the field of enterprise modeling is described in some detail. This is followed by a description of various enterprise process modeling languages and approaches. The strengths and limitations of each modeling language are also elucidated.

3.1. Terminology

Before exploring the techniques and various enterprise modeling languages, one should be well acquainted with the terms that are frequently used. The terms presented herein are a collection from standards ISO 15704 (1999); ISO 14258 (1999); GERAM (1999); and WFMC-TC-1011 (1994).

Enterprise: It is a group of organizations sharing a set of goals and objectives to offer products and services (ISO 14258 1999).

Process: It is a formalized view of business operations, represented as a co-ordinated (parallel and/or serial) set of activities that are connected in order to achieve a common goal. It can also be considered as a network of activities (WFMC-TC-1011 1994).

Sub-Process: It is a process that is enacted or called from another (initiating) process (or sub-process), and which forms part of the overall (initiating) process. There can be multiple levels of sub-process (WFMC-TC-1011 1994).

Activity: It can be considered as a unit of a sub-process or of a process. An activity could be further broken down into tasks and sub-tasks. In this thesis, an activity means that a unit of a process that uses some input to produce an output(s). A process also has inputs and outputs, but at a higher level than an activity. Decomposition of an activity into tasks and sub-tasks is not done in this thesis.

Event: It is a point in time, which indicates that an activity or a process has taken place. It acts as a trigger to one or more activities/processes when it occurs (Keller and Detering 1996).

Enterprise Modeling: It is a process of representing what an enterprise intends to accomplish and how it operates (GERAM 1999). The results of enterprise modeling are the various designs, models prepared for analysis, executable models to support operation and so on. Process modeling is only a part of enterprise modeling. All other customary design and analysis activities that create descriptions or models of the enterprise in any phase of the life cycle (such as engineering drawings, charts, etc.) fall under enterprise modeling. The current emphasis on process modeling is because of the fact that it has not received much attention in earlier efforts (GERAM 1999)

Process Modeling: It is the activity that results in various models of management/ control, and of service/production processes and their relationships to the resources, organization, products, etc., of the enterprise. Process modeling allows the user to represent the operation of enterprise entities in all their aspects: functional, behavior, information, resources and organization (GERAM 1999)

Framework: A framework is a structural diagram that relates the component parts of a conceptual entity to each other (ISO 15704 1999). The interpretation of this definition is that a framework is an abstract idea at a higher level than architecture and methodology. It is an abstract representation of all the components of an enterprise model, and guides the user or the modeler as to what goes where.

Architecture: An architecture is a description of the basic arrangement and connectivity of the different parts of a system, including both physical and conceptual objects/entities (ISO 15704 1999).

Methodology: A methodology is a set of instructions provided through text, computer programs, tools, etc., that is a step-by-step aid to the user (ISO 15704 1999). A methodology guides the user in the modeling process.

Tool: A tool is an aid with which the user can model a system or a process. Thus a data flow diagram (Dao and Rodjak 1991) or an IDEF0 diagram (Colquhoun *et al.* 1993;

Bravovo and Yadav 1985) is a tool to model a process. An entity relationship diagram (Song and Froehlich 1994) is a tool to model data.

View: Due to the complexity and the size of an enterprise model, a model can be presented to the user in different subsets of an integrated model. This concept is explained in GERAM (1999). "Views contain a subset of facts present in the integrated model allowing the user to concentrate on the relevant questions that the respective stakeholders may wish to consider during enterprise modeling." GERAM has identified the following views:

- *Entity Model Contents Views*: function, information, resource, and organization.
- *Entity Purpose Views*: customer service and product, management and control.
- *Entity Implementation View*: human implemented tasks and automated tasks.
- *Entity Physical Manifestation Views*: software and hardware.

Additional views could also be defined as per the user requirements.

3.2. Data Flow Diagram (DFD)

A data flow diagram (DFD) is a technique to model the processes in a system using a simple set of graphical symbols (Dao and Rodjak 1991, Whitten and Bentley 1998). In a DFD, a square box represents an external source or destination (sink) of data. A rounded rectangle represents a process. A data store is represented by an open-ended rectangle (open on its width on one side or on both sides in another notation). A context level diagram shows the main components of the system under consideration. This diagram is then decomposed until the level of detail required is reached. The general principle in data flow diagramming is that a system can be decomposed into subsystems, which can further be decomposed into lower level subsystems, and so on. Each subsystem represents a process or activity in which data is processed. At the lowest level, processes can no longer be decomposed. Each process in a DFD has the characteristics of a system. Just as a system has input and output, a process has input and output. Data enters the system from the environment; data flows between processes within the system; and data is produced as output from the system. Directed arcs represent the flow of data from one component of the model to the others. Arcs are labeled to indicate what flows across the system components. An example of an item shipping process is shown in Figure 3.1, which is taken from (Burch 1992).

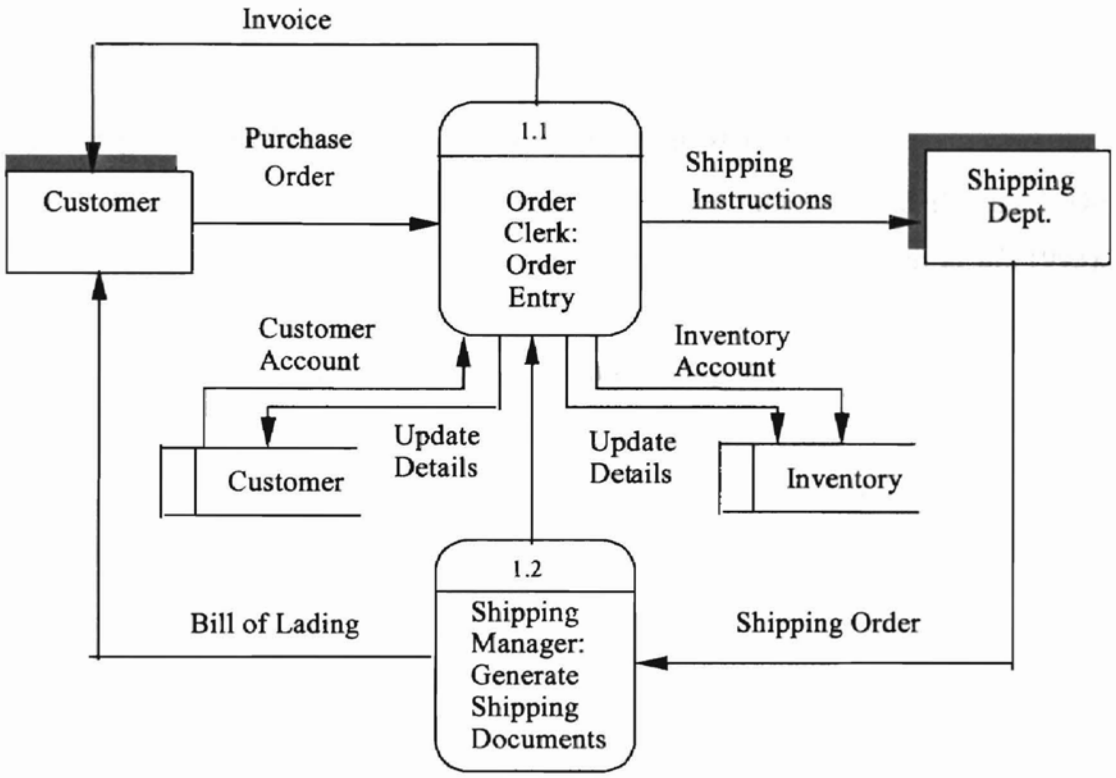


Figure 3.1: DFD for a Shipping Process

Strengths: DFD is probably the simplest process modeling technique. A person who is well acquainted with a given process/system can quickly begin to model the system with the DFD technique. It is recommended that a process be broken down functionally as far as possible. This will help an organization clearly understand the requirements of its different functions as they are expressed graphically. One key benefit of DFDs is that they promote awareness of information sharing among different functions within an organization (Burch 1992).

Limitations: A common misconception with a DFD is that the graphical language implies a process sequence like a flow chart, which is not the case. The graphical language lacks a formal theory base, and hence, analysis based on a DFD is very subjective and constrained by the user's/modeler's understanding of the system. Time and control flows cannot be represented in a DFD, and as a result, it is not possible to capture the process dynamics. The many data stores, in today's world, can be represented by a single information system or data warehouse. Hence, there is perhaps no need to represent different data stores in the DFD and complicate the diagram. This idea is also expressed in (Millet 1999).

3.3. Integration Definition for Function Modeling (IDEF0)

IDEF0 is a language which evolved from Structured Analysis and Design Technique (SADT) developed by Douglass Ross and SofTech Inc. (Colquhoun *et al.* 1993). The United States Air Force commissioned the developers of SADT to develop a functional modeling method for analyzing and communicating the functional perspective of a system. This lead to the development of IDEF0.

IDEF0 is a graphical modeling language which can be used to define the requirements of a system, specify its functions, and design implementations. The basic components of an IDEF0 model are

- an activity box; and
- data or object interfaces represented by arrows.

A box provides information about an activity that takes place in the system. To represent a complete process, these boxes can be linked with each other. The basic unit of an IDEF0 model is as shown in Figure 3.2. The arrow connecting the box from the left represents the input to the activity and the arrow coming out from the box from the right represents the output from the activity. The arrow connecting to the box from the top represents the control or the constraint for the activity. The arrow connecting to the box from the bottom represents the resources or the mechanisms by which the activity is performed.

The result of applying DEF0 to a system is a model that consists of a hierarchical series of diagrams (e.g., see Figure 3.3), text and glossary cross referenced to each other (Colquhoun *et al.* 1993; Bravovo and Yadav 1985). The exposition of details is gradual and controlled as IDEF0 allows only a maximum of six lower-level activity blocks when expanding a higher-level activity.

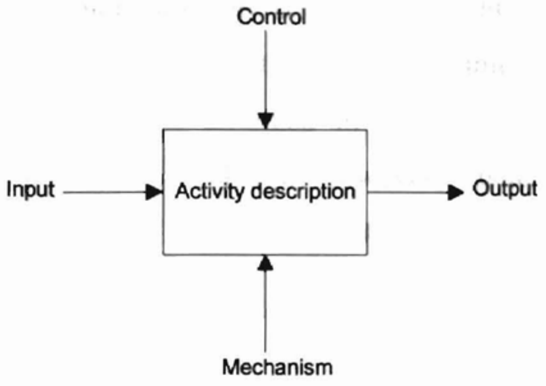


Figure 3.2 Basic Building Block of IDEF0

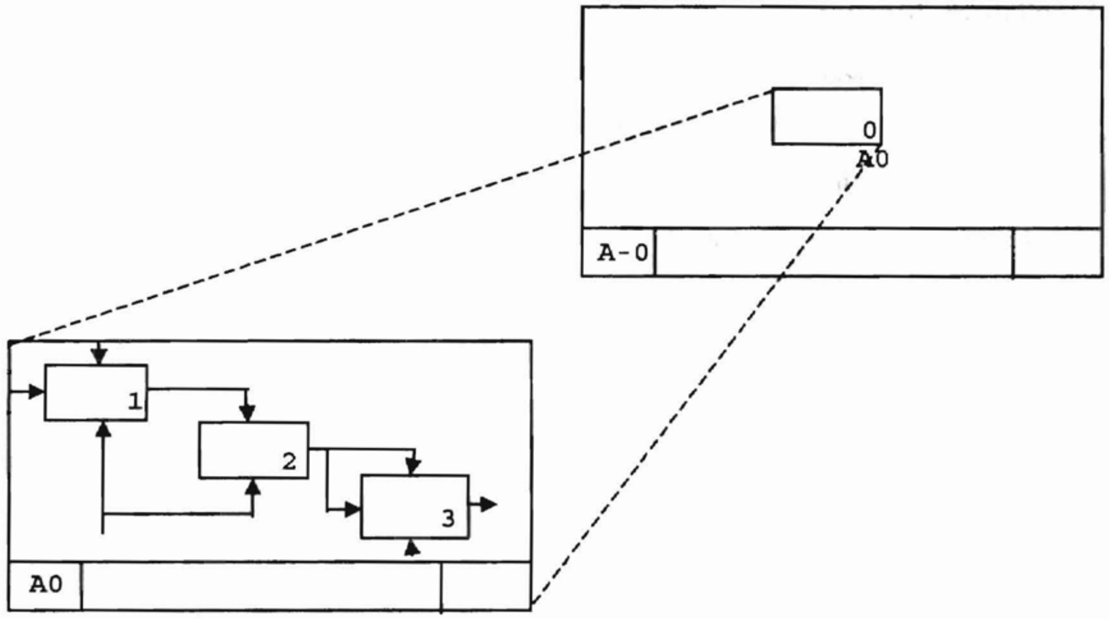


Figure 3.3: Example of Functional Decomposition in IDEF0

Strengths: An activity can be broken down to its lowest sub activities using a structured approach. IDEF0, therefore, supports structured analysis. The associated methodology is well defined, and an average user can construct an IDEF0 model with limited training in the use of IDEF0 diagrams. The graphical language has a well-defined syntax. For example, IDEF0 has an excellent numbering scheme to link parent and child diagrams in a hierarchical set. It became the Federal Government standard for information processing in the early nineties.

Limitations: The most common misinterpretation about IDEF0 is that 'it is a sequence of activities/diagrams'. It is important to note that IDEF0 assumes sequence independence, although the activity blocks can be arranged in such a way (from right to left – sequentially) that it could represent a workflow. However, it is left to the discretion of the user to model a sequence or not. The notion of time is not included in IDEF0. It can become difficult to identify and distinguish between inputs and controls (triggers) for a given activity. Decision logic is not represented in IDEF0. This technique does not support prescriptive analysis. This is because of lack of a formal supportive theory.

3.4. IDEF3 – A Process Description Capture Method

IDEF3 is a process description language and not a self-contained modeling tool (www.idef.com). It was developed to capture details of a process as described by a person actually involved in the process. It is a structured method for capturing details of a process and has two distinct representations of the process within the boundary of the process. This boundary is termed as a scenario. A process can be represented by a sequence of activities using a "process description diagram," (see Figure 3.4) in which each activity is represented by a rectangle termed as a "unit of behavior (UOB)." The process sequence is represented by a number of UOBs connected together by directed arcs to represent control flow. The precedence, causality, and logical relationships within a scenario are captured in this process description diagram. The second representation in IDEF3 is termed as Object State Transition Network (OSTN) diagram (Figure 3.5). The OSTN diagram describes what happens to an object as it passes through a sequence of activities. This representation focuses on objects involved in the process and their state change behavior in a single or multiple scenarios. Object states are represented by circles, and state transition links are represented by directed arcs.

Strengths: IDEF3 has a clearly defined methodology for describing the details of a process. By separating the process description into a process description diagram and an object centric diagram, one can construct multiple views of the same process. This is beneficial when users with different backgrounds want to have a description of the same process. Precedence, causality, and logical relationships are represented in IDEF3 diagrams. IDEF3 has a clearly defined syntax

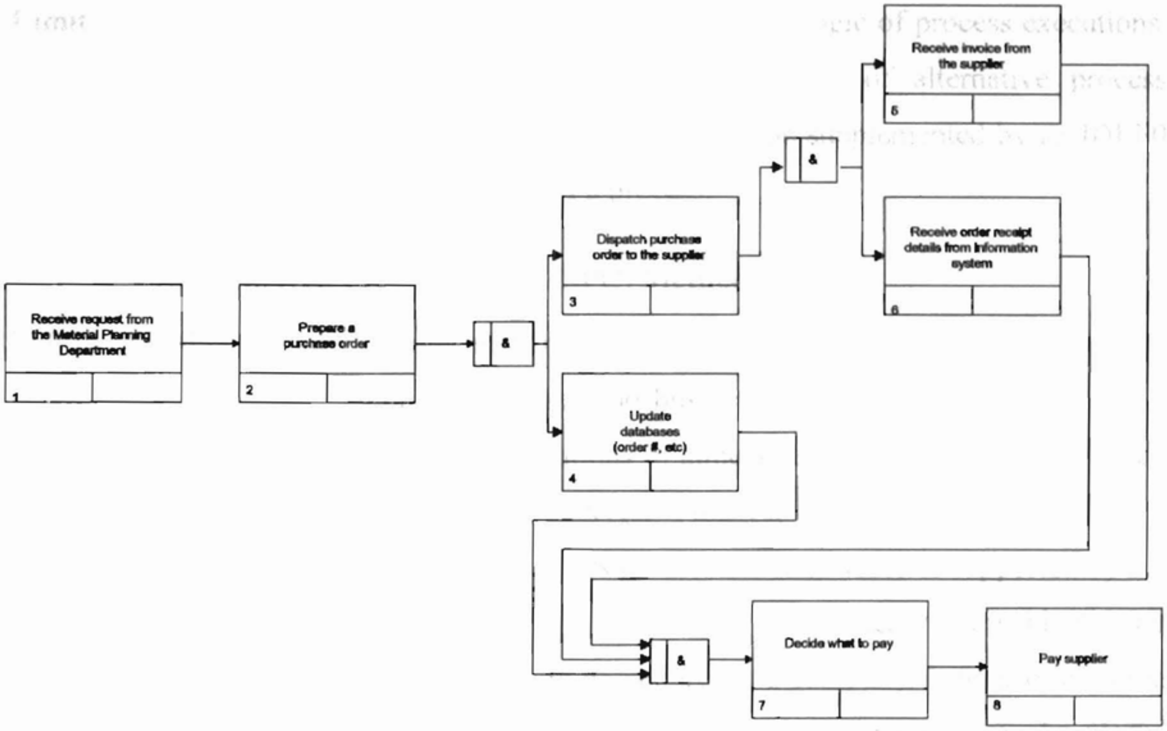

Receive request from the Material Planning Department
1
Prepare a purchase order
2
&
Dispatch purchase order to the supplier
3
Update databases (order #, etc)
4
&
Receive invoice from the supplier
5
Receive order receipt details from Information system
6
&
Decide what to pay
7
Pay supplier
8

Figure 3.4: IDEF3 Process Description Diagram

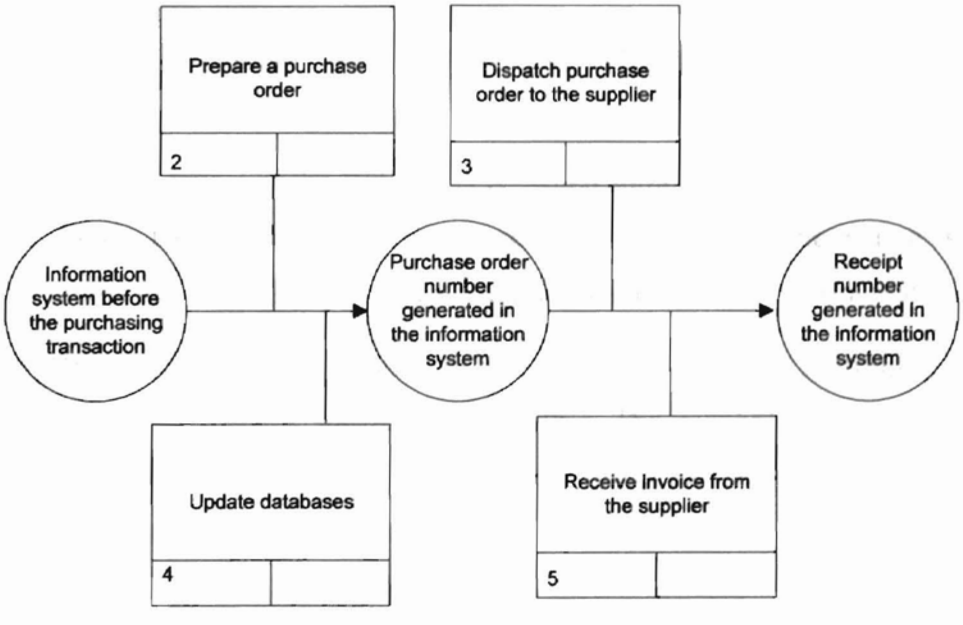

Prepare a purchase order
2
Dispatch purchase order to the supplier
3
Information system before the purchasing transaction
Purchase order number generated in the information system
Receipt number generated in the information system
Update databases
4
Receive invoice from the supplier
5

Figure 3.5: IDEF3 Object State Transition Diagram

Limitations: The goal of IDEF3 is to describe in detail the logic of process executions. An IDEF3 description can be used to support simulation of alternative process implementations. However, an IDEF3 description should be supplemented by an IDEF0 representation to capture complete details of a process.

3.5. SAP's Event-driven Process Chain (EPC) Method

SAP is a leading provider of inter and intra-enterprise software solutions that integrate the processes within and among enterprises and business communities (www.sap.com). SAP's R/3 software package provides solutions to almost all business applications such as Sales and Distribution, Human Resources, Finance, and Production. SAP has developed the R/3 reference model (a business process model), which is supported by the R/3 software. The SAP R/3 reference model is primarily intended for identifying possibilities for optimizing the routines and procedures in a company. It focuses on three basic design principles that are always relevant when a company is analyzed for reengineering, and they are:

- A task or a function which describes what is to be done.
- An organization which describes who should be doing the task.
- An information object which describes what information is needed to process the job.

The most important aspect of the R/3 model is "when something should be done", that is, the control logic. This is of prime importance in today's world where time is money. SAP's R/3 model has two main goals: customer orientation and model orientation.

Customer Orientation: The description of processes is captured in a group of symbols, which are clear and have a meaningful syntax. The symbols are in a particular arrangement so that a person can understand and grasp the process structure quickly. This method of arranging the symbols is known as the Event Driven Process Chain (EPC) method.

Model Orientation: In order to avoid an information overload when the size of the model gets too large, the R/3 model describes the business processes that are commonly needed in practice along with their variants.

Event Driven Process Chain: EPCs are made up of active components that do things (functions) and passive components that only come into play in response to certain business situations (Keller and Detering 1996). The events, which are represented by hexagons, act as triggers for the functions. It is not easy to identify events in a process, but once they are identified, the resulting EPC acts as a powerful control flow logic for the organization. The constructs and their meaning are shown in Figure 3.6. The example in Figure 3.7, taken from Keller and Detering (1996), illustrates the use of the constructs to create a process model that clearly captures the control logic. The process described by the constructs is the flow of goods in a company.

Navigation between individual process models is done with the aid of initial and final events. In order to analyze business processes and their variants, a "Lean EPC" diagram can be constructed with only functions, events and the logical operators. To handle organizational questions, an "EPC Assignment diagram" can be constructed with the organizational units and the information units along with their inputs and outputs. Thus, the EPC can be broken down into two diagrams for business process flow analysis and organizational/information flow analysis.

A striking feature of the R/3 architecture is the process selection matrix which comprises standard business processes. The columns represent the business processes and the rows represent the scenarios or the variants of the business processes. The first step in modeling is to arrange the elements of the matrix as per the business process. The process selection matrix allows R/3 users to get into the process models of R/3 and answer questions about the characteristics of each business process and its logical time sequence inside of the defined scenario (Keller and Detering 1996).

Strengths: By connecting events with the functions using logical operators, and by forming a sequence of events and functions, a complex business process can be defined quite accurately. The focus of the R/3 reference model is complete chains of processes in response to business events. This is done to a very limited extent in IDEF0. This could play a vital role when we deal with enterprise structures such as supply chains. Apart from describing the control flow, an EPC diagram captures the information/data necessary towards the development of an information system. It also describes which department is involved in the process. The process selection matrix assists the user in the modeling process with standard business processes and scenarios.

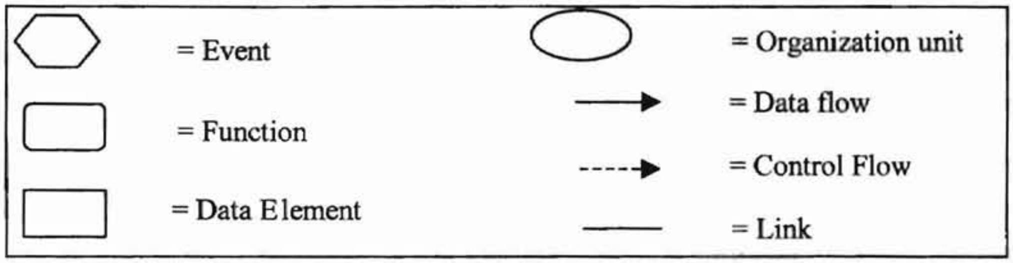


Figure 3.6: Constructs for the EPC Method

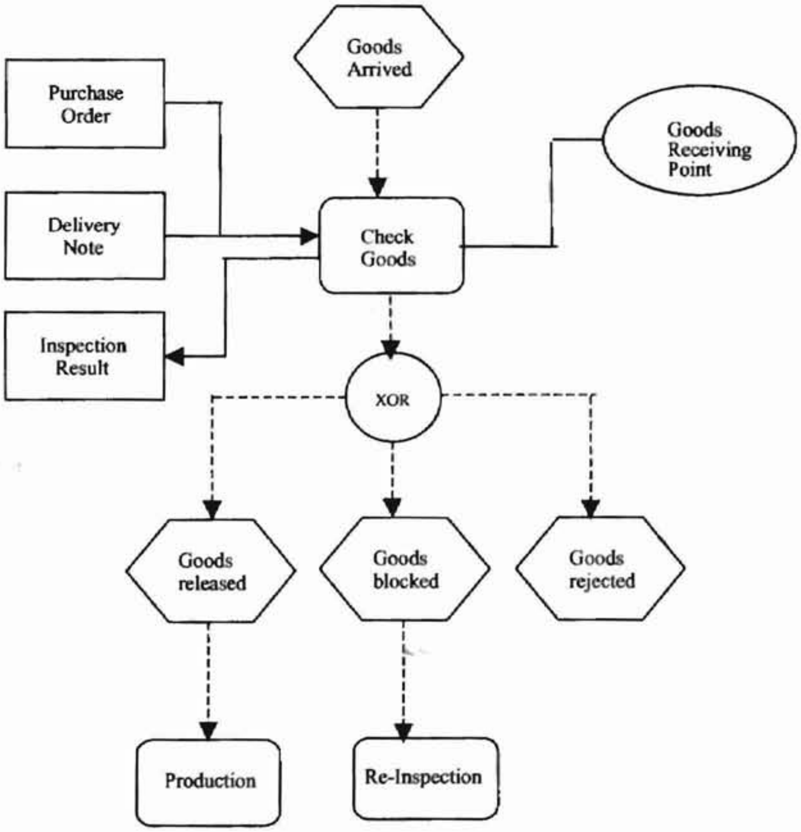


Figure 3.7: Example of an Event-driven Process Chain

Limitations: Unlike other approaches, the SAP R/3 model does not provide a methodology that would serve as a guide to the modeler. The R/3 model was not developed with a view to the creation of an information system, but to clarify the business solutions of the R/3 systems in simple model screens. Although the process selection matrix can serve as a guide to the user, the company specific processes require iterative prototyping by choosing the appropriate elements of the matrix. This process has to be done carefully so that the company specific process can be modeled correctly.

3.6. CIMOSA – CIM Open System Architecture

CIMOSA is the outcome of the efforts undertaken at the European Strategic Program for Research and Development in Information Technology (ESPRIT) Consortium AMICE (Zelm *et al.* 1995). CIMOSA includes an enterprise modeling language, which supports model-driven enterprise in all the life-cycle phases of an enterprise. The modeling language is supported by a well-defined framework, architecture and methodology.

The framework has three dimensions – (i) model life cycle, (ii) views and (iii) instances of the model. The modeling process starts with a requirements definition model followed by the design specification model and finally the implementation model. There are four different views – Resource view, Function view, Information view and Organization view. There are three instances of a model, generic, partial and particular, which represent the levels of abstraction. A generic model is applicable for a broad domain whereas the particular model is for a specific set of business processes or in a specific enterprise. CIMOSA provides a set of constructs to build a particular enterprise model.

Figure 3.8, taken from Kosanke (1995) shows the basic building blocks of the CIMOSA business modeling process. The Domain process (DP) is at the highest level. Processes, events and enterprise activities are the object classes that describe functionality and behavior (dynamics) of the enterprise operation. Inputs and outputs of enterprise activities define the information (enterprise object) and the resources needed. Organizational aspects are defined in terms of responsibilities and authorization (organization elements) for functionalities, information, resources, and organization. They are structured in organizational units or cells. CIMOSA employs the object-oriented concepts of inheritance, structuring its constructs into a hierarchy of object classes (Kosanke 1995).

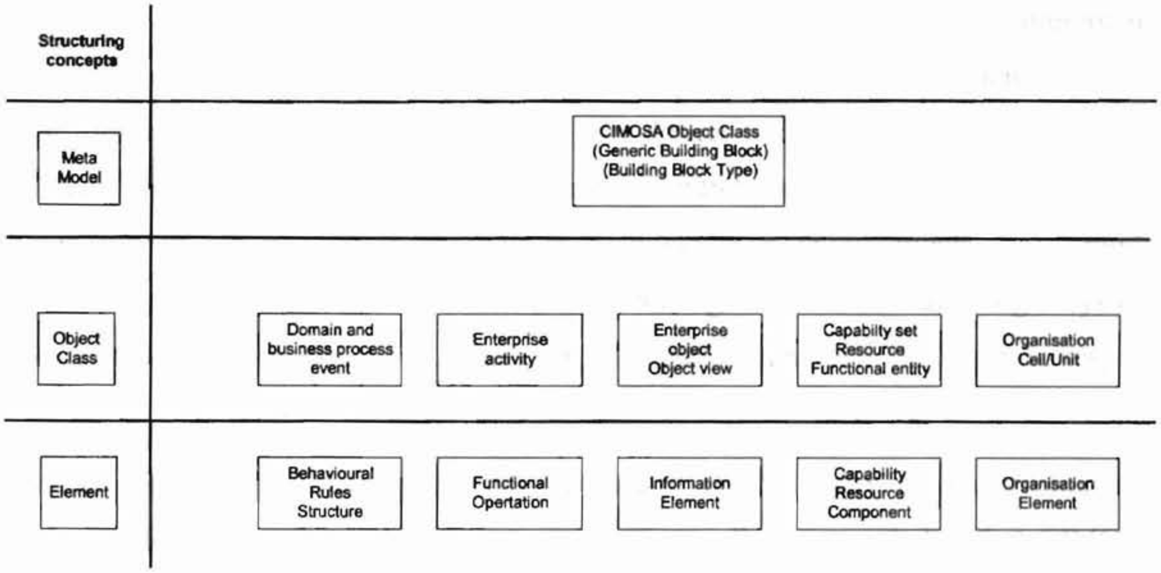


Figure 3.8: CIMOSA Business Modeling Constructs

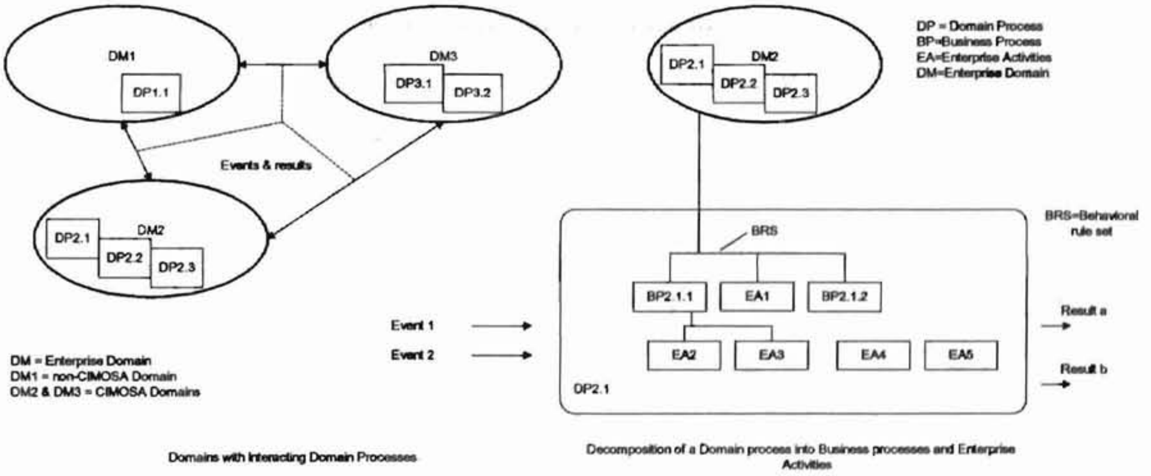


Figure 3.9: CIMOSA Domains, Business Processes and Enterprise Activities

The CIMOSA constructs are shown in Figure 3.9. The CIMOSA business modeling process starts with the identification of CIMOSA and non-CIMOSA domains. These domains are connected by events and results. The CIMOSA domain is then broken down into Domain Processes (DP) and Business Processes (BP) at the next level, which are further broken down into Enterprise Activities (EA). The EAs are linked by a set of behavioral rules (BRS) that represent control flow.

Strengths: CIMOSA supports the life cycle of an enterprise from the requirements definition stage to its implementation. It has a well-defined framework which provides a clear idea as to at what level (instance) the user desires to model. The results of a study conducted (Didic 1994), demonstrated that CIMOSA offers valuable concepts for integration of information technology in manufacturing environments, from model creation to model execution. CIMOSA gives useful guidelines in structuring a system. The link between process modeling and model execution provides flexibility to respond to changes in the enterprise processes.

Limitations: The results of the study conducted on CIMOSA model creation and execution for a casting process and manufacturing cell also concluded that CIMOSA architecture is neither complete nor consistent (Didic 1994). CIMOSA concentrated more on the architectural framework. Effort on system detail is necessary for a successful creation and execution of a CIMOSA system. Methodology for the transition from a general to a partial and then to a particular model of an enterprise is not defined. The functional view shows a generic workflow control, but not to the extent of the EPC in the R/3 reference model. The behavioral rule set has to be correctly defined to initiate the correct business processes or enterprise activities. The R/3 representation provides a more natural way of specifying the events. The decomposition of a domain into domain processes and business processes at the next level, which are further broken down into enterprise activities, can create confusion for the user when classifying processes and activities.

3.7. Integrated Enterprise Modeling (IEM)

IEM was developed to support computer integrated manufacturing; however the concepts can be extended to non-manufacturing enterprises. The concept of IEM is that the representation of the different aspects of manufacturing enterprises as views of one unique model (Mertins *et al.* 1992 and 1997). IEM uses the object-oriented modeling approach, and applies its three main features – encapsulation (the close relation between functions and data of an object), inheritance and the class concept. The kernel or meta model of IEM comprises two main views, a generic activity block, and three operands or objects of a manufacturing enterprise. The two main views are the function view and the information view. The concept of encapsulation in the object-oriented approach links the two views. The three objects of a manufacturing enterprise are: *product, order and resource*. This classification in IEM is derived from the three main questions within a manufacturing enterprise, what to order – *product*, how and how many – *order*, and what is required to manufacture the product – *resources*. These objects, viewed with different perspectives, link both views - function and information with each other. The basic building block of IEM is similar to that of the IDEF0 technique and is shown in Figure 3.10. The generic activity model (GAM) of IEM is shown in Figure 3.11.

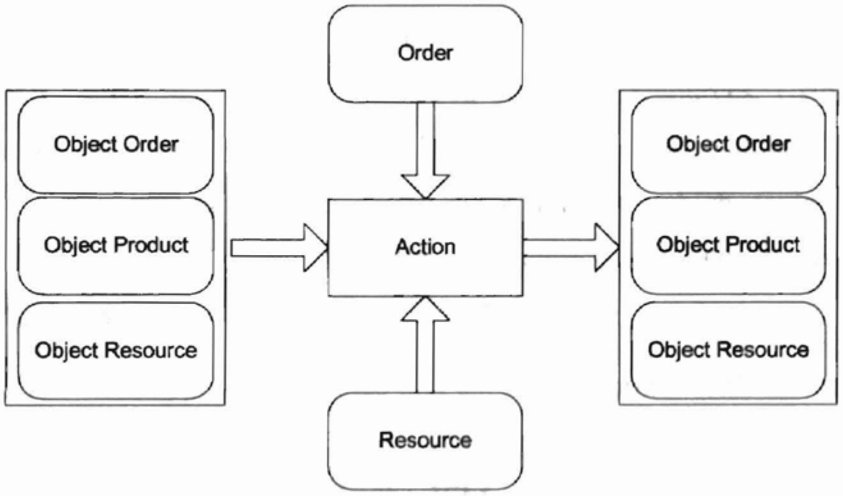


Figure 3.10: Basic Building Block of IEM

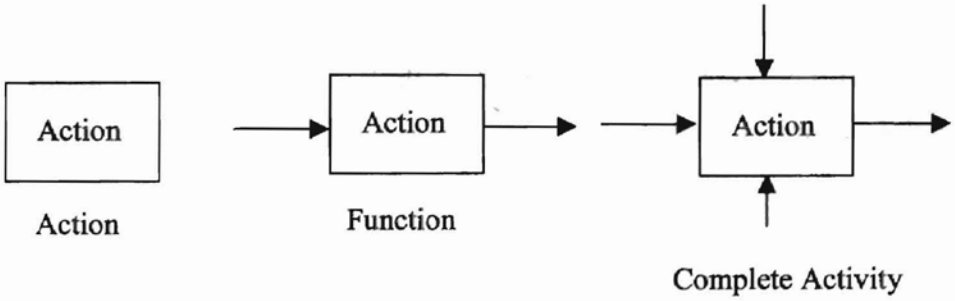


Figure 3.11: IEM General Activity Model

First, the action is described in the rectangular block. The input and the output objects are identified in the next step, which results in the function block. The trigger (order) and the resources when added to the function block result in a complete activity model.

The functions can be linked together to form networks of functions in an enterprise. The complete activity blocks can be linked together to yield a particular model for an enterprise. There can be a standard or common set of particular models which the user can manipulate to suit specific processes in any enterprise. The objects, which were identified during the functional modeling, can be studied for the data which they contribute. This data can be represented or structured in three models, product, order and resource in a single database of the information system.

Strengths: The kernel of IEM tries to accommodate all entities in an enterprise as subclasses of the three main classes – product, order and resources. Using the object-oriented concepts, we can relate and link all objects with their relations. In an object-oriented approach it is customary to define class libraries, which contain standard or commonly used objects or sub classes. IEM has a class library with three broad views, namely production processes, technical views and non-technical views.

Limitations: The IEM approach assumes that users are familiar with the object-oriented concepts, which might not be the case. Without the knowledge of object-oriented concepts it becomes difficult to apply the IEM method. The classification of objects under the three main classes would itself be difficult. IEM assumes that views cannot be predefined, but can be defined as the model is being developed depending on the perspective required. Hence, instant creation of views is difficult. Moreover, if the modeling constructs do not support the required views then the usefulness of the model will be limited. Taking into account all of all these drawbacks, it is difficult to say that IEM integrates all views into one consistent model, although it aims at doing so. IEM does not provide a clear modeling methodology.

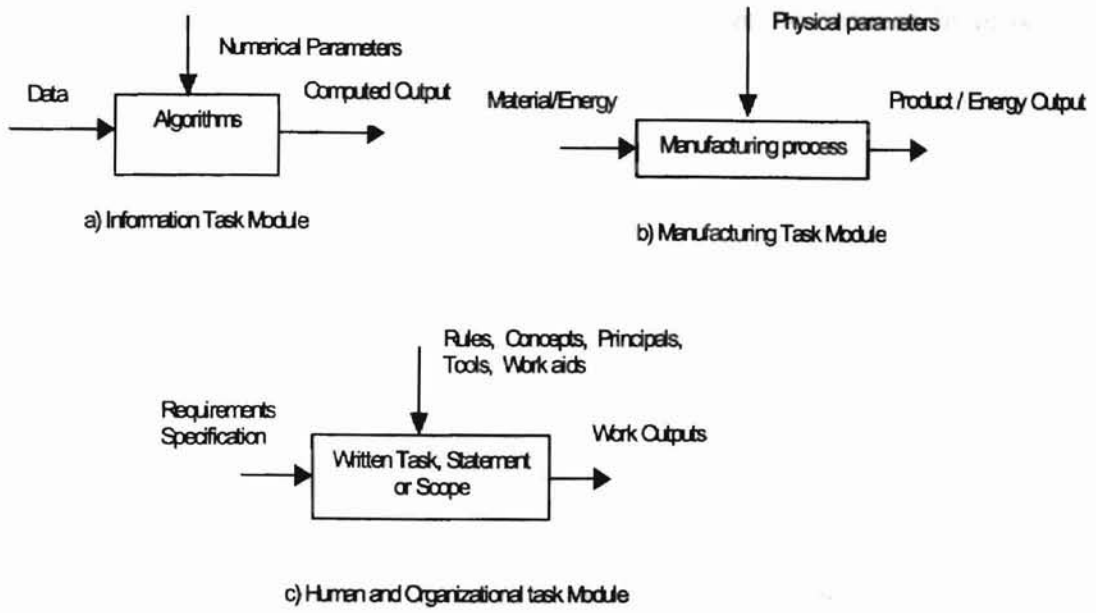
3.8. The Purdue Enterprise Reference Architecture (PERA)

The uniqueness of PERA lies in the definition of the human tasks and functions in enterprise modeling (Williams 1994). PERA categorizes enterprise requirements into information and physical manufacturing tasks. A group of connected tasks comprises a function. PERA covers the entire life cycle of an enterprise, right from the mission definition to the obsolescence of the enterprise. The PERA architecture has two views – functional and implementation. The implementation view covers definition, design, construction, and installation and operation phase. Each phase has its own definition of the three architectures – *Information, Human, and Manufacturing*. Within PERA, "automatability line" defines the maximum extent to which a task can be automated beyond which human intervention is required. The "humanizability line" defines the maximum extent to which a human can perform a task and beyond which a task needs to be automated. The actual extent of automation depends on the economic, social and technical limitations.

PERA uses a bottom-up modeling approach. The basic construct of PERA reflects the IDEF0 building block without the representation of a resource. The resource aspect is partially captured by having three variations of the task module (Figure 3.12). The algorithm in the information architecture can be mathematical, computer based or even a descriptive flow chart or sequence of statements. A network of these tasks forms functions.

Strengths: Explicit definition of human roles is not found in any other framework or architecture. PERA methodology covers the entire life cycle of an enterprise.

Limitations: PERA deals with a development of a master plan like in an ISO 9001 or a QS 10000 quality manual. It does not focus on how to model an enterprise, but focuses on the action plans that are a set of actions to be followed in an enterprise. Due to the lack of a meta model like in the CIMOSA Framework or the process selection matrix in the SAP R/3 reference architecture, it seems that PERA does not support a model-driven enterprise. It tries to model an enterprise using the bottom-up approach. It would be difficult to integrate all these tasks to form a complete enterprise model. The basic modeling construct is similar to that of IDEF0, and could include the hierarchical representation for integrating tasks into higher level processes.



a) Information Task Module

b) Manufacturing Task Module

c) Human and Organizational task Module

Figure 3.12: PERA – Different Task Modules

3.9. GRAI-GIM (GRAI Integrated Model)

The purpose of GRAI-GIM is to support the designer of a computer integrated manufacturing system in describing the CIM system and deriving its definition (Chen and Doumeingts 1996). There are three aspects to the model, GRAI conceptual reference model, the formalisms and the structured approach. In the reference model, as shown in Figure 3.13, the manufacturing system is divided into the physical model (people, machines, etc.) and the control system, which controls the physical system. The control system is further decomposed into the decision model and the information model.

The decision model operates at various levels of the enterprise and the three broad categories of decision-making levels are strategic, tactical, and operational. The decisions are taken on the functional entities namely, product, resource, and planning. Note that this classification corresponds exactly to the IEM classes. The decision-making levels and the functional decomposition leads to a two-dimensional matrix which is shown in Figure 3.14. The physical hierarchy is related to the decision hierarchy. Each decision level controls a specific physical process (level). The information required for a decision is provided by the information system at any level, but the information is filtered and only relevant information for the level is provided. A specific decision level is called a decision center.

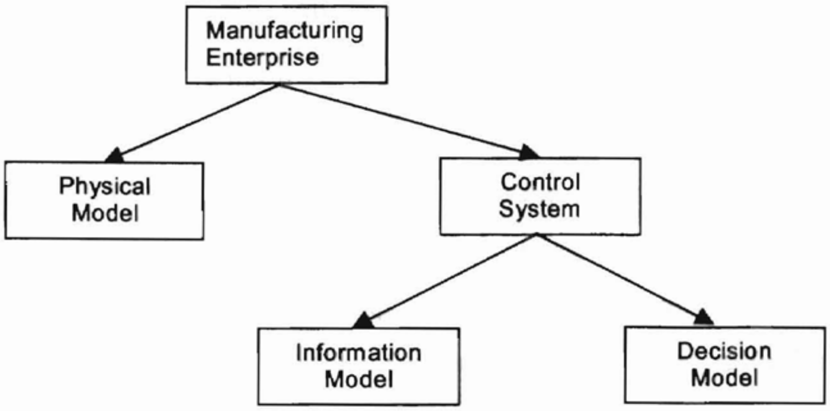


Figure 3.13: GRAI-GIM Reference Model

Functional Decomposition (View)

Hierarchical Decomposition	Product Mgt	Planning	Resource Mgt
Strategic			
Tactical			
Operational			

Figure 3.14: GRAI-GIM 2-D Matrix

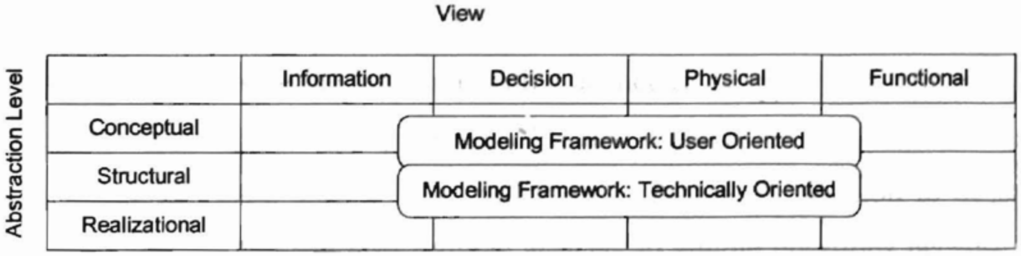


Figure 3.15: GIM Modeling Framework

The approach in the GRAI-GIM methodology is to build a user-oriented model first and then the technical model. In the matrix in Figure 3.14, a functional view is added and the modeling framework is shown in Figure 3.15.

The user-oriented model and the technically-oriented model have to be tightly coupled. The constructs used for modeling any element of the grid are: entity relationship diagram (Song and Froehlich 1994) for conceptual and structural modeling of information, and the GRAI grid and network for the conceptual and structural decision model. The flow can be represented by IDEF0 diagrams. Imposing data coherence does validation of the complete model. The structured approach focuses on three phases, initialization, analysis and design. Implementation is not a focus in GIM.

Strengths: The separation of the control system into decision and information is a unique feature of GIM. This can help in proper structuring of the decision/control hierarchy and also help in the structuring of the information system.

Limitations: GRAI-GIM is only concerned with the analysis and design phase of an enterprise. The design specification can only serve as a guide to the implementation phase. The purpose of separating the user oriented model and the technically oriented model is to help the users capture the processes independent of the technology. The developments in technology will change the technically-oriented model. However, a strong link has to exist between the technically-oriented model and the user-oriented model, both must be current with respect to each other. The prescriptive ability of the model lies in the choice of the modeling tool for the implementation phase.

3.10. TOronto Virtual Enterprise (TOVE) Method

TOVE is an enterprise modeling language with a deductive capability. It defines a shared terminology for modeling the enterprise activities and implements semantics in "a set of axioms" (Fox *et al.* 1993). TOVE is implemented in Prolog. In addition to describing enterprise activities, TOVE can also provide answers to simple questions because of its deductive capability.

Unlike CIMOSA or other frameworks/architectures that decompose functions into lower level blocks, TOVE uses a single construct called "Activity." An activity can be performed when something enables it. When an activity is performed, something is caused. The basic activity block is shown in Figure 3.16.

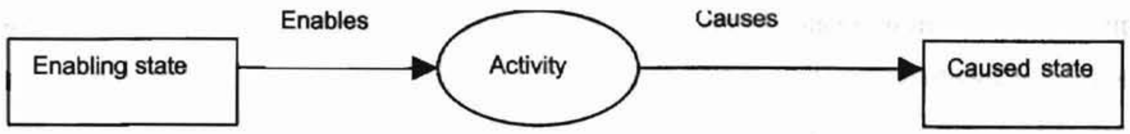


Figure 3.16: TOVE Activity State Model

An activity along with its enabling and caused states is called an activity cluster. Different types of states are defined; they are broadly classified into terminal and non-terminal states. "Consume" and "Release" are two examples of a terminal state. Non-terminal states include Boolean combinations like AND, exclusive OR, and inclusive OR. Time is represented by a temporal relation between the terminal state and the activity, which is specified by the user. The relationship between a non-terminal state and an activity can be automatically deduced using a logical set of temporal relations. Resources (material, equipment, etc.) are assumed to be temporally or physically divisible. The terminology about activity, states, time and resources are clearly defined and then represented in the form of first order logic in Prolog. The concept of an activity and its enabled and caused states is similar to the concept of Petri nets, that have been widely used for modeling and analyzing systems.

Strengths: Compared to other languages, TOVE is an exception because of its underlying theory base. The combination of deductive and descriptive abilities is not found in any of the other enterprise modeling languages.

Limitations: The user interface is limited to an activity cluster diagram that relates activities with each other. The graphical symbology does not extend beyond the activity and its states. This reflects poorly on user friendliness. TOVE lacks a reference architecture or framework and the modeling methodology is unclear. The concept of views and life cycle is not defined in TOVE. Although the underlying formalism is appealing, it might be hard to implement.

3.11. Baan's Dynamic Enterprise Modeling (DEM) Technique

Baan's Dynamic Enterprise Modeling tool is essentially a Petri net representation of a process (van der Rijst 1997). A Petri net comprises sets of places, transitions, and input and output arcs. Places are connected to transitions and transitions to places by directed arcs (Srihari *et al.* 1990; van der Aaslt and van Hee 1995). The constructs used in DEM modeling are shown in Figure 3.17.

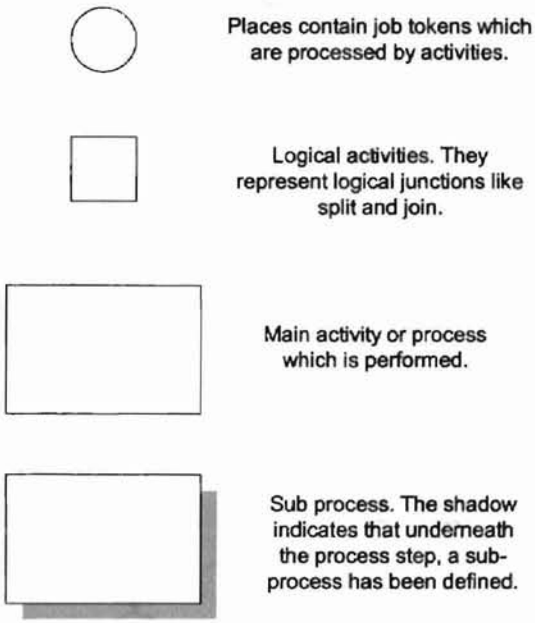

Places contain job tokens which are processed by activities.
Logical activities. They represent logical junctions like split and join.
Main activity or process which is performed.
Sub process. The shadow indicates that underneath the process step, a sub-process has been defined.

Figure 3.17: DEM Modeling Constructs

The logical activity can be an AND, inclusive OR, or an exclusive OR. The type of the logical connection has to be mentioned on the graphical representation. DEM methodology provides guidelines to the user for modeling a business process. DEM has conventions for representing a process in the Petri net form. DEM has structured processes at two levels, namely the main processes and the detail processes. The main processes cover more than one business function, whereas the detailed processes contain the detailed activities. DEM is implemented in Baan's ERP suites. Figure 3.18 shows a manufacturing example in DEM.

Strengths: Petri net possesses powerful process analysis capabilities (Zurawski and Zhou 1994). Hence representation of a process in a Petri net form provides scope for analysis in addition to a theory base for the modeling language. The modeling approach is event-based with emphasis on control flows. The graphical language uses only a few symbols, and is easy to understand.

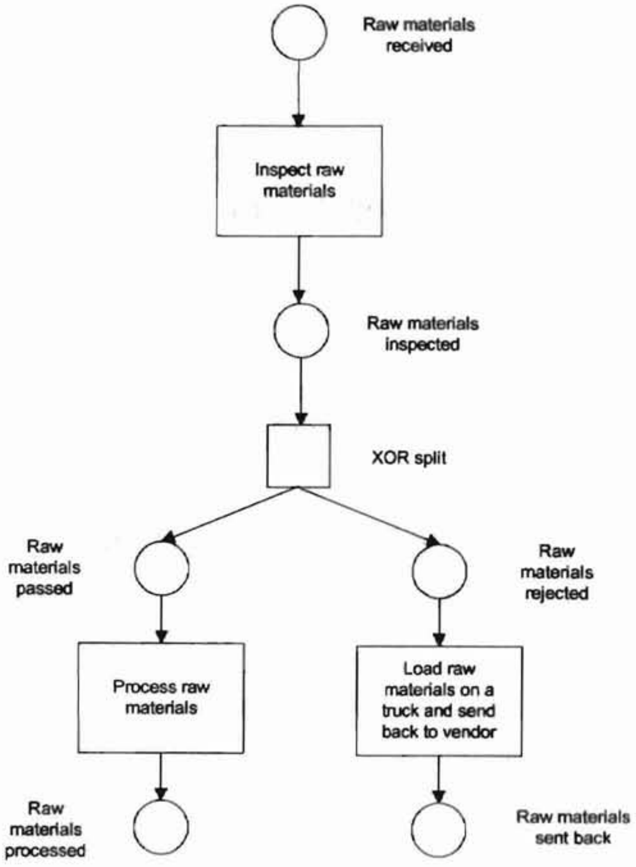


Figure 3.18: Manufacturing Example in DEM

Limitations: The DEM graphical representation does not show data flow. Since control activities or logical operators are represented by the same construct, the type of the operator, that is AND, OR, or an XOR, has to be captured using a label. As each activity block is preceded and succeeded by a place, the description of a process could become lengthy.

3.12. Generalized Enterprise Reference Architecture and Methodology

GERAM was developed by evaluating several existing modeling languages, and the result is a generalized reference architecture for enterprise modeling. GERAM claims that it is not another architecture; instead it serves to organize the existing architectures

(GERAM 1999). However, it combines the concepts from most of the architectures described earlier in this chapter, and has its own framework and methodology. GERAM categorizes modeling levels that are shown in Figure 3.19.

Strengths: GERAM has generic concepts covering wide areas related to enterprise modeling in the form of a reference architecture.

Limitations: GERAM does not have an enterprise modeling language of its own. It suggests the use of any suitable modeling language to model specific portions of the enterprise, and then suggests integration of the models. According to GERAM, requirements of the modeling constructs should include human roles, activity based management, etc. However, no details of implementation are given in GERAM. It emphasizes model portability and interoperability in the information world with model-driven operational support by providing real-time access to the enterprise environment. GERAM also hints at including the economic aspects into consideration while modeling. GERAM defines in a broad sense, what an ideal enterprise modeling architecture or framework should be capable of doing, but does not provide all the details.

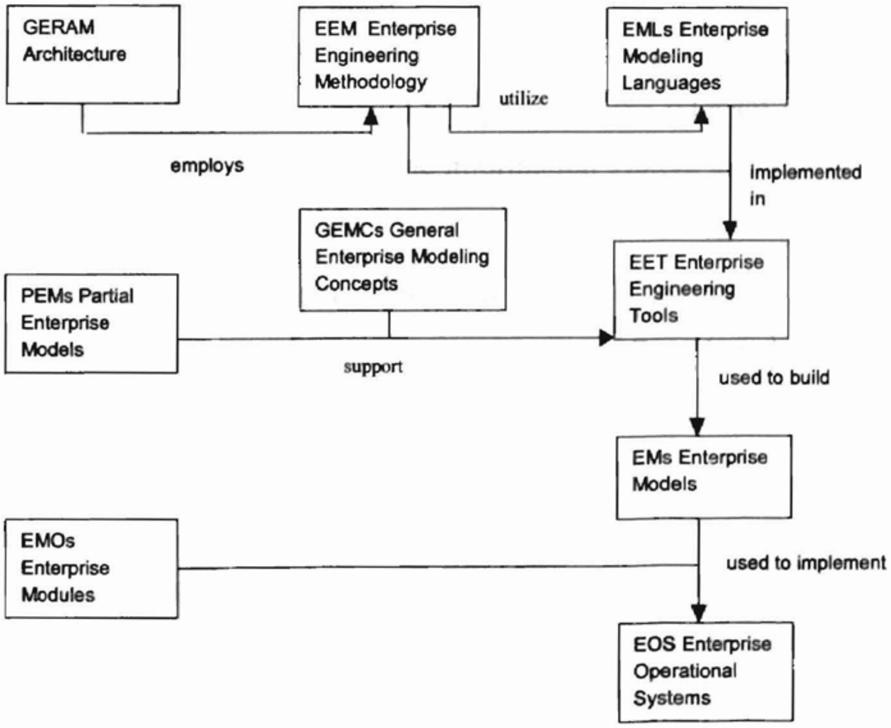

GERAM Architecture
EEM Enterprise Engineering Methodology
EMLs Enterprise Modeling Languages
utilize
employs
implemented in
GEMCs General Enterprise Modeling Concepts
PEMs Partial Enterprise Models
EET Enterprise Engineering Tools
support
used to build
EMs Enterprise Models
EMOs Enterprise Modules
used to implement
EOS Enterprise Operational Systems

Figure 3.19: GERAM Framework Components

3.13. Architecture for Integrated Information Systems

Architecture for Integrated Information Systems (ARIS) (Scheer 1992), defines an architecture for a complete, enterprise-wide information system. ARIS defines three views, namely, the functional view, information view and organization view. These views are defined in all life cycle phases of the information system, namely, requirements definition, design specification, and implementation description. All three views are treated in isolation, and the relationships between the three views are represented by a control view. ARIS architecture is derived on the basis of a structured process chain which is event - based. Figure 3.20 shows an example of a process chain for a general production process.

The "production process" is triggered by the event "production order" and the resultant event after the execution of the process is a "finished production order." During the process, many components that are related to the process are either consumed, transformed or used. For example raw material is transformed into finished part. The resources required for the transformation of material are the machining center and employees (workers) from one or many organizational departments. The activity results in some information flow; namely, information pertaining to inventory levels of raw materials and the number of parts would be updated in the relevant databases. The term "environmental conditions" is used for the information system and similar or supporting media to absorb all components of the information system and is represented by a circle.

The data view comprises the start and end events, and the environmental conditions. The organization view comprises the organizational unit and users/employees. The resource view comprises information technology equipment, machines, etc. The process or the function view comprises the processes.

Information systems are classified into 3 levels of abstraction, from general to specific objects in a manner similar to object-oriented concept of classes and objects. For example, a set of similar elements like events or processes can be grouped under one class, and the relationships between the elements of each class can be established with the object-oriented concepts of polymorphism, hierarchy, etc.(Pascoe 1986).

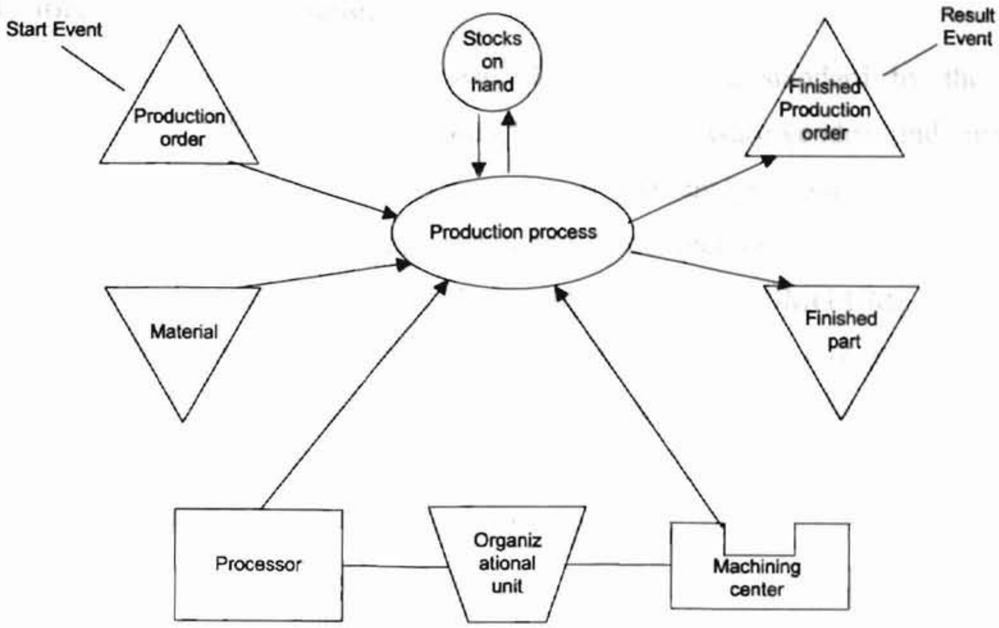


Figure 3.20: Basic Building Blocks of an ARIS Process Model

Using the process chain as a starting step, the ARIS architecture supports the development of the complete, integrated information model for an enterprise. The main building block in the development of the information model is constructing the process chain. Process improvement can be done after analyzing the process chain using decision-support tools. The changes/improvements in the processes can directly change the relevant information in the repository, since the process view is connected with all the other views (data, function, etc) by the ERM.

There are striking similarities between the event-driven process chain in SAP's R/3 reference model and the process model in ARIS. Both stress on an event-driven approach to process modeling.

Strengths: ARIS information architecture is derived on the basis of an event-driven process chain. Most of the relevant subjects pertaining to process modeling are captured using graphical constructs. When the process model is transformed into an information model, the result is an integrated enterprise model.

Limitations: Methodology related to the development of a process model is not clear. Process modeling is not elaborately described within ARIS.

3.14. Unified Modeling Language

Unified Modeling Language (UML) was developed as a standard by the Object Management Group (OMG), a consortium of over 800 software vendors and customers. "UML provides system architects working on object analysis and design with one consistent language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling (OMG UML specification 1999). One of the main objectives of UML is to provide a visual tool that has platform independence. UML is structured architecturally and organized by packages. Each package has the definition of the abstract syntax, rules and semantics. UML defines following set of diagrams based on views of a system:

- Use case diagram
- Class diagram
- Behaviour diagrams
- State chart diagram
- Activity diagram
- Interaction diagrams
    - Sequence diagram
    - Collaboration diagram
- Implementation diagrams
    - Component diagram
    - Deployment diagram

A detailed explanation of these diagrams is given in OMG UML specification (1999). These diagrams provide multiple perspectives of the software system under development. Activity diagram is the tool used for process modeling. Within the activity diagram, the constructs shown in Figure 3.21 are used for process modeling. An example of activity modeling in UML is shown in Figure 3.22, adapted from Marshall (1999).

An approach based on UML which uses activity diagrams to represent business processes, is documented in Schader and Kortahus (1998).

Strengths: UML is emerging as a widely accepted modeling standard, particularly in the software industry. It provides different perspectives of the system under consideration. UML standard is a convergence of best practices in the object-technology industry.

Limitations: UML standard does not prescribe a specific process modeling approach. Activity diagram is a tool that can be used for process modeling. There are many sets of diagrams within UML, all of which may not be relevant for process modeling.

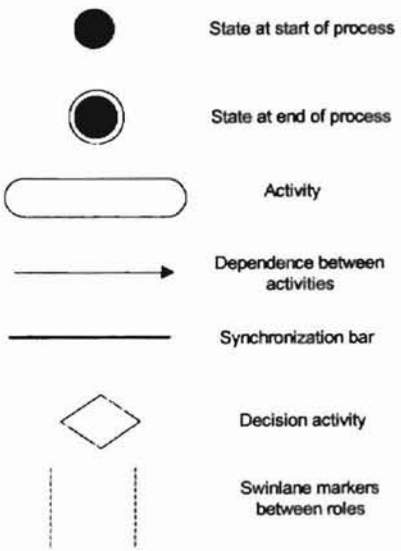

State at start of process
State at end of process
Activity
Dependence between activities
Synchronization bar
Decision activity
Swinlane markers between roles

Figure 3.21: UML Activity Diagram Constructs

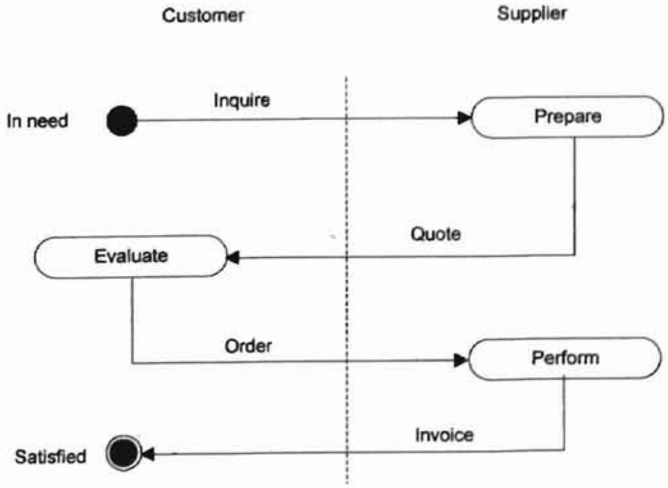


Figure 3.22: UML Activity Diagram Example

3.15. Summary

In this chapter, we reviewed thirteen enterprise modeling efforts with a focus on their process modeling capabilities. A thorough review of existing enterprise process modeling languages reveals that no single language has all required characteristics for a complete graphical representation of a process. These characteristics are summarized in the following chapter.

4. Characteristics of an Enterprise Process Modeling Language

As described in the introduction, one of the basic problems in creating a model-driven enterprise is the lack of a single modeling framework. Such a framework would serve the dual purpose of describing the business processes in an enterprise, and then performing analysis in order to help formulate alternatives for improvement. The scope of this thesis is limited to developing the modeling language that can provide descriptive ability within a modeling framework. Addition of the prescriptive ability will be another part of a NSF funded project underway at Oklahoma State University.

4.1. Characteristics

The conclusion from the literature survey is that no single modeling language possesses all the desirable characteristics. These characteristics are presented next along with a discussion of related issues.

Descriptive ability

Some languages or techniques like data flow diagrams and IDEF0 are strong in capturing and describing process details. But once the process description is captured, they do not support any further analysis. The primary purpose is to capture the process details for documentation and communication. The models may support further analysis or implementation, but do so only in an ad-hoc manner.

Prescriptive ability

One of the purposes of modeling is to deduce ways of improving processes without actually experimenting on them in the real world. If present, logical and mathematical capability of a modeling language could serve this purpose by analyzing possibilities of improvement and suggesting alternatives. The capabilities to describe and prescribe could create conflicts. This is because prescriptive ability requires that the process be described in a format that a mathematical or logical technique can understand, whereas the descriptive ability lies in describing a process to the extent that all details are captured in a format that is clearly and easily understood by the user.

Hierarchy of constructs

Consider an example where a financial analyst would likely require analyzing an enterprise from the perspective of the cost of operations. A CEO of an enterprise would likely require only the major variables that would reflect on the cost. The CEO should be given the option of viewing the "economic view of the enterprise model" at a higher level. A financial analyst would require the finer details of the economic view and the cost information at the lower most level. For an economic view to exist, the modeling architecture should incorporate views or perspectives for different types of users. For a CEO and a financial analyst to work on the same view, the concept of data hiding and data encapsulation from the object-oriented principles could be applied. In other words, a CEO should be supplied filtered information as compared to an analyst who would require the finer details. This means that the model should be structured in a hierarchical set of constructs. Will this require a different set of constructs at the higher level than at the lower level; or can the same constructs be used for modeling and analyzing at all levels?

Control flow

As described in Keller and Detering (1996), control flow has become a key factor in defining the success of an organization. The language must provide constructs for modeling control flow. A process model should clearly describe when an activity can start, what other activities will the completion of this activity trigger and so on.

Dynamics representation – behavior

As per ISO 14258, enterprise models should be capable of representing sequentiality, events, actions, conditions, states, state changes, start states, end states, sequencing relationship between actions, and descriptions of transformation functions. Enterprise models should also be capable of representing time duration, dynamic performance of processes, and sequential phenomena after specific units of time. Dynamics can be captured in a process modeling language that is mostly event-driven or state-based.

Performance measures

The improvements suggested as a result of the analysis process should be quantifiable in the form of performance measures. Modeling or capturing performance metrics has not been a issue in many of the techniques reviewed in Chapter 3. This may become more important as we move towards model-driven enterprises.

Feedback

Some activities might require information or feedback from other activities that are performed at a different hierarchical level, or from activities which are in a different sub process. None of the process modeling approaches mentioned in chapter 3 address this issue. If there is a separate construct or syntax for feedback, the iterative processes in an enterprise can be represented.

User friendliness

A descriptive modeling language usually has the greatest scope for providing user friendliness. However, when prescriptive ability is added, user friendliness should not be compromised. For example, in TOVE the underlying logic formalism makes the language less user friendly as compared to a data flow diagram or an IDEF0 representation.

Clarity of semantics and syntax

This requirement is very obvious. A modeling language should have a well-defined meaning for its constructs, and should clearly specify the rules for putting the constructs together to describe a process along with other details like resource requirements, information requirements, trigger for the process, etc. There should be no ambiguity in the meaning of constructs.

Implications of the Internet

Nowadays, most businesses are linked to the Internet. The Internet has lead to new concepts such as business-to-business and business-to-consumer commerce. Processes might require data from any location in the world, and the Internet is the desired mode of data. The data required by processes can be separated into physical data and electronic data, and this aspect should be clearly represented in a model.

Integration with analysis

The graphical modeling language developed in this thesis will be linked to a back-end tool, which is Petri net-based. Analysis would then be performed using the Petri net-based representation. This would require that the front-end map into the back-end tool for ease of conversion of process data. The constructs and the model should be easily transformable into a Petri net. However, the development of the mapping of the front-end with the back-end representation is beyond the scope of this thesis.

4.2. Elements of a Process Model

The following is a list of elements of a process that could be captured in a process model.

- Activity, which is the description of a single unit of the process.
- Control flow, which captures the order in which activities are performed. Logical constructs like "AND, XOR, OR" are a part of control flow representation.
- Inputs to, and outputs from activities. These may be physical or electronic.
- Resources or the mechanism or medium to perform an activity. Resources can be broadly classified as human, machine and computer.
- Conditions or event triggers, which are the pre and the post-signals accompanying an activity.
- Constraints or rules, which include the procedures, guidelines, standards, requirements, policies that govern or guide an activity.
- Stores, containers, or buffers, which could be a computer database or a physical store
- Sources and sinks, which are entities external to the process under consideration.
- Activity duration, which specifies a standard time for the activity to complete.
- Link to an organizational unit.
- Feedback.
- Functional relationships between output and input.

5. Enterprise Process Modeling Language (EPML)

This chapter describes the constructs, their semantics and syntax that together define the new enterprise process modeling language (EPML). Each construct is explained with its semantics and notation. If necessary, an example is described to bring out the meaning of the construct.

5.1. Activity

Semantics

It would be worthwhile to refer to the definitions of process, sub process, and activity in Section 3.1 at this point. Usually, a process can be broken into a number of activities. The words task, activity and process have been used interchangeably in the literature. In some cases, as in PERA, tasks are grouped to form a function. For the purpose of this thesis, a process comprises many activities. It is left to the user to decide what can be classified as a process and what as an activity. An activity can comprise many (sub) activities. However, the formal classification of sub-activities or decomposition into tasks and sub tasks is not done in this thesis.

An activity takes place when inputs are transformed into outputs, with the help of enablers or resources that might aid the transformation. An activity takes a finite amount of time for the conversion of inputs into outputs. Since resources in some form or the other are consumed when the transformation takes place, execution of an activity "costs" an enterprise.

In this thesis, only one construct - "activity," is used to represent the transformation of input to an output at all levels of abstraction. The numbering scheme for the activity construct, which is explained in detail in the notation that follows, provides the capability to trace an activity at any hierarchical level in the process. If different constructs were used for representing the input-output transformation at different hierarchical levels, the user would be faced with the difficult task of classifying activities based on constructs.

Also, as new activities are added, it would be difficult to decide at what level of abstraction they should be added. CIMOSA has classified activities at different hierarchical levels (Section 3.6), and this can be a limitation from a modeling perspective. The purpose of having only one construct to represent the input-output transformation is to avoid this modeling difficulty, and provide flexibility to easily represent the transformation at any level of abstraction.

Notation

An activity is represented by a rounded rectangle as shown in the Figure 5.1. This notation is common in most of the enterprise modeling languages.

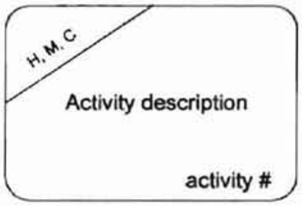


Figure 5.1: Activity block

The activity description is written in the center of the rectangle. The activity description should start with a verb, for example, 'assemble components', and 'process request'. Only when the activity is performed by an entity that is external to the domain being modeled, the description should begin with the name of the entity that performs the activity, for example, 'customer fills request' and 'bank pays loan.' This is necessary to clarify that an entity external to the domain being modeled, performs the activity. However, in a decomposition of the activity to a lower level of abstraction, it is not necessary to repeat the name of the external entity for the sub activities. The alphabets H, M or C at the top left corner of the activity block denote that the activity is performed by a human, machine or computer. This concept is taken from PERA (Williams 1994). All three entities may be required to perform an activity. In such a case, the three alphabets are separated by commas. Some processes are automated and do not require human intervention. Such activities are represented by only 'C' in the top left hand corner indicating that the activity is executed by a computer. The alphabets H, M, and C, provide quick visual feedback to the user in the sense that when a "H" appears in an activity construct, the user can quickly identify manual activities that are potential candidates for computerization/automation in a re-engineering effort.

The three entities, namely, H, M and C are treated as the three main resource sets under which all resources can be classified. The users could define sub classes as per their requirements. An activity might require many resources and specifying all of them in the activity block or around it would clutter the diagram. Hence, a separate table of resources and their quantity required could be maintained for each activity. In the software implementation, a user could specify the resources in the fields provided after clicking on the sets H, M and/or C.

Cost associated with the activity and the activity duration, are modeled as separate attributes of activity. There would be separate fields for cost and activity duration in the software implementation of the model. The activity number is specified at the bottom right hand corner of the activity block. Decimal points are used to separate the activity numbers at different levels in the hierarchy. An example of the numbering system is

3.12.17.21

This number indicates that the current activity is a fourth level activity because the activity number has 3 decimal points included in it. The activity is the 21st activity in the decomposition of the activity number 3.12.17. Likewise, activity 3.12.17 is the 17th activity in the decomposition of activity number 3.12. Likewise, activity 3.12 is the 12th activity in the decomposition of the activity number 3 in the decomposition of the root process. The purpose of the numbering system for activities is to link an activity at any level of hierarchy to its parent and child activities. However, a point to be noted is that if a new activity is to be added between two activities, at a later stage, the new activity will have to be given the last number at that hierarchical level. This can create confusion if one tries to associate activity numbers with the sequence of activities. An alternative way to keep track of the activities while drilling down the hierarchical levels is by providing a hyperlink between the activities in the software implementation. The choice between the numbering scheme and hyperlinks is an implementation decision.

5.2. Control Flow

Semantics

A process can be viewed as a chain of activities that follow each other in a sequence or take place in parallel or in some combination of both. When an activity is executed, it usually signals the start of another activity, possibly the next one in sequence. When an activity triggers the start of a succeeding activity, we say that control flow has occurred. Capturing control flow in a process is of prime importance and the SAP's event-driven process chain (Keller and Detering 1996) is based on the concept of an event triggering one or many activities.

Notation

A directed dashed line as shown in Figure 5.2, is used to represent a control flow.

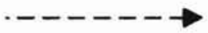


Figure 5.2: Control Flow

When the directed dashed line as shown in Figure 5.2 connects two activity blocks, it implies that when the preceding activity is completed, it triggers the beginning of the next activity in the sequence. This line does not represent data flow in any form. The succeeding activity can start only when all other conditions, such as other activity completions, if any, are satisfied.

Sometimes an activity may require a signal from the environment, which is external to the domain being modeled. This is viewed as a control flow from an external entity.

The symbol for an external trigger is shown in Figure 5.3.

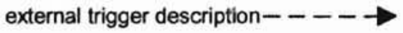


Figure 5.3: External Trigger

The symbol for an external trigger is same as the one used to represent control flow. However, the directed arc for a control flow is not labeled. An external trigger comes from a different domain, and hence, requires a label. The label should end with a verb, for example, "Invoice arrived," or "Material received," or "Data entered." This is different from an input because an input is an entity and not an event.

5.3. Data Flow

Semantics

Data could be the input to or output from an activity. Classical process modeling techniques like DFD, focused more on data flow because the flow of paper files and forms containing data controlled the execution of activities to a great extent.

Data flow can take place in two forms, physical and electronic. Presently, most of the data is available electronically. However, there might be instances where data might not be available in a computerized database because of technological reasons. Hence, separate constructs are provided for physical and electronic data flows.

Notation

A directed solid line with an embedded box as shown in Figure 5.4 represents physical data flow.

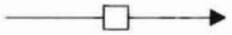


Figure 5.4: Data Flow

This line can be labeled at either end. If the flow is an input to an activity, the label is at the tail of the arc, and if the flow is an output from an activity, the label is at the arrow end of the arc. Data flow is shown only as an input or an output to an activity. It cannot be used to connect two activities with each other like a control flow. The label for an input and an output must be a noun or a noun phrase. A verb indicates that an event has taken place, whereas a noun represents an entity, and hence the convention of a noun/noun phrase for describing the inputs and outputs.

5.4. Electronic Data Flow

Semantics

New computer technologies have considerably eased the tasks of data storage and handling. Many activities require data, which is typically stored in a database.

Notation

The construct shown in Figure 5.5 represents an electronic data flow usually via the Internet or Intranets.

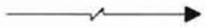


Figure 5.5: Electronic Data Flow

The interpretation is the same as in the case of physical data flow arc shown in Figure 5.4. The only difference is that instead of being a physical data flow, this is an electronic data flow. It can be the input to, or the output of an activity, and cannot connect two activities. It has the same naming conventions as that of the data flow arc.

5.5. Material Flow

Semantic

An activity, in many cases (e.g. manufacturing) requires raw material(s) as input to produce finished component as an output. This flow of material is represented by the construct shown in Figure 5.6.

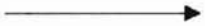


Figure 5.6: Material Flow

Notation

The construct shown in Figure 5.6 represents a material flow. The rules for connection to activity construct and the naming conventions are identical to that of the data flow construct.

5.6. Connectors

Semantics

A process model is a graphical representation of a process, and hence, comprises a series or a sequence of activity blocks and other constructs. There is a start and an end to a process and also to a sub process, when it is considered at its level in the hierarchy. When a process model is completed, the first and the last activity in each sub process will have no link to its previous and next activity, respectively. The purpose of a connector is to specify the link between the activities in such cases. In Figure 5.7 process number 3 is broken into its constituent sub processes 3.1 and 3.2. The connector before Activity 3.1 implies that the control flow to Activity 3.1 comes from Activity 2.1. Similarly, the connector after Activity 3.2 implies that after Activity 3.2 is executed, it transfers control to Activity 4.1 in sub process 4. Similarly, when an activity in a process is triggered by an activity from a different process, a connector could be used to model the flow of control.

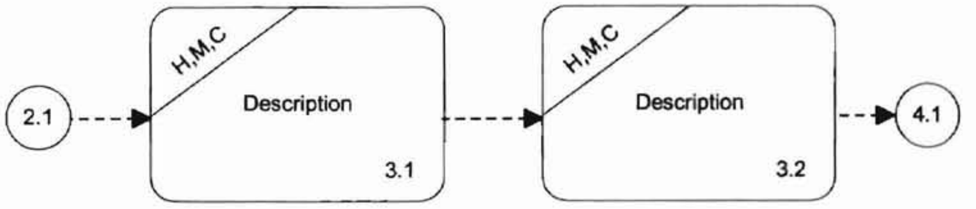


Figure 5.7: Use of Connectors

Notation

A circle with an activity number written in the center, as shown in Figure 5.8, represents a connector.

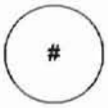


Figure 5.8: Connector

Control flow arcs are connected to the connector depending on whether it is on the input side or the output side of the activity block. However, only one control flow arc can be connected to a connector, to model either the incoming or the outgoing control flow. A connector with a "start" in the center instead of an activity number connected to an activity in a process model denotes the first activity of the process being modeled. Similarly, a connector with a "end" in the center instead of an activity number, connected to an activity in a process model, denotes the last activity of the process.

5.7. Time Trigger

Semantics

Sometimes, an activity can start only at a scheduled time or has to be repeatedly executed after a specific time interval. The timing requirement in such cases is represented by a time trigger as shown in Figure 5.9.

Notation

Time trigger is represented by the construct shown in Figure 5.9. The dashed arrow part of the construct indicates that the time trigger is a form of a control flow.

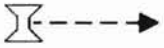


Figure 5.9: Time Trigger

When a time trigger is specified for an activity, it means that the activity cannot take place before the specified time instant. Even if all the preconditions for an activity are satisfied, the activity will not be executed until triggered by the time trigger. If the time trigger has already signaled the start of an activity and if the precondition(s) is/are not satisfied, the execution of the activity will be delayed until the precondition(s) is/are satisfied. This would indicate error in planning or delay in execution of certain activities, which could then be traced. In a software implementation, the time interval for the periodic start of the activity could be entered in a separate field upon clicking the time clock symbol.

5.8. Logical Operators

AND operator (one-to-many split)

Semantics

An activity can simultaneously trigger many activities. In this case, the signal is passed to all the triggered activities at the same time. For example, when an aircraft lands on an airport runway, many activities follow the landing. The baggage is unloaded, customers leave the aircraft and the aircraft is refueled for the next flight. This scenario is shown in Figure 5.10 to illustrate the use of the AND operator.

Notation

The activity "taxi aircraft to the gate" in Figure 5.10 triggers all the activities that follow it. The control flow goes to an AND operator. The AND operator splits the control flow, and signals the start of all of the activities indicated.

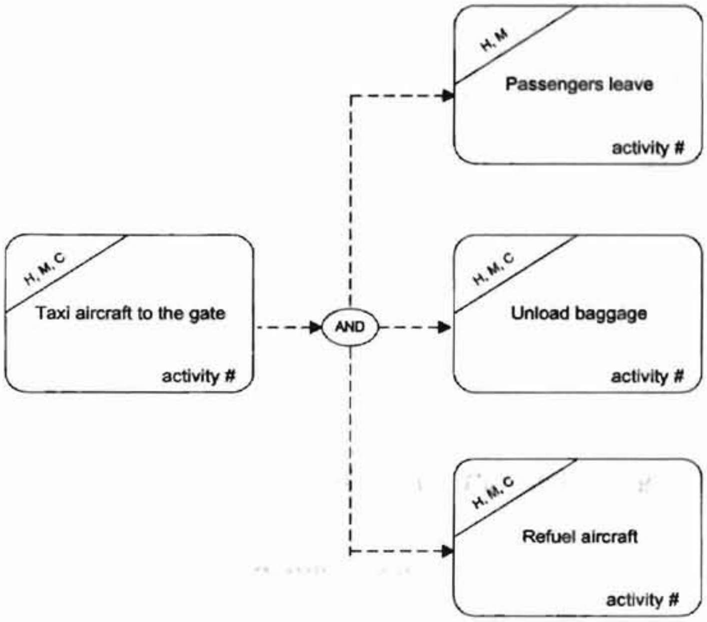


Figure 5.10: AND Operator – Parallel Split

AND operator (many-to-one merge)

Semantics

Many activities may have to be completed before an activity can be executed. For example, an aircraft can take off only when all the passengers have boarded the aircraft, the baggage has been loaded, and the aircraft is cleared for takeoff. This process is represented in Figure 5.11.

Notation

In Figure 5.11, the control flow goes from the three activities – board the aircraft, load baggage and clear aircraft for takeoff, to an AND operator. The AND operator acts like a union, and passes the control flow to the next activity only when the signal is received from all of the previous activities.

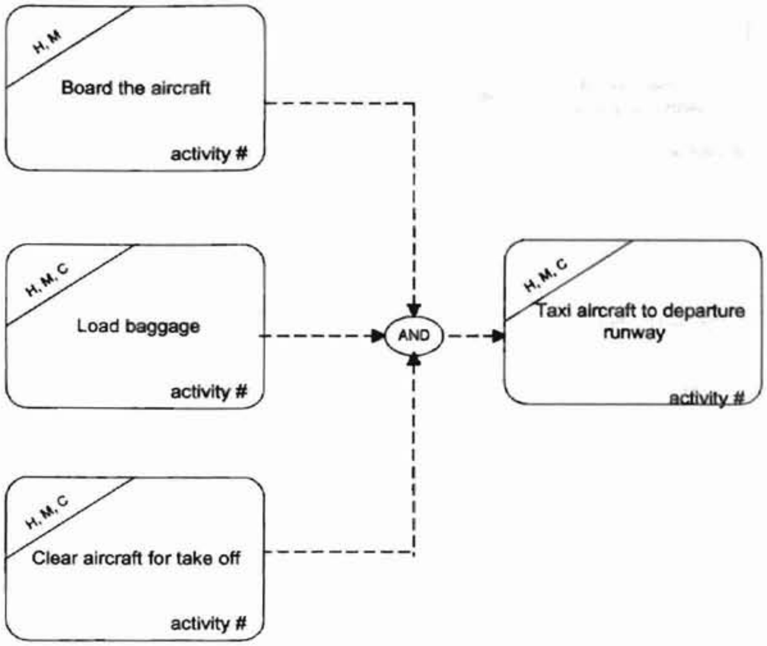


Figure 5.11: AND Operator – Parallel Merge

Exclusive OR operator (XOR) (from one to one-out-of-many)

Semantics

Consider an example of a component that is inspected after a manufacturing process. There can be three possibilities after inspection, the part is accepted for further processing, the part is rejected and scrapped, or the part is reworked and sent for further processing. Thus, one activity can lead to only one of many possible activities.

Notation

Refer to the example shown in Figure 5.12. The control flow arc from the activity "inspect part" leads to an XOR operator. The control flow splits into three arcs leading to the three activities "accept part and process further," "reject part and scrap," and "rework part and process further." However, the control flow from the activity "inspect part" can trigger only one out of the three possible, succeeding activities.

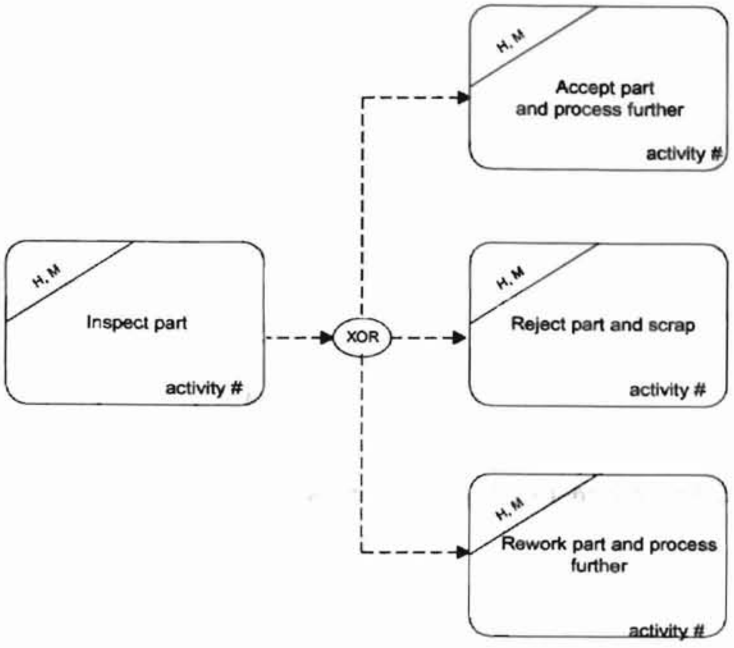


Figure 5.12: XOR Operator – (One to one-out-of-many)

Exclusive OR operator (XOR) (from one-out-of-many to one)

Semantics

Consider an example of a component which is to be processed to obtain a mirror finish and then checked for the value of the surface finish. A mirror finish can be obtained by either grinding the part or by buffing it. Thus out of the many possibilities (two in this case), only one can trigger the start of the next activity.

Notation

Refer to the example shown in Figure 5.13. The control flow arcs from two activities "grind part" and "buff part," converge at the XOR operator and then only one control flow arc connects to the activity "check surface finish." The XOR operator implies that out of the two activities – grind part and buff part, only one passes the control flow to the activity "check surface finish."

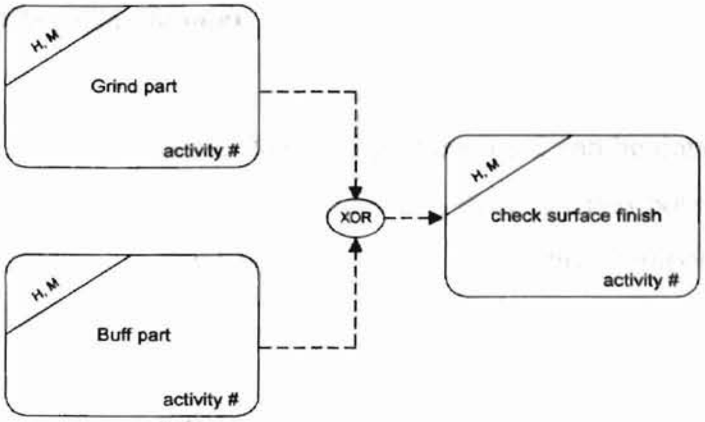


Figure 5.13: XOR Operator – (One-out-of-many to One)

Inclusive OR (IOR) (one to many)

Semantics

Consider a supplier payment process. When goods are received from the supplier, the buyer could pay the supplier by cheque, by cash, or part by cheque and part by cash. Thus one activity can lead to any combination of several activities.

Notation

Refer to Figure 5.14. The control flow arc from the activity "receive goods" leads to an IOR operator. The control flow splits into two arcs leading to the two activities - "pay by cash" and "pay by cheque." The control flow from the activity "receive goods" can trigger either both succeeding activities or only one of them.

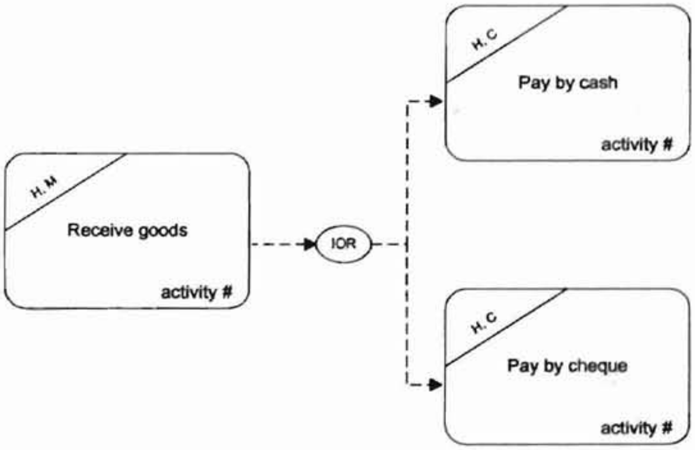


Figure 5.14: IOR Operator – One to Many

Inclusive OR (IOR) (many to one)

Semantics

Consider a manufacturing example. Machining operation 2 can be done on a component only after machining operation 1. For operation 1, there are two possibilities, it can be done in-house or it can be done by a subcontracter. When components which have completed operation 1 are available, operation 2 can be done on the components. It is possible that in a batch of 100 components which are ready for operation 2, 50 components might have completed operation 1 at the subcontractor, and the remaining 50 might have finished operation 1 in-house. It is also possible that all 100 components completed operation 1 at the subcontractor only or in house only. Thus, out of the many possible activities, there can be any combination of activities that can trigger the next activity.

Notation

Refer to Figure 5.15. The control flow arcs from the two activities "Subcontract operation 1" and "Do operation 1 in-house" converge at the IOR operator, and then only one control flow arc connects to the activity "Do operation 2." The IOR operator implies that out of the many possible preceding activities, any combination of the preceding activities can trigger the start of the next activity.

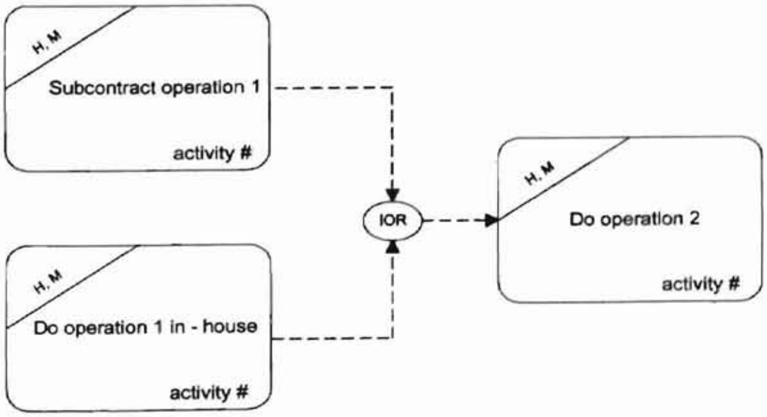


Figure 5.15: IOR Operator – Many to One

5.9. Binary Decision

Semantics

In process or data modeling, there are situations where a binary (yes/no) decision has to be made to determine the subsequent flow of control. Such cases could be modeled by using an XOR logical operator. An XOR logical operator gives the impression that the control flow can be optional, that is any one of the succeeding activities could be triggered. However, when a binary decision construct is used, it is clear that the resulting control flow is a result of a simple yes/no decision. For example, in Figure 5.16, after performing operation1, there are two possibilities, that is, the second operation can be done on either the preferred machine or the alternate machine. However, the selection is based on the outcome of a simple yes/no decision. If an XOR construct had been used instead, we would have shown two choices, but would not have been able to show that choice is determined by the outcome of a logical decision.

Notation

The construct for the binary decision is shown in Figure 5.16. It is represented by a diamond with the question written inside the diamond. There can be only one input control flow to the binary decision construct, and exactly two output flows.

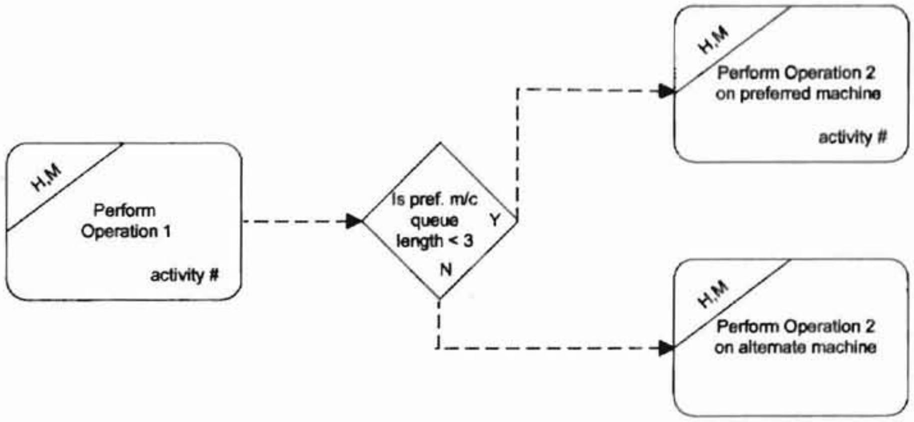

H,M
Perform Operation 1
activity #
Is pref. m/c queue length < 3
Y
N
H,M
Perform Operation 2 on preferred machine
activity #
H,M
Perform Operation 2 on alternate machine

Figure 5.16: An Example of a Binary Decision

5.10. Constraint, Rules, and Guidelines

Semantics

Sometimes it might be necessary that execution of an activity conform to a set of rules, guidelines, or constraints. For example, an assembly operation is constrained by the product structure specified in the bill of material. Figure 5.17 demonstrates the use of the constraint construct for capturing the constraints, rules or guidelines for an activity.

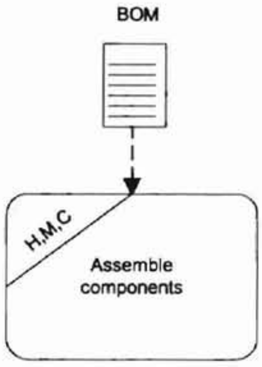


Figure 5.17: Constraint Construct

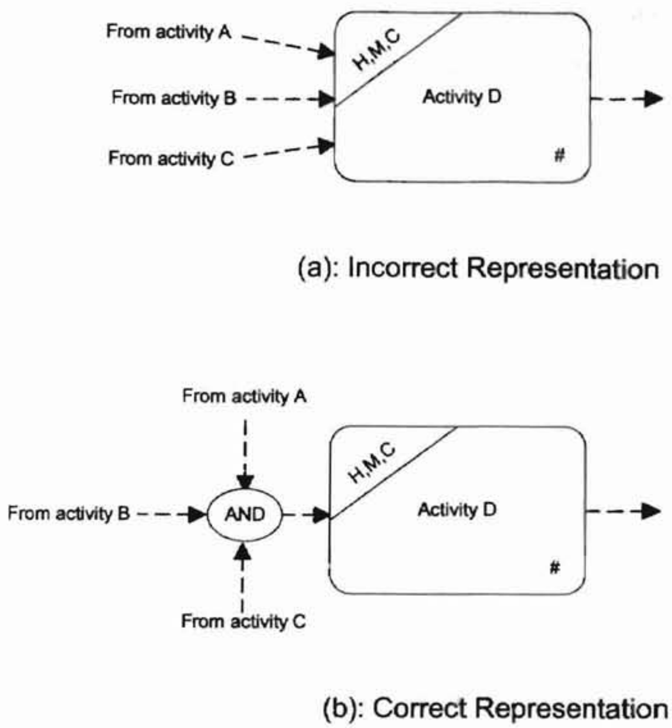
Notation

The constraint construct is represented by a rectangle with horizontal lines in it as shown in Figure 5.17. It is connected to the activity construct with a control flow arc. A constraint serves like a rule or information necessary for the activity to be executed. A constraint is not transformed like the input-output transformation after the execution of the activity. Hence, a control flow arc is used to connect the constraint construct to an activity construct. The implication of the constraint construct is that the activity's execution must adhere to the rules or guidelines specified.

5.11. Rules for Representing Control Flow to an Activity

In EPML, only one control flow arc can be directly connected to an activity construct. The following two cases which use variants of the control flow are not affected by this rule. They are (i) when an activity requires a time trigger, which is represented by the construct shown in Figure 5.9, and (ii) when the execution of an activity has to take place with respect to guidelines or constraints, which is represented by the construct shown in Figure 5.17.

Consider the example shown in Figure 5.18.



(a): Incorrect Representation

(b): Correct Representation

Figure 5.18: Rules for Representing Control Flow

Suppose that activity D in Figure 5.18 can be executed only after activities A, B, and C, have occurred. Figure 5.18 (a) shows an incorrect representation of the scenario. It is not clear from the representation whether all of the activities A, B, and C have to take place, or a combination of the activities has to take place to trigger activity D. This ambiguity is avoided by correctly representing the scenario as in Figure 5.18 (b), by using the logical construct "AND" and connecting the resultant control flow to activity D.

5.12. Feedback Representation

In this section, the use of EPML constructs for capturing feedback will be demonstrated. This is a unique feature of EPML. Consider an example of new product development process. There are four main sub processes, namely, 1) product design, 2) prototype building, 3) testing, and 4) manufacturing. Each sub process can be decomposed into many activities. Consider the situation where a design activity needs feedback from a testing activity. Note that feedback is required from an activity in sub process 3, that is testing, to an activity in sub process 1, that is design. Let us arbitrarily number the concerned activity in sub process 1 as activity 1.4.5, and the concerned activity in sub process 3 as activity 3.2.6. The three stages – design, prototyping, and testing are iterative. However, when the initial design is in progress, feedback cannot take place because the prototype is not yet developed and is yet to be tested. Figure 5.19 shows how feedback can be represented in this case.

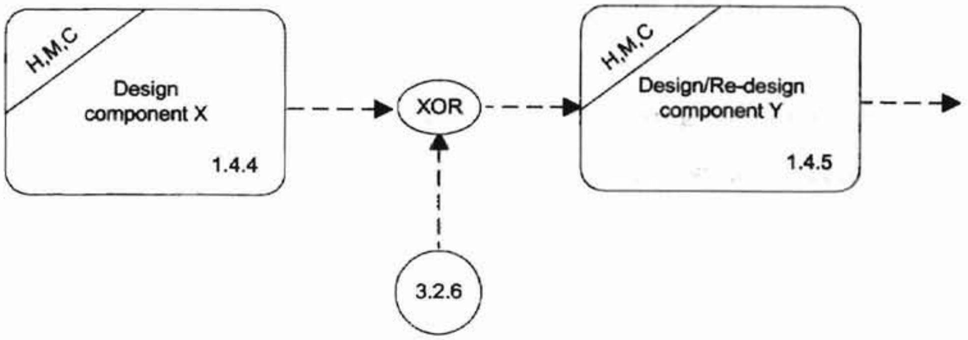


Figure 5.19: Feedback Representation

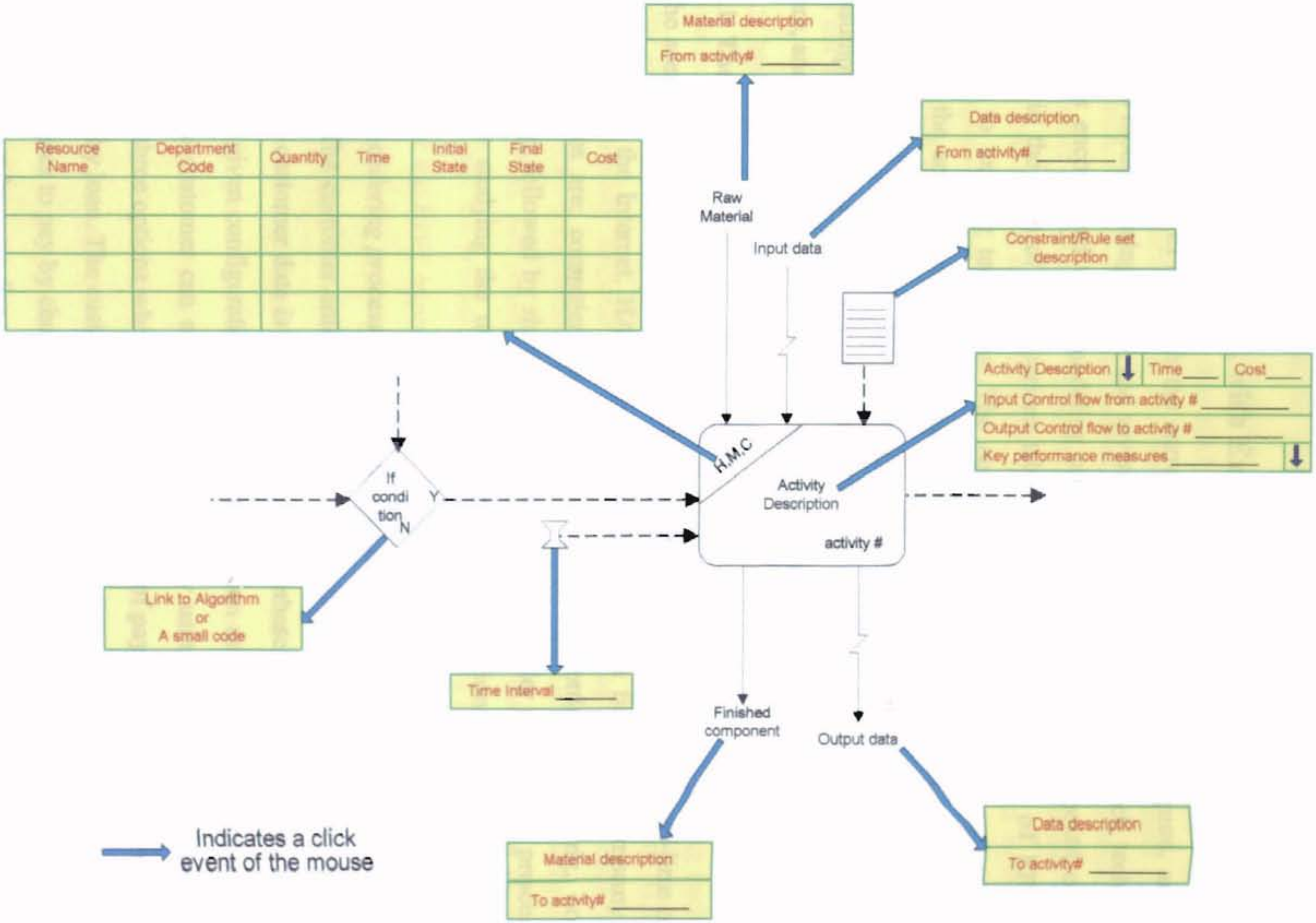
Activity 1.4.5 can be triggered by the completion of activity 1.4.4, in the case of a new design, or by activity 3.2.6, in the case of an existing design.

5.13. Data Flows and Logical Constructs

Logical junctions for data flows can be represented in a manner similar to that of control flows. However, it could clutter the graphical representation and hurt readability. It is left to the user whether to model junctions for data flow or not, because the focus of EPML is on modeling control flows. Furthermore, such details could be easily captured in a software implementation.

5.14. EPML Implementation within a Software Environment

As mentioned earlier in Section 1.4, this research is a part of a NSF funded effort focused on creating a user-oriented framework for process modeling. A preliminary design of a graphical front-end implementation that would let the user enter detailed information is shown in Figure 5.20. Capturing process details that supplement EPML constructs is simplified by providing various user interface constructs. Thus, when EPML is implemented in software, the language constructs along with the user interface constructs will enable a user to capture most of the details of a process. The data provided by the user will most likely be stored in an XML representation to provide interoperability and platform independence in a Web-based environment (Kamath *et al* 2001). In Figure 5.20, resource details can be captured in a tabular form that will pop up when the user clicks on a particular resource set. Similarly, after clicking on "input data," the user will be prompted to enter data description and the activity number (if applicable) from where the data originated. Clicking on "output data" will prompt the user for the description of output data and (if applicable) the activity number to which the output data will serve as an input. Thus, origins of data flows can be captured using the graphical interface. Time and cost associated with the activity can be entered in a similar manner as part of the activity description. Binary decision constructs will be associated with an algorithm or logic to choose the resultant control flow. When the user clicks on a binary decision construct he/she will be prompted to provide link to an algorithm or scripting code.



Resource Name	Department Code	Quantity	Time	Initial State	Final State	Cost

Figure 5.20: Software Implementation Details of EPML Constructs

6. Comparison of EPML with Existing Process Modeling Techniques

In order to compare EPML with the current process modeling techniques, an example scenario is modeled using EPML and several existing techniques. For each technique, the limitations encountered while modeling are explained from a user's point of view. A summary in the form of a comparison table highlights the advantages of EPML when compared to existing techniques. The constructs, semantics and the syntax used for modeling the scenario under consideration, are based on information available in the literature and Web sources. When specific information was not available, reasonable assumptions were made to complete the modeling exercise. However, such cases were rare, and are clearly identified in the ensuing sections.

6.1. The Scenario

The scenario that is modeled for a qualitative evaluation of EPML is an on-line ordering system using the Internet. HAL Computer Corporation is a fictitious company whose main processes are: computer selection by the customer, order processing, computer manufacturing, followed by shipping of the product. The order processing process is an ideal case for studying the usability, strengths and limitations of different modeling techniques.

The computer ordering process starts after the product is checked out by the customer via the Internet. The customer enters personal details in the forms provided on the browser interface. The customer data is stored on the HAL database server. Programs to calculate the price of a given configuration of a computer are run using the data submitted by the customer. The customer can see the price details by using the browser interface. The customer has three options while selecting the mode of payment, namely, by cheque, by credit card or by loan. The customer can choose only one option out of these three. If the customer chooses to pay by cheque or by credit card, the cheque or the credit card details are entered in a browser form, and the payment takes place via Internet. Once HAL receives the payment, an internal production order is sent to the manufacturing department. If the customer opts for a loan, then some additional steps have to be completed before the order can be placed. The customer has to complete a loan application on the Internet and submit it to HAL. HAL then processes the loan request. If the customer is a past customer, then HAL checks for their installment history and credit balance. If HAL is satisfied with the customer records, and the customer has enough credit balance, the loan request is accepted, and HAL places an internal production order. If the customer records are not "satisfactory," it is left to the bank to decide whether to accept the loan request or not. Similarly, if the loan request is made by a new customer, the bank makes the final decision. If the loan is accepted, HAL receives the loan payment from the bank and places an internal production order. If the loan is rejected, the bank informs HAL. Then the customer has the choice to pay either by credit card or by cheque, or not to buy the computer.

6.2. Modeling using IDEF Techniques

Figures 6.1 to 6.4, show the ordering process using IDEF process modeling techniques. IDEF has a family of languages, namely IDEF0, IDEF1X, IDEF2, IDEF3, etc. IDEF0 is a functional modeling language and IDEF3 is a process description capture method. IDEF0 and IDEF3 complement each other in the sense that IDEF0 captures input output data, resource data, and constraint data, whereas IDEF3 captures the control flow. Thus, in order to capture the process details, two different techniques have to be used, which is a drawback of the IDEF approach. If an IDEF0 diagram is treated in isolation, logical flows like conjunction and disjunction cannot be shown. In Figure 6.1, the output "cost details on customer's browser" from activity 2.2, leads to three activities – "process cheque", "process loan" and "process credit card," and it is not clear that only one activity can take place. In IDEF0, a page can contain only a maximum of 6 activities. Hence, if a process contains more than 6 activities, some of the activities have to be grouped together as a sub-process. For example in Figure 6.1, the activity "process loan" is broken down into sub activities and represented in Figure 6.2. In IDEF0, an output from the previous activity seems to trigger the next activity. However, in case of an event

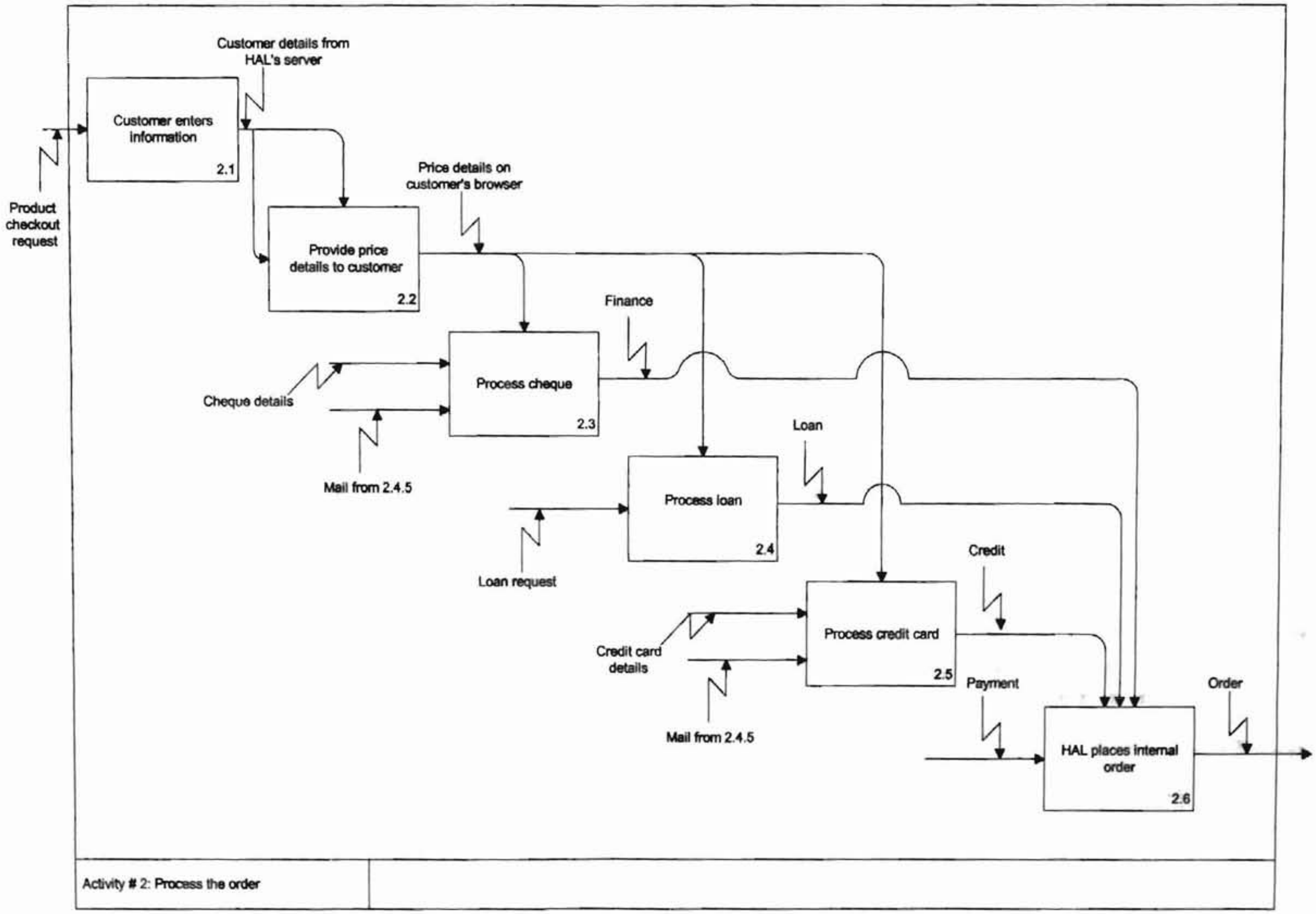


Figure 6.1: IDEF0 Representation of HAL's Ordering Process

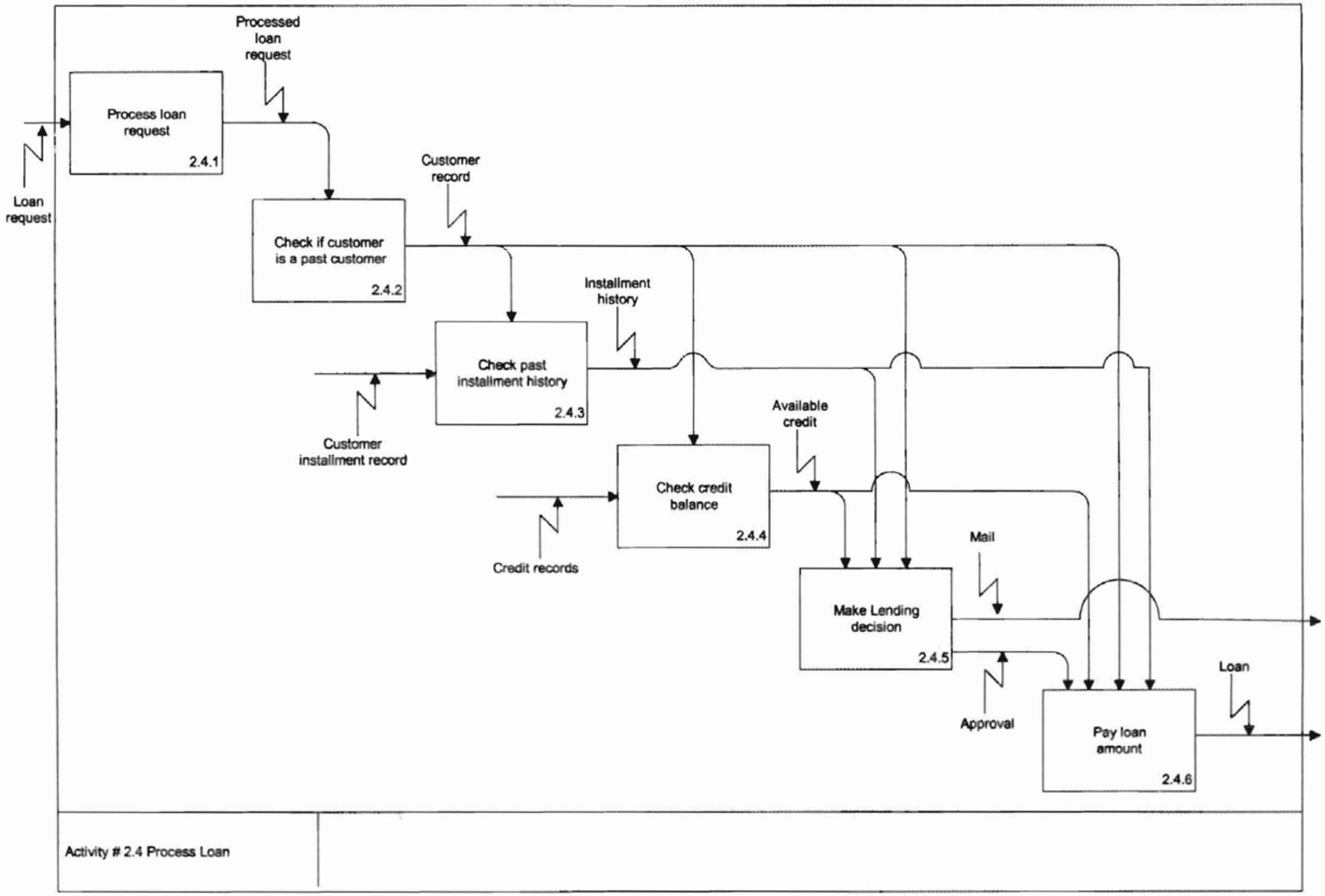


Figure 6.2: IDEF0 Representation of HAL's Ordering Process, - Sub process # 2.4

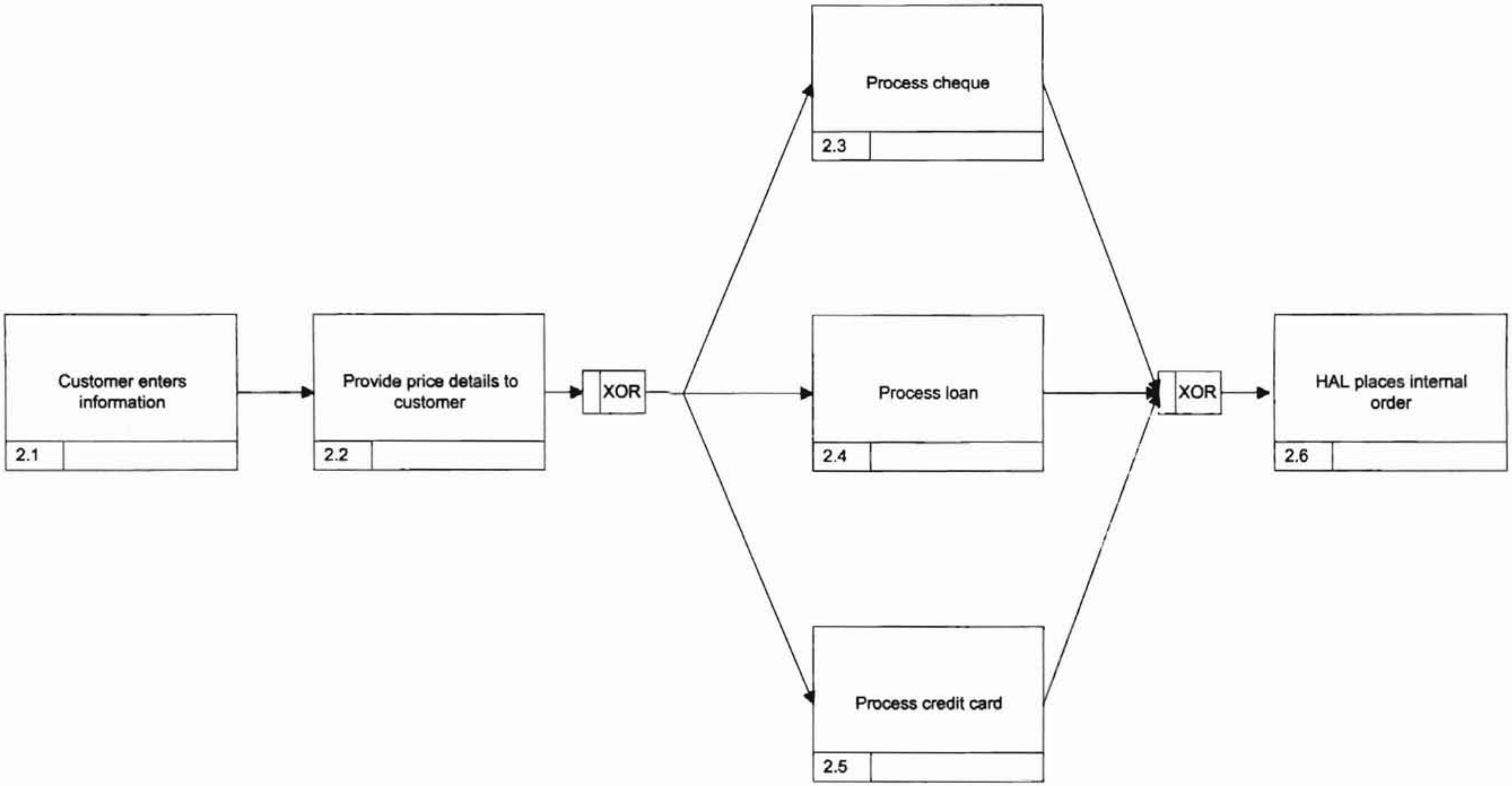

Customer enters information
2.1
Provide price details to customer
2.2
XOR
Process cheque
2.3
Process loan
2.4
Process credit card
2.5
XOR
HAL places internal order
2.6

Figure 6.3: IDEF3 Representation of HAL's Ordering Process

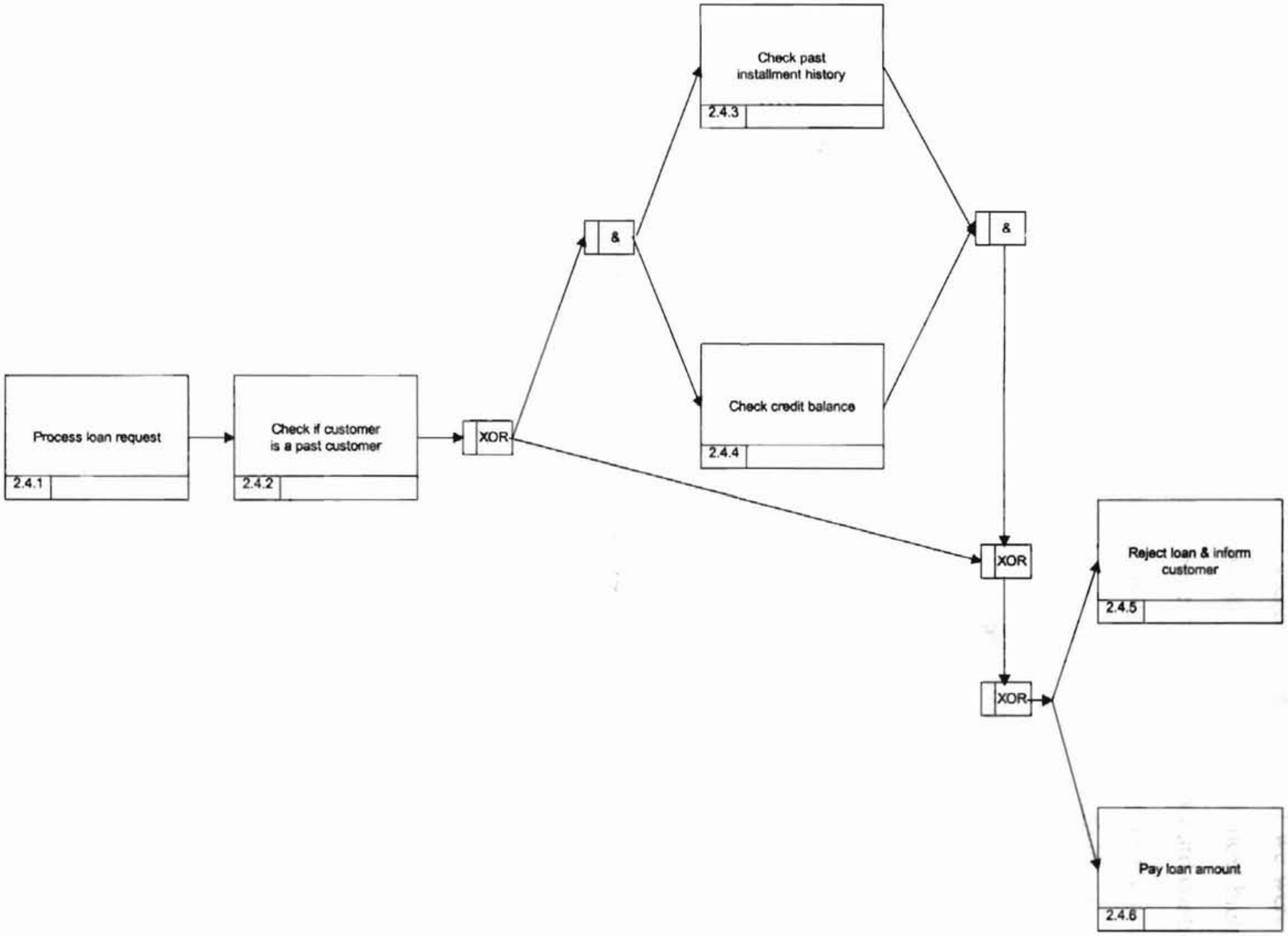


Figure 6.4: IDEF3 Representation of HAL's Ordering Process - Sub process # 2.4

like "customer decides to buy or not" which would be modeled using a binary decision construct in EPML, we cannot show the yes/no split in an unambiguous manner from an activity in the IDEF0 diagram. Morever, in the scenario considered, if we were able to model this event in IDEF0, it would be in a separate figure and at the next hierarchical level. It is interesting to note that IDEF3 could have modeled the backward control flow after the decision to buy the computer or not is taken by the customer in the event of a loan rejection. However, because of the limitation of the number of activities in a single IDEF0 diagram, the process had to be broken down into two levels. Thus, syntactic limitations limit the "naturality" in modeling. Although tunneled arrows can model data flows between hierarchical levels of processes, the semantics of tunneling are difficult to understand (www.idef.com).

6.3. Modeling using CIMOSA

The first step in CIMOSA is to decide the level at which the process is to be modeled. That is, whether it is a domain process, or a business process, or an enterprise activity, or a functional operation (Kosanke 1995). The scenario under consideration is best modeled at the level of enterprise activities. Referring to Figure 6.5, it is seen that CIMOSA does model control flow, but not as clearly as in EPML, IDEF3 , or SAP's EPC method. There are no logical operators and hence, directed arcs lead to all possible activities even if only one can take place. This leads to many intersecting arcs on the diagram. Additional information has to be presented in a textual format with clear logic like in an algorithm at points where control flows split or merge. Each activity should be clearly defined in the textual representation with its inputs, outputs, resources, etc. Each logical split has to be clearly explained with if-then, and/or statements. Without the textual representation, the model would be incomplete. The focus of the CIMOSA modeling language is to obtain a clear algorithmic description of the process, rather than a graphical description.

6.4. Modeling using IEM

The IEM model of HAL shown in Figures 6.6 and 6.7 is based on the information available in Mertins *et al.* 1992. The IEM graphical model has three levels of description,

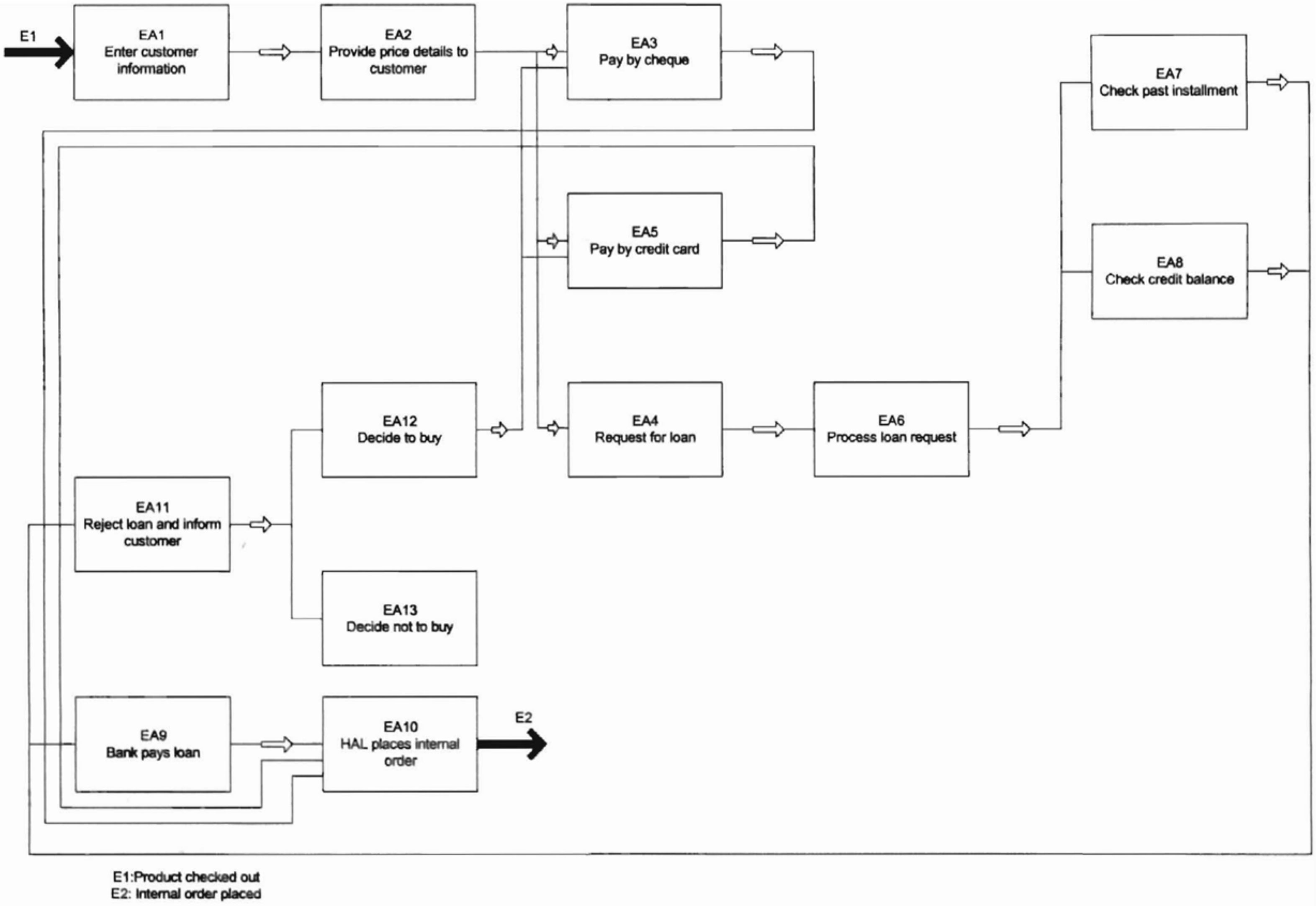


Figure 6.5: CIMOSA Representation of HAL's Ordering Process

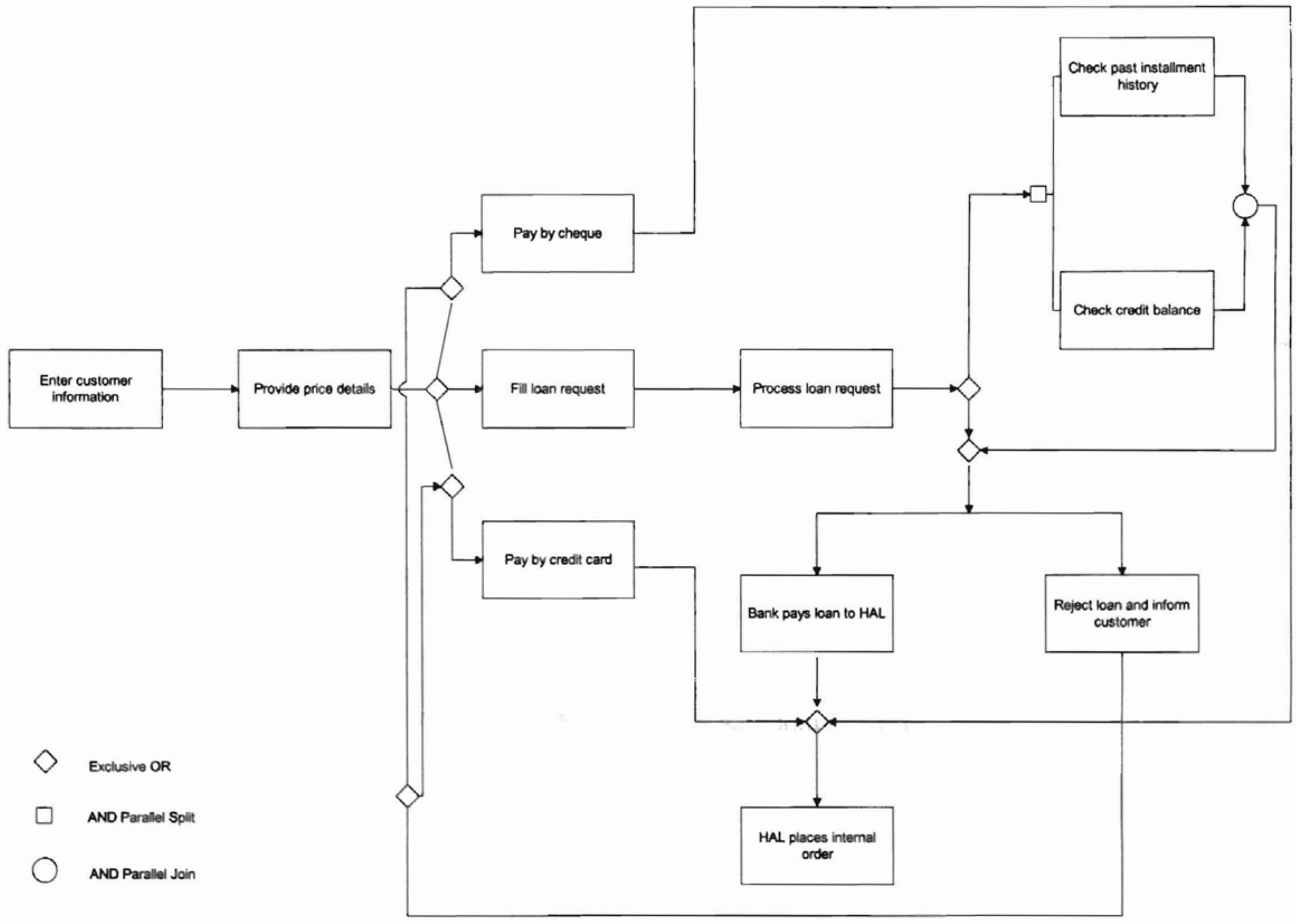


Figure 6.6: IEM Function Chain for Hal's Ordering Process

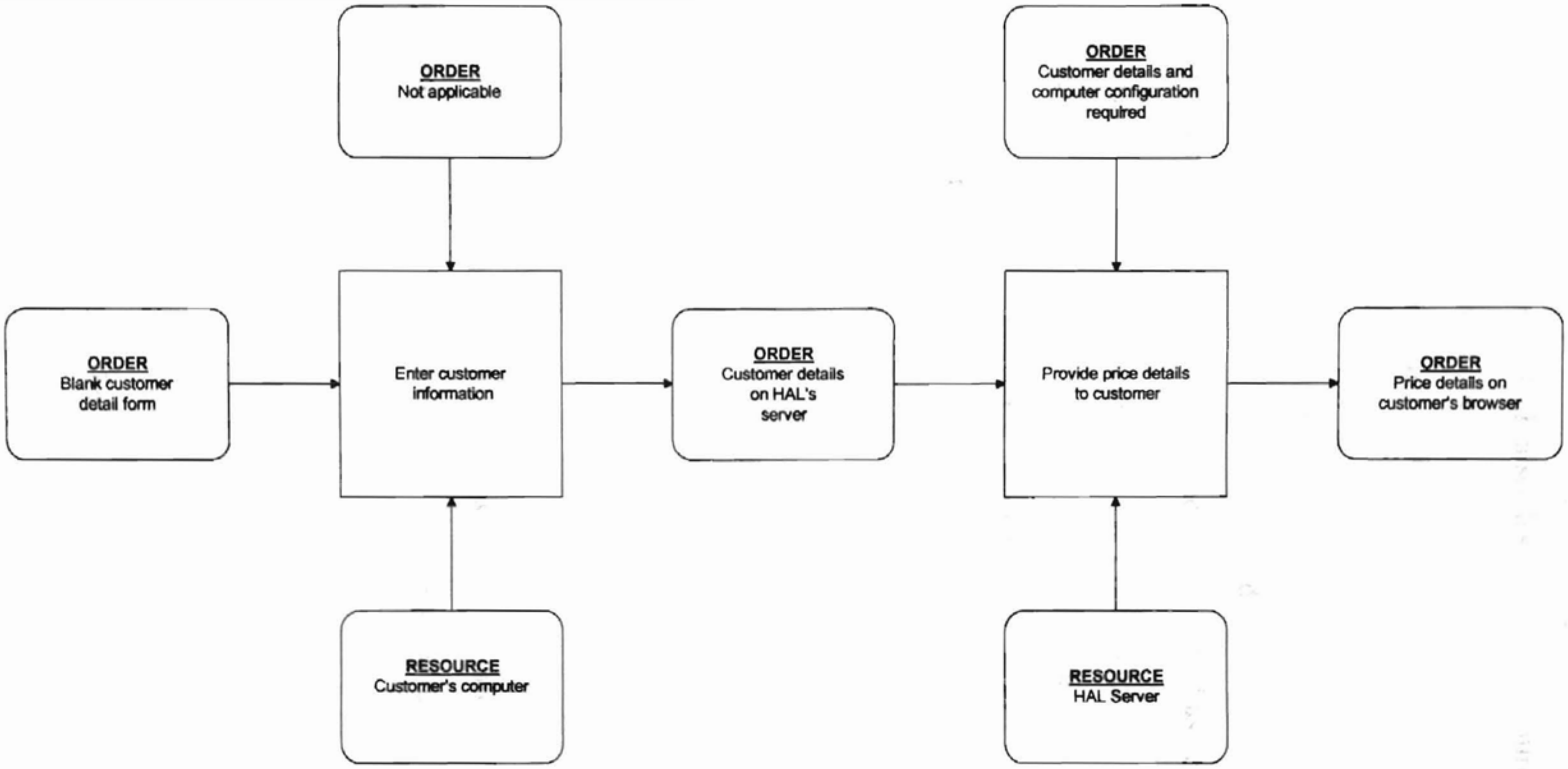

ORDER
Not applicable
ORDER
Customer details and computer configuration required
ORDER
Blank customer detail form
Enter customer information
ORDER
Customer details on HAL's server
Provide price details to customer
ORDER
Price details on customer's browser
RESOURCE
Customer's computer
RESOURCE
HAL Server

Figure 6.7: IEM Partially Autonomous Unit

namely the partial model which represents the sequence of tasks, the functional partial model which includes inputs and outputs along with the sequence of activities, and the partially autonomous unit which also describes the resources and the control triggers for all activities in addition to the description in the functional partial model. Each input, output, resource, or constraint, is classified under three classes, namely order, product and resource. Figure 6.6 shows the function chain of HAL's ordering process and Figure 6.7 shows the complete activity model for only two sample activities. The Figure 6.7 indicates that modeling HAL's complete ordering process would result in several pages of diagrams. Hence, modeling is not compact in IEM as compared to EPML.

The symbols for logical operators are not clearly defined in Mertins *et al.* (1992), and hence, in the example shown, their assumed meanings are stated. Each element of modeling needs to be classified as objects or as sub-classes of the three main super classes. This requires a clear grasp of the object-oriented concepts on the user's part.

6.5. Modeling using TOVE

The graphical model of HAL's ordering process using TOVE modeling constructs and concepts is shown in Figure 6.8 and 6.9. Figure 6.8 represents the activity abstraction diagram and Figure 6.9 represents the activity state cluster for only the first two activities. The purpose of the former diagram is to model all activities and show the control flow. For a complete view of the process, the activity-state cluster diagram in Figure 6.9 has to be referred. It is possible to represent all elements of the process in one diagram, that is, the activity-state cluster diagram. However, the graphical representation required for the representation of the complete information, like resources, etc. is extensive. In EPML, the activity block has three elements, H, M and C, which represent human, machine and computer resources, respectively. The details of the resources are not represented graphically in EPML; however, in TOVE the resources are represented graphically, but at the expense of the extensive graphical representation. The information on resource states in TOVE can be extended to EPML, but instead of cluttering the process model diagram, it would be better to model the resource state concepts as attributes.

TOVE treats all objects, namely the inputs, outputs, constraints, etc. as resources. The activity abstraction diagram (Figure 6.8) does not have symbols for logical operators to

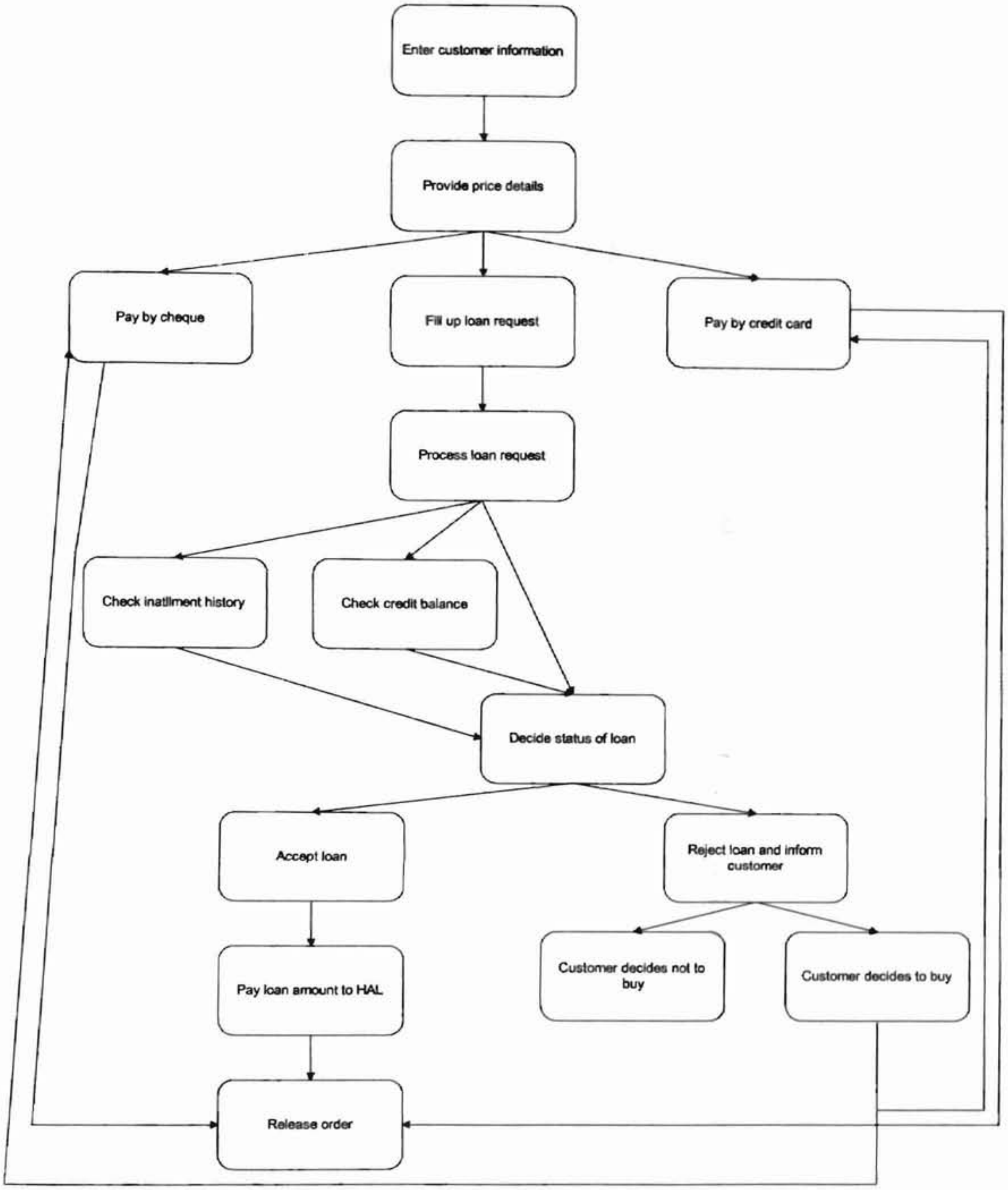


Figure 6.8: TOVE – Activity Abstraction Diagram – HAL's Ordering Process

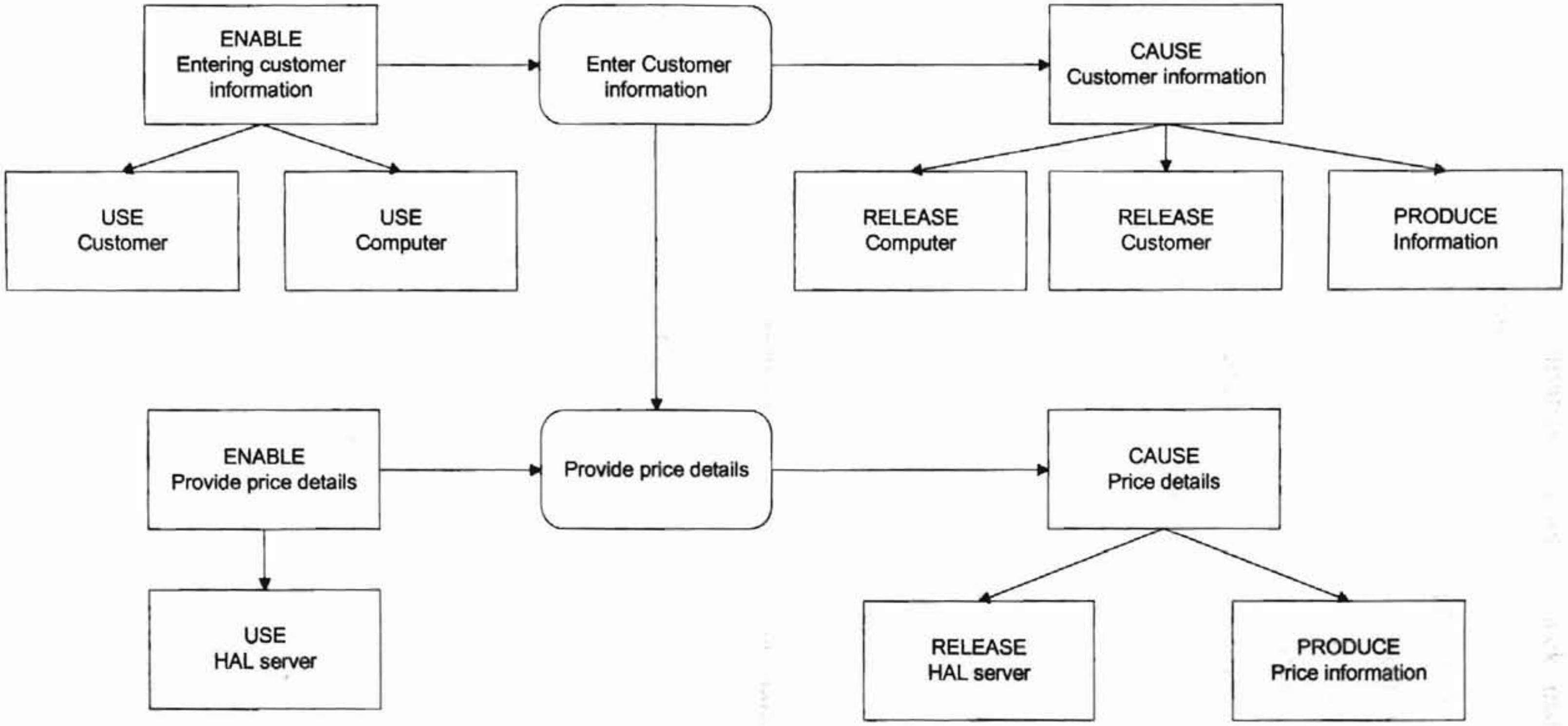


Figure 6.9: TOVE – Activity Cluster Diagram

model the logical control flow. Referring to Figure 6.8, directed arcs lead from "process loan request," to three activities, namely "check installment history," "check credit balance" and "decide status of loan." It is not clear whether all activities take place or only one of them takes place after processing the loan request. This also leads to many intersecting arcs on the diagram. TOVE has logical or Boolean representation for the resource states, but not for the control flow. A set of axioms has to be developed in first order logic to define the control flow. Within EPML, a binary decision construct can conveniently represent a decision like "loan approval" as in Figure 6.15. In TOVE, an additional activity "decide status of loan" has to be defined for clarity.

6.6. Modeling using SAP's EPC

SAP's event driven process chain (EPC) model of HAL's ordering process is shown in Figures 6.10 and 6.11. There is no differentiation between physical data and electronic data. The description tends to be in a vertical format which limits flexibility in modeling. Each activity block is preceded and succeeded by an event block which acts as a trigger. In EPML, the control flow arcs represent a trigger, and pass control on to the succeeding activities. SAP's description of the HAL example requires almost twice the amount of space than that required by an equivalent EPML description because of the representation of events by a separate construct. Thus, EPML has a more compact representation for the same amount of information as compared to SAP's EPC method.

6.7. Modeling using Data Flow Diagram

Data flow diagram (DFD) tends to separate departments/entities as shown in Figure 6.12. The customer and the bank are modeled as external entities. The customer is an integral part of the modeling process. Process 2.1 cannot be modeled unless some external entity initiates it, as per syntactic rules in DFD. Hence, customer is modeled as an external entity. There is a symbol for representing a database, but there cannot be any direct interaction between a database and an external entity in a DFD. All such interactions have to take place through a process. If the loan is rejected, the customer has to decide if he wants to buy the computer by credit card, or a cheque or not to buy. This decision process cannot be shown in Figure 6.12 because no data flow takes place in the decision

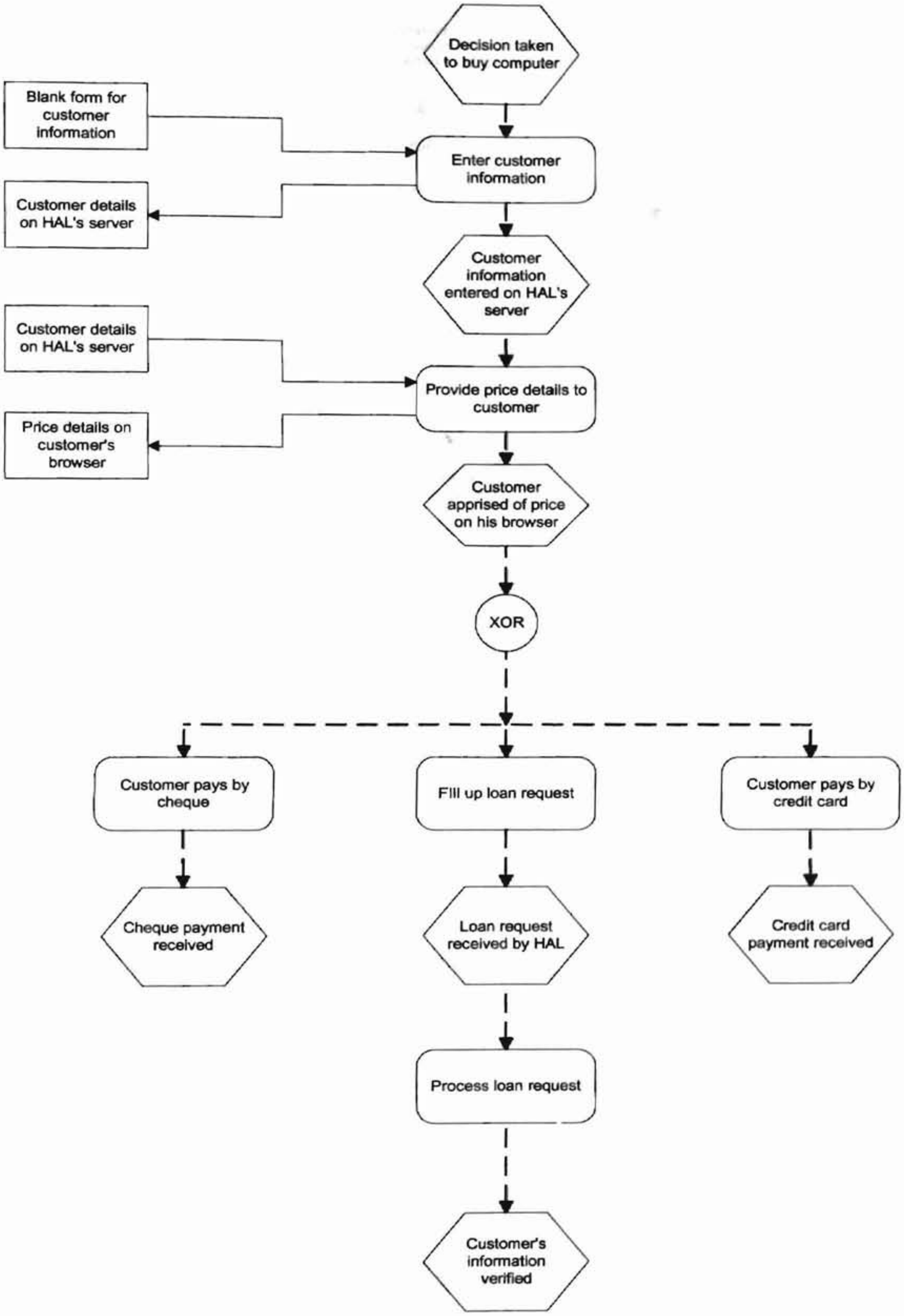

Decision taken to buy computer
Blank form for customer information
Enter customer information
Customer details on HAL's server
Customer information entered on HAL's server
Customer details on HAL's server
Provide price details to customer
Price details on customer's browser
Customer apprised of price on his browser
XOR
Customer pays by cheque
FIll up loan request
Customer pays by credit card
Cheque payment received
Loan request received by HAL
Credit card payment received
Process loan request
Customer's information verified

Figure 6.10: SAP's EPC Representation of HAL's Ordering Process

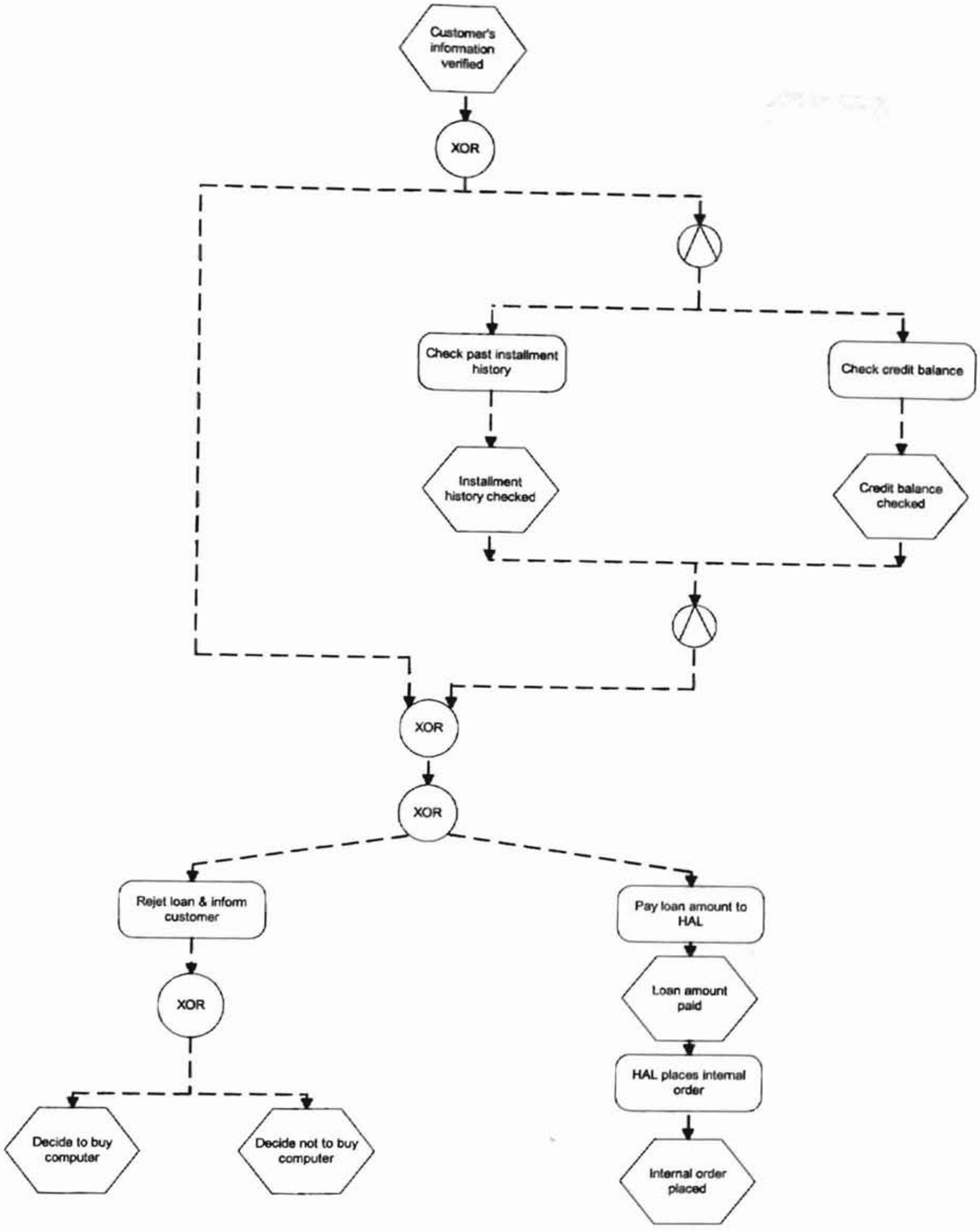

Customer's information verified
XOR
∧
Check past installment history
Check credit balance
Installment history checked
Credit balance checked
∧
XOR
XOR
Rejet loan & inform customer
Pay loan amount to HAL
XOR
Loan amount paid
HAL places internal order
Decide to buy computer
Decide not to buy computer
Internal order placed

Figure 6.11: SAP's EPC Representation of HAL's Ordering Process, (contd.)

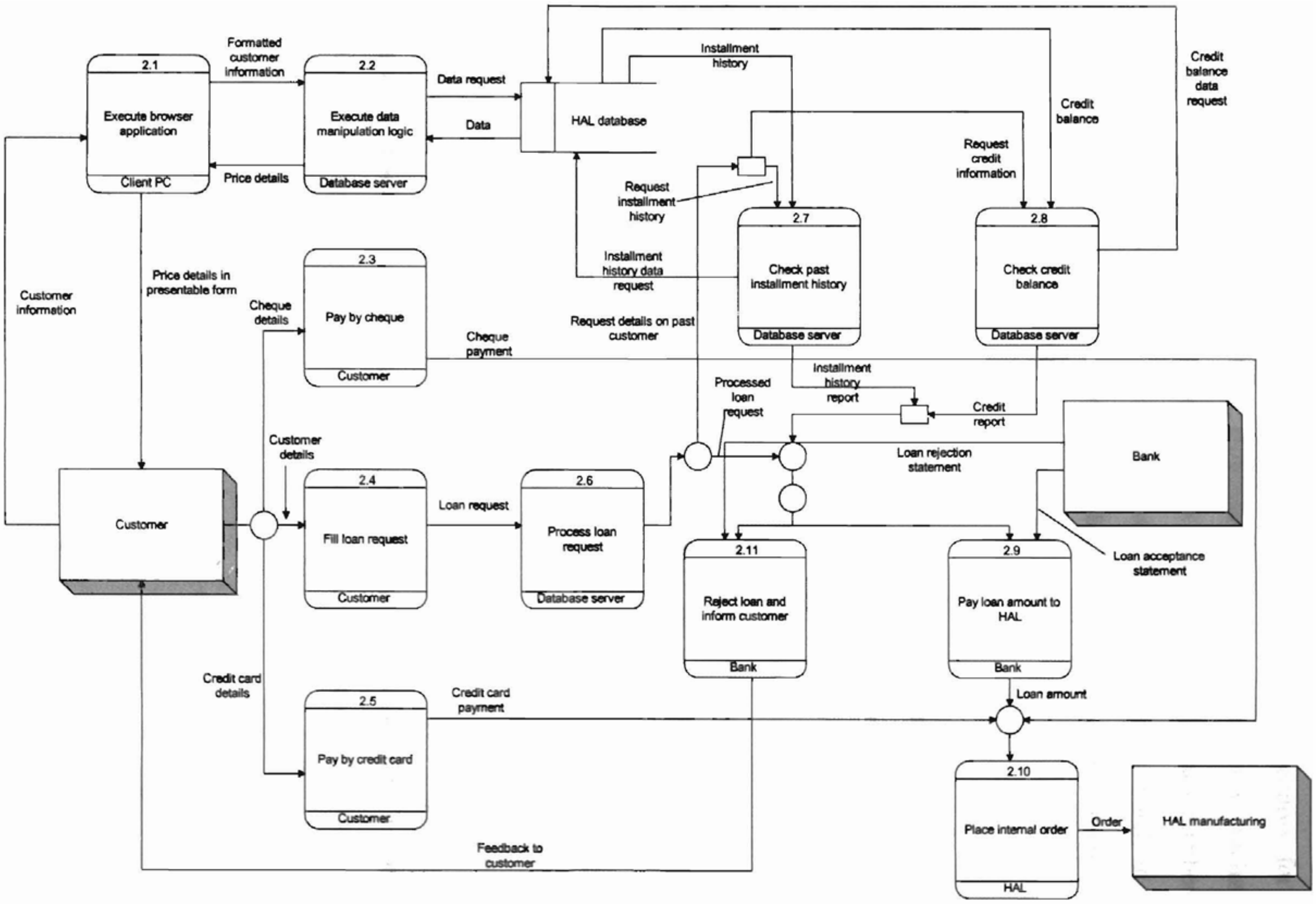


Figure 6.12: DFD Representation of HAL's Ordering Process

process, and also because the customer is an external entity. In DFD modeling there has to be an input and an output from each process. Control flow and data flow tend to get mixed with each other as in IDEF0. There are symbols for logical operators, for "parallel split and join" and an "exclusive OR". However there is no symbol for an 'inclusive OR.' When many data flows connect to one process block, it means that any number of them can serve as inputs or outputs to the process, which is the manner in which an inclusive OR situation is modeled. This clutters the graphical representation. DFD modeling seems to have a focus on data flow rather than on control flow.

6.8. Modeling using BAAN's DEM

Dynamic Enterprise Modeling (DEM) Method is similar to a Petri net representation of a process. DEM models are shown in Figures 6.13 and 6.14. A Petri net is an excellent tool for process analysis, however it is not very user friendly. Every activity in the DEM is preceded by states or places as in a Petri net. This is similar to an event in SAP's R/3 reference model, or the directed arc in EPML. The representation of states lengthens the diagram like the SAP's EPC model. Morever, there are no separate constructs for the different types of logical operators. Each logical junction is represented by a square with its description on the side. Control flow is depicted in the model, however other details such as the inputs, outputs, resources, etc. are not represented in the graphical model.

DEM's guidelines mention that it is better to structure a model in such a way that process flow is limited to only 5 to 10 activities (van der Rijst 1997). If more steps are required a sub process should be constructed. This leads to the same modeling limitations as in IDEF0. For example in Figure 6.13, the activity "pay loan amount" is broken into sub activities and represented in Figure 6.14. If the loan is rejected, the customer has to decide whether to buy the computer or not. If the customer decides to pay for the computer by either credit card or by cheque, the flow of control cannot be shown because paying by credit card and paying by cheque are shown at the higher level in the hierarchy, in Figure 6.13.

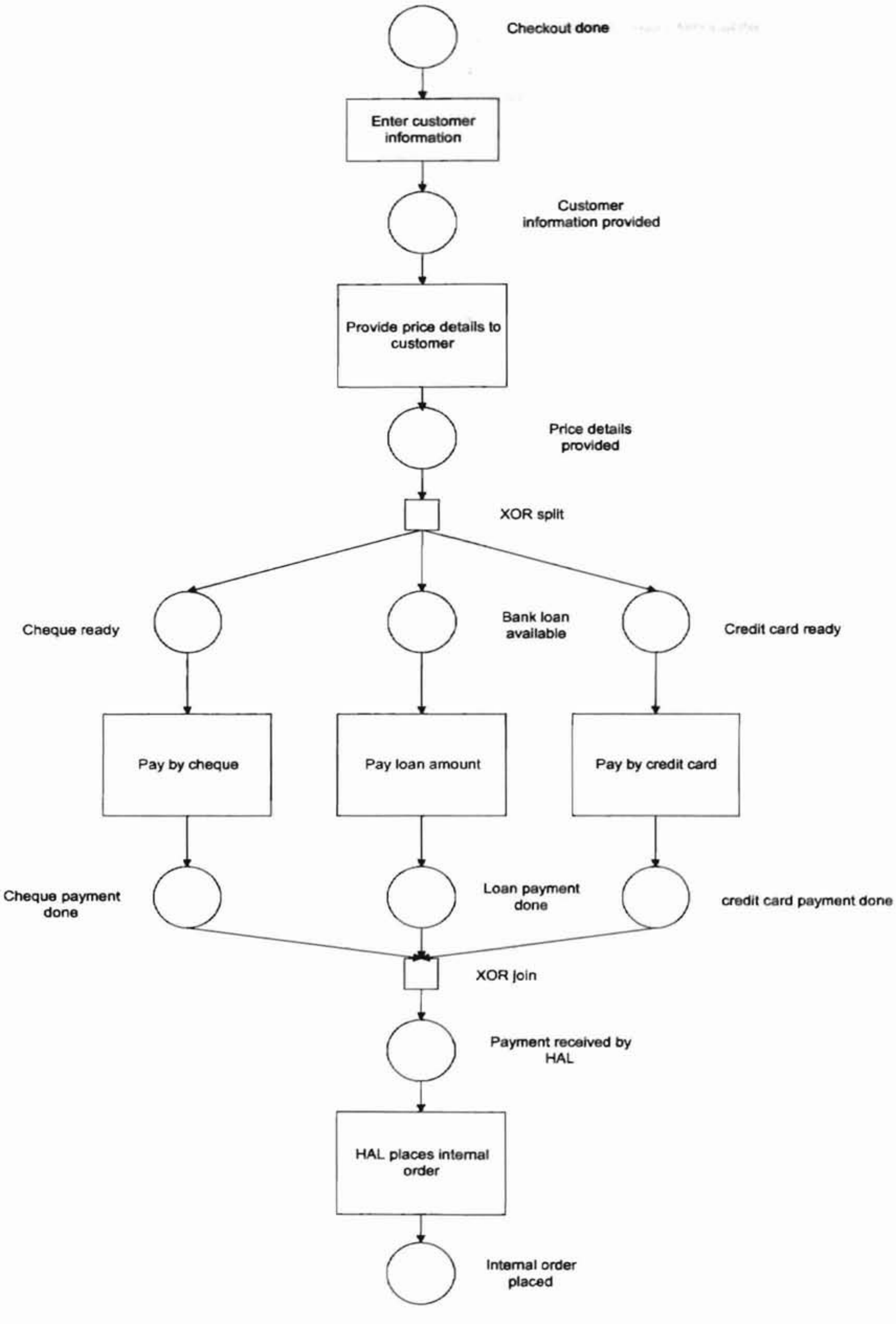


Figure 6.13: Baan's DEM Representation of HAL's Ordering Process

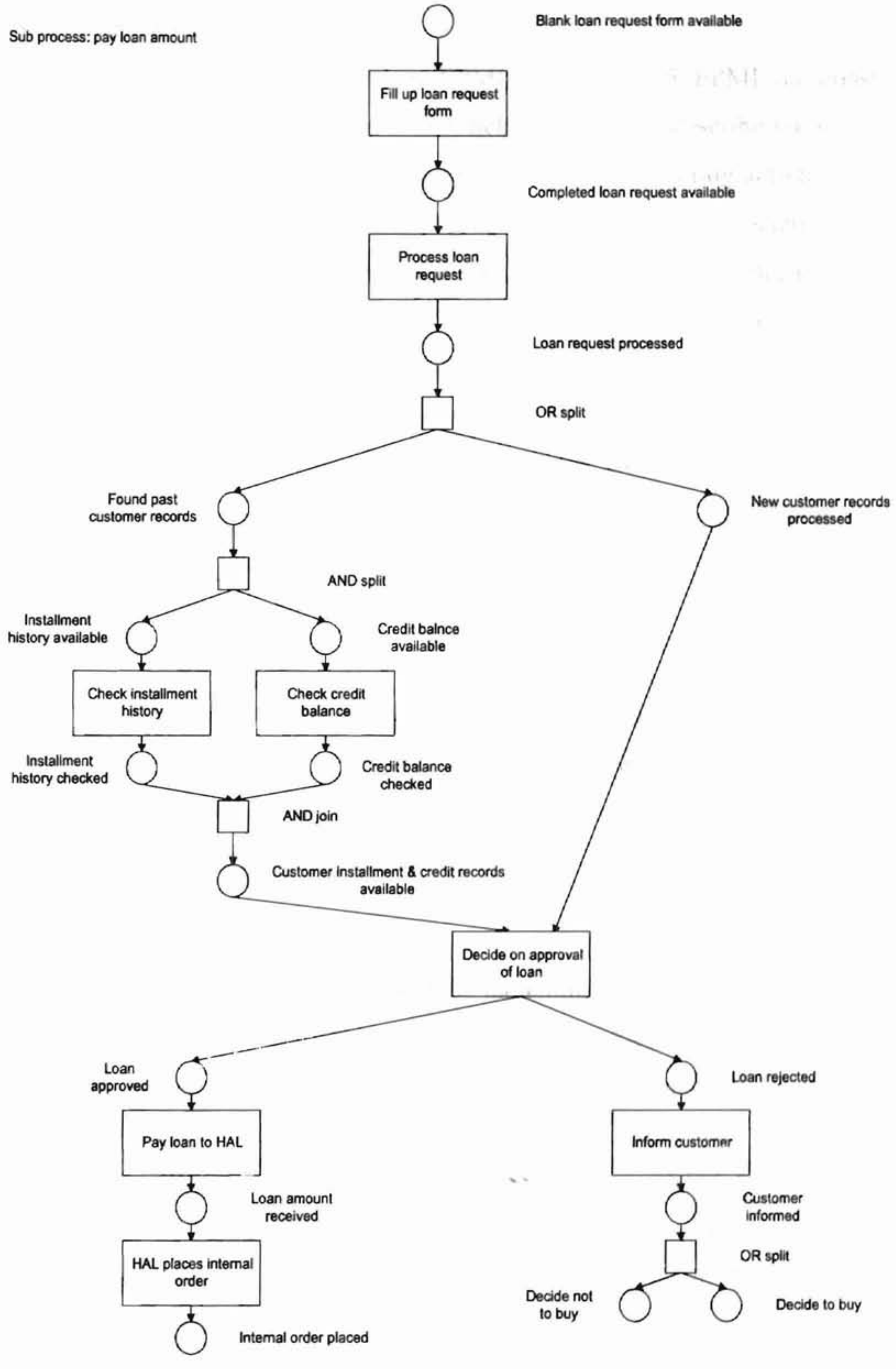

Sub process: pay loan amount
Blank loan request form available
Fill up loan request form
Completed loan request available
Process loan request
Loan request processed
OR split
Found past customer records
New customer records processed
AND split
Installment history available
Credit balnce available
Check installment history
Check credit balance
Installment history checked
Credit balance checked
AND join
Customer installment & credit records available
Decide on approval of loan
Loan approved
Loan rejected
Pay loan to HAL
Inform customer
Loan amount received
Customer informed
HAL places internal order
OR split
Decide not to buy
Decide to buy
Internal order placed

Figure 6.14: Baan's DEM Representation – Sub Process-HAL's Loan Payment System

6.9. Modeling using EPML

HAL's ordering process is represented using EPML in Figure 6.15. EPML has constructs with well defined semantics and syntax which help the modeler describe the process as seen in reality. There is a strong control flow represented by modeling activities as they occur in sequence. A dashed directed arc signifies that the event associated with the completion of the preceding activity has occurred. There is no need to represent an event by using a separate event construct as in SAP's EPC diagram, or like a place in the DEM. This makes the EPML representation very compact. The modeling of a binary decision by a separate construct enhances the representation by making it clear and closer to reality. A decision is based on some activities done earlier and the result is either of the two paths that follow this construct. For example in Figure 6.15, after processing the loan request, the control flow depends on whether the customer is a new customer or a past customer. With the exception of a flow chart, other process modeling languages cannot depict this situation as clearly as EPML.

Input and output data flows can be either physical or electronic. Nowadays, most processes require some sort of electronic data in addition to physical data. Both, electronic and physical data flows can be compactly and conveniently represented by the two different types of directed arcs in EPML.

6.10. Comparison Criteria

It is worthwhile to mention again that enterprise modeling covers both process modeling and data modeling. An architecture serves as a guideline for the modeler. An architecture may or may not include a process modeling technique. The purpose of the comparison exercise is to evaluate different process modeling techniques in the context of graphical process modeling. Hence, the criteria for comparison are related to the modeling techniques and are not framed to compare architectures. Each criterion is defined clearly so that the comparison of the techniques is on the same ground. The criteria were developed based on (i) study of various enterprise modeling approaches in the course of the thesis, (ii) the insights gained by modeling the HAL ordering example using different process modeling techniques. (iii) criteria available in literature, particularly in

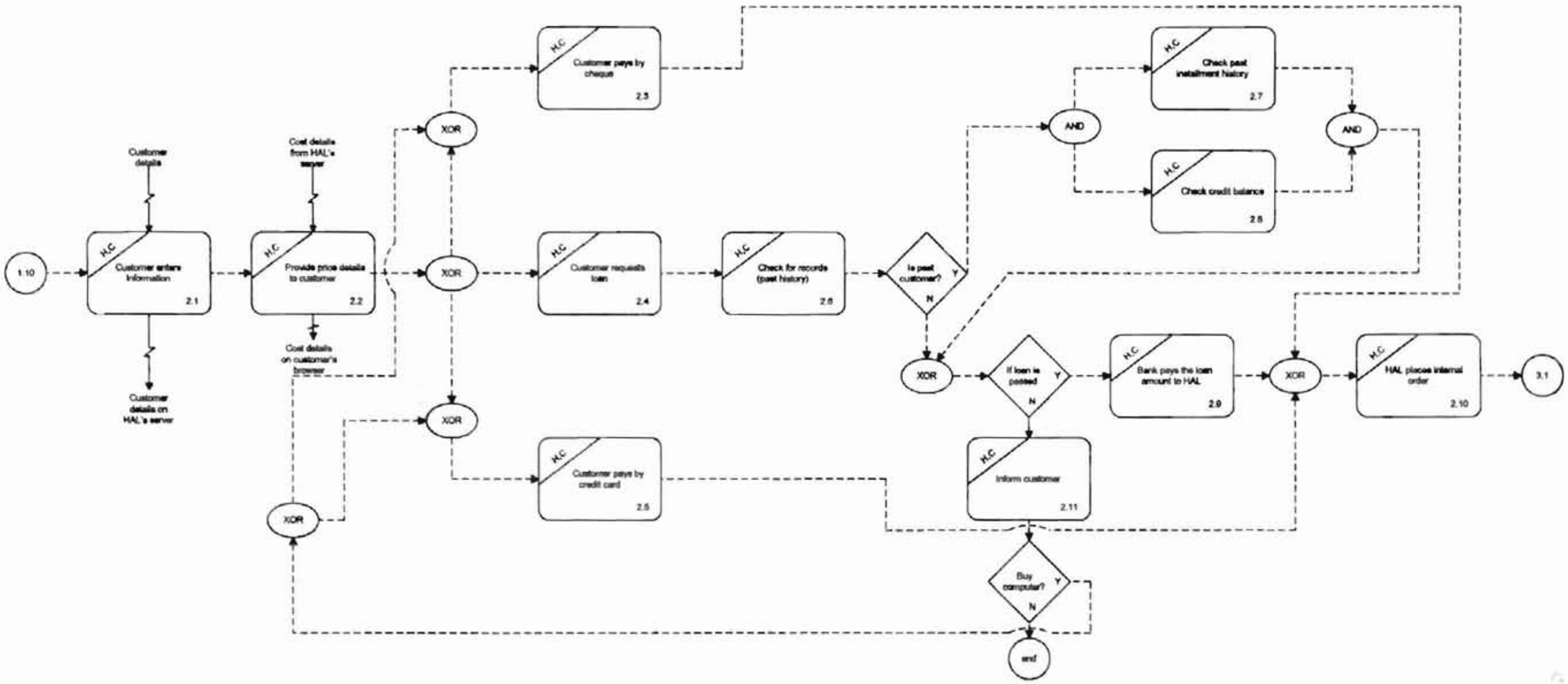

1.10
Customer details
H,C
Customer enters Information
2.1
Customer details on HAL's server
Cost details from HAL's server
Provide price details to customer
2.2
Cost details on customer's browser
XOR
Customer pays by cheque
2.3
Customer requests loan
2.4
Customer pays by credit card
2.5
Check for records (past history)
2.6
Is past customer?
Y
N
AND
Check past installment history
2.7
Check credit balance
2.8
If loan is passed
Bank pays the loan amount to HAL
2.9
Inform customer
2.11
Buy computer?
end
HAL places internal order
2.10
3.1

Figure 6.15: EPML Representation of HAL's Ordering system

CENT TC310 WG1 (1994) and iv) viewing process modeling from a user's perspective. The criteria are as follows:

1) **Ease of modeling**: The ease with which a modeler can model the process using the given constructs and their semantics and syntax.

2) **Control flow representation**: How well is the control flow in the process represented in the model?

3) **Accuracy of modeling**: How accurately does the process model represent the real-world process?

4) **Differentiation between physical and electronic data**: Are the physical and electronic data flows differentiated in the model?

5) **Clarity of semantics**: Is the meaning of the construct clear? Will the modeler use the construct for representing what is actually intended to be modeled by the construct?

6) **Clarity of syntax**: This is important particularly in modeling conjunctions and disjunctions in a flow. Is it clear as to how constructs are to be linked in a process representation so that the model accurately represents the situation.

7) **Is the technique self-contained?**: How well does the technique capture all relevant process details in a single representation? Does it require additional supporting techniques or a textual description?

8) **Separation of data and control flows**: Are there separate constructs for modeling data and control flows?

9) **Support for hierarchical modeling**: Can the technique model sub-processes and is there a traceability of sub process in both directions within the hierarchical structure?

10) **Ability to support formal analysis**: A graphical model alone cannot lead to a formal analysis of a process. However, graphical techniques can be used to collect information required for a formal analysis tool. What degree of informational support can the graphical technique provide to a formal analysis tool (CENT TC310 WG1 1994)?

11) **Sufficiency of constructs for representing relevant subjects of a process**: How comprehensive is the technique in providing constructs for modeling function, data, decision, time, space, organizational units, IT components, manufacturing components, material, product, human resources, etc.?

12) **Are the building blocks object-oriented?**: Do the constructs support object-oriented concepts of classes and objects?

The ratings for the comparative study were: low, medium and high (L, M and H). In a pilot exercise, a five-point scale was considered. Because of the limited scope of the comparison study, a three-point scale was finally chosen. A yes/no response is appropriate for certain criteria. If a criterion does not apply to a technique, then 'N.A.' is used to represent 'not applicable'. Ratings were decided on the basis of the insights developed by performing a thorough review of the existing process modeling techniques; by evaluating their strengths and limitations; by modeling a real-world example using various techniques; and by the author's participation in the NSF project on the development of a new process and performance modeling framework. It would be wise to mention that on the basis of further research, perhaps a better rating mechanism can be evolved. However, the criteria and the ratings presented in this thesis will certainly serve as a stepping-stone for future research.

Because of the lack of a process-modeling tool PERA, GRAI-GIM, and GERAM, were not included in the comparison study. UML is an emerging standard, and has not yet gained widespread acceptance as a business process modeling language (Schader and Korthaus, 1998). Also the relationships among the various diagrams within UML makes it difficult to use the activity diagram as a stand alone tool. Because of these reasons, UML was not included in the comparison study.

6.11. Summary

From Table 1, we can see that EPML has high ratings on most of the criteria. Of the existing techniques, SAP's EPC has high ratings on several criteria. EPML is user-friendly and provides flexibility to the user to represent a real world process accurately. It has a strict syntax, without sacrificing modeling flexibility. EPML retains the power and appeal of existing language constructs while adding some new ones to address the needs of today's enterprises. Data is divided into two types, namely physical and electronic, and a construct is provided to model the flow of each type. Presently, data is stored and manipulated electronically, and usually in a distributed computing environment. The EPML constructs allow the user to quickly identify physical data flows and manual (H) activities, which are usually prime candidates for improvement in a reengineering effort. There is one common construct for activities at different abstraction levels, and activities in different sub processes or abstraction levels can be linked by the numbering scheme. With the use of connectors we can easily model feedback at various abstraction levels. The preliminary design of a front end tool using EPML, which is shown in Figure 5.19, demonstrates the descriptive ability of EPML by representing only the most useful and necessary information graphically and capturing finer details such as time and cost parameters using user interface constructs. There is a clear separation of data flow and control flow in EPML. A graphical representation in EPML can be easily transformed into a Petri net representation for performing formal qualitative and quantitative analyses (Sivaraman 2001).

Table 1: Summary of the Comparison Study

Property	DFD	IDEF0	IDEF3	CIMOSA	SAP's EPC	TOVE	IEM	Baan's DEM	EPML
Ease of modeling	M	M	M	L	H	L	L	M	H
Control flow representation	M	L	H	M	H	M	H	H	H
Accuracy of modeling	M	M	M	L	H	L	L	H	H
Differentiation between physical and electronic data flow	No	No	N.A.	N.A.	No	No	No	N.A.	Yes
Clarity of semantics	H	M	H	L	H	H	L	H	H
Clarity of syntax	M	H	H	M	H	L	L	H	H
Is the graphical technique self-contained?	H	L	L	L	H	L	L	M	H
Separation of data & control flows	No	No	N.A.	N.A.	Yes	N.A.	Yes	N.A.	Yes
Supports hierarchical modeling?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Constructs for relevant subjects of process	M	L	L	L	H	M	L	L	H
Are building blocks object oriented?	No	No	No	Yes	No	No	Yes	No	No
Ability to support formal analysis	L	L	M	L	M	M	L	H	H
Compactness of representation	M	L	L	L	M	L	L	L	H

7. Conclusions and Future Work

This chapter is divided into three sections. The first section summarizes the research done. The second section lists the contributions of this thesis to the existing body of knowledge. The third chapter outlines areas for future work.

7.1. Research Summary

Focus of Research: An enterprise process model is a representation of a real world enterprise's structure and operations, typically using graphical modeling tools and techniques. The existing process modeling techniques have semantic and syntactic limitations, and as a result, the representation of the process may not always reflect reality. The primary objective of this research was to develop an enterprise process modeling language, EPML, which would provide a rich set of constructs to the modeler, and allow him/her to focus on the correct representation of processes instead of worrying about the semantic and syntactic limitations of the modeling language. EPML takes into account the shift to on-line and distributed execution of business processes since the advent of the Internet.

One of the main purposes of a process model is to support performance analysis of the enterprise. A unique feature of the EPML is that the representation emphasizes control flow within the process, thereby facilitating easy transformation of the graphical model into a formal representation such as the one using Petri nets, for performing qualitative and quantitative analysis.

Methodology Employed: The first step in this research was to conduct a thorough investigation of the existing process modeling languages to identify their purpose, strengths and weaknesses. The next step was to identify modeling constructs, theory, etc. of the available modeling languages that could be adapted for use within EPML. This was followed by the design of a new set of modeling constructs including a complete definition and detailed explanation of their semantics and syntax. The superiority of the proposed EPML over the existing modeling techniques was demonstrated using a comprehensive example and qualitative arguments.

Results: The outcome of this thesis is EPML, a graphical enterprise process modeling language that provides a rich set of constructs to the modeler to represent a real world process without distortion in the representation. Each graphical construct in the EPML has clearly defined semantics and syntax, which serves as an excellent guide to the modeler. EPML also supports hierarchical modeling. A process can be expanded to show its activities, and activities can be aggregated into a single parent process. The flexibility in the use of constructs provides the modeler the ability to accurately represent a real life process in a compact graphical representation.

7.2. Contributions

The main contribution is an enterprise process modeling language - EPML. EPML is an integral part of a new process and performance-modeling framework that is under development as part of an on-going NSF project. In the process of developing the EPML, extensive study was done on the existing process modeling languages. For the purpose of evaluating EPML, an example scenario was modeled using the existing languages and EPML. Hence, this thesis will help fellow researchers gain insight into the salient features of existing process modeling languages and EPML, and hopefully stimulate new research in the development of process modeling languages.

7.3. Future work

In the course of this research, many characteristics of existing process modeling techniques were studied. EPML blends the strengths of the existing techniques, and presents a user-friendly process modeling language. However, not all aspects of an ideal language could be included in EPML. Some directions for future research in this area are presented.

1. The representation of resources and their consumption is best represented in TOVE. TOVE has the starting and the ending resource states in the graphical language to depict resource consumption. The concepts of a resource being used, consumed, released, or produced, along with the temporal dependence of resources, can help the

user probe into resource utilization and in resource planning. Further work needs to be done on resource modeling within EPML.

2. A software environment based on EPML is another area for future work. Preliminary work along this direction was done as a part of the NSF project. With the help of latest technologies, most of the process details that are not represented graphically, can be captured as attributes. Examples of these are cost, time taken for completing an activity, and performance metrics.

3. Introducing the tool to select groups of users in industry, and then soliciting their feedback could establish the usability, strengths and limitations of EPML. The user feedback could lead to enhancement and extensions of the EPML constructs. This should be done in two steps.

    (i) The graphical process modeling technique should be introduced to users who are aware of existing techniques and who use them to model real-world examples. Their feedback can be valuable in improving the semantic and the syntactic details of EPML.

    (ii) A test version of the software tool with EPML as the front end and Petri nets as the underlying analytical engine, should be introduced to select users who may not be aware of existing process modeling techniques, but who use a software package for process modeling. Their feedback will provide enhancements to the modeling framework as a whole.

    Publishing articles in conference proceedings and journals would be an avenue for obtaining feedback from the academic community.

4. One of the main goals of process modeling is process improvement. To identify improvement opportunities or to evaluate proposed alternatives, it becomes necessary to perform quantitative or qualitative analysis using the process model. Integrating EPML with formal process analysis techniques such as simulation, queuing, and Petri nets would be another potential research topic.

5. Development of a process modeling methodology that would support EPML, and guide the user in the creation of a process model would be a future research topic.

The modeling methodology should be developed within the framework of the NSF project and should consider the technology that will be used in the development of a Web-based process-modeling tool.

Bibliography

Bernus, P., Nemes, L., and Morris, R., 1996, "The Meaning of an Enterprise Model," in *Modeling and Methodologies for Enterprise Integration*, Bernus, P. and Nemes, L. (eds.), Chapman & Hall, UK, 183-200.

Brandimarte, P. and Cantamessa, M., 1995, "Methodologies for designing CIM systems: A critique," *Computers in Industry*, 25, 281-293.

Bravoco, R.R. and Yadav, S.B., 1985, "Requirement Definition Architecture – An Overview," *Computers in Industry*, 6, 237 – 251

Burch, J.G., (1992), Systems Analysis, Design, and Implementation, Boyd and Fraser, Boston, MA.

CEN TC310 WG1, 1994, "An Evaluation of CIM Modeling Constructs – Evaluation Report of Constructs for Views According to ENV 40 003," *Computers in Industry*, 24, 159-236.

Chen, D. and Doumeingts, G., 1996, "The GRAI-GIM Reference Model, Architecture and Methodology," in *Architectures for Enterprise Integration*, Bernus, P., Nemes, L., and Williams, T.J., (eds.), Chapman & Hall, 102-126.

Colquhoun, G.J., Baines, R.W., and Crossley, R., 1993, "A State of the Art Review of IDEF0," *International Journal of Computer Integrated Manufacturing*, 6, 252 – 264.

Dao, K.L. and Rodjak, D., 1991, "CIM Flows from Data Flow Diagrams," *Manufacturing Systems*, 67-72.

Didic, M., 1994, "CIMOSA Model Creation and Execution for a Casting Process and a Manufacturing Cell," *Computers in Industry*, 24, 237-247.

Fox, M.S., Chionglo, J.F., and Fadel, F.G., 1993, "A Common-Sense Model of the Enterprise," *Proceedings of the 2nd Industrial Engineering Research Conference*, Norcross GA, 425-429.

GERAM, 1999, Generalized Enterprise Reference Architecture and Methodology, Version 1.6.2, Annex to ISO WD15704, IFIP-IFAC Task Force.

Harhalakis, G., Lin, C.P., Hillion, H., and Moy, K.Y., 1990, "Development of a Factory Level CIM Model," *Journal of Manufacturing Systems*, 9, 2, 116-128.

ISO 14258, 1999, Industrial Automation Systems—Concepts and Rules for Enterprise Models, World Wide Web version WG1.

ISO/FDIS 15704, 1999, Industrial Automation Systems, Requirements for Enterprise-Reference Architectures and Methodologies.

Kamath, M., Dalal, N. P., Kolarik, W. J., Chaugule, A.A, Sivaraman, E., and Lau, A.H., 2001, "Process Modeling Techniques for Enterprise Analysis and Design – A Comparative Evaluation", *Proceedings of the 10th Annual Industrial Engineering Research Conference.*

Karni, R., Molcho, G., and Kasif, L., 1999, "Blueprint for Research into Knowledge Process Modeling," *Baan Brothers Foundation 2nd Roundtable Proceedings*, Utrecht, The Netherlands, 77–85.

Keller, G. and Detering, S., 1996, "Process-Oriented Modeling and Analysis of Business Processes Using the R/3 Reference Model," in *Modelling and Methodologies for Enterprise Integration*, Bernus, P. and Nemes, L. (eds.), Chapman & Hall, 69-87.

Knowles, G., 1999, "Next Generation Enterprise Modeling: A Step Beyond Baan Application Modeling," *Baan Brothers Foundation 2nd Roundtable Proceedings*, Utrecht, The Netherlands, 95-115.

Kosanke, K., 1995, "CIMOSA – Overview and Status," *Computers in Industry*, 27, 101 – 109.

Kosanke, K., 1996, "Process Oriented Presentation of Modelling Methodologies," in *Modelling and Methodologies for Enterprise Integration*, Bernus, P. and Nemes, L. (eds.), Chapman & Hall, 45-55.

Marshall, C., 1999, *Enterprise Modeling with UML, Designing Successful Software Through Business Analysis*, Addison Wesley, Boston, MA.

Mertins, K., Jochem, R., and Jakel, F.-W., 1997, "A Tool for Object-oriented Modeling and Analysis of Business Processes," *Computers in Industry*, 33, 345-356.

Mertins, T., Susseguth, W. and Jochem, R., 1992, "An Object Oriented Method for Integrated Enterprise Modeling as a Basis for Enterprise Co-ordination," *Enterprise Integration Modeling: Proceedings of the First International Conference*, MIT Press, MA, 249-258.

Millet, I., 1999, "A Proposal to Simplify Data Flow Diagrams," *IBM Systems Journal*, 38, 1, 118–121.

Pascoe, G.A., 1986, "Elements of Object-Oriented Programming," *BYTE Magazine*, McGraw-Hill Inc., New York.

Savolainen, T., Beeckmann, D., Groumpos, P., and Jagdev, H., 1995, "Positioning of modelling approaches, methods and tools," *Computers in Industry*, 25, 255-262.

Schader, M. and Korthaus, A., 1998, "Modeling business processes as part of the Booster approach to business object-oriented system development based on UML," *IEEE Proceedings of the Second International Workshop on Enterprise Distributed Object Computing Workshop*, 56-67.

Scheer, A.-W., 1992, "*Architecture of Integrated Information Systems: Foundations of Enterprise Modelling*," Springer-Verlag, Berlin.

Sivaraman, E. 2001, "On the Use of Petri Nets for Business Process Modeling," Working Paper, Center for Computer Integrated Manufacturing, Oklahoma State University.

Song, I.-Y. and Froehlich, K., 1994, "Entity-Relationship Modeling," *IEEE Potentials*, 13, 5, 29-34.

Srihari K., Emerson, R.C., and Cecil, J.A., 1990 "Modeling Manufacturing with Petri Nets," *Journal of CIM Management*, 6, 3, 15-21.

van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., and Barros, A.P., 2000, "Workflow Patterns," *BETA Working Paper Series*, WP 47, Eindhoven University of Technology, Eindhoven, Netherlands.

van der Aalst, W.M.P. and van Hee, K.M., 1995, "Framework for Business Process Redesign," *IEEE Proceedings on the Fourth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 36-45.

van der Rijst, N., 1997, "Dynamic Enterprise Modeling Development Method: Guidelines and Conventions," User Manual, Baan Business Innovation, The Netherlands.

Vernadat, F.B., 1996, "CIM Business Process and Enterprise Activity Modeling," in *Modeling and Methodologies for Enterprise Integration*, Bernus, P. and Nemes, L. (eds.), Chapman & Hall, UK, 170-182.

WFMC-TC-1011, 3.0, 1994, "Workflow Management Coalition Terminology & Glossary," Workflow Management Coalition, Hampshire, U.K., 1-65.

Whitten, J.L. and Bentley, L.D., 1998, *Systems Analysis and Design Methods,* Irwin/McGraw-Hill, Boston, MA.

Williams, T.J., 1994, "The Purdue Enterprise Reference Architecture," *Computers in Industry*, 24, 141 – 158.

Zelm, M., Vernadat, F.B., and Kosanke, K., 1995, "The CIMOSA Business Modeling Process," *Computers in Industry*, 27, 123 – 142.

Zurawski, R. and Zhou, M.C., 1994, "Petri Nets and Industrial Applications: A Tutorial," *IEEE Transactions on Industrial Electronics*, 41, 6, 567-583.

VITA

AMIT A. CHAUGULE

Candidate for the Degree of

Master of Science

Thesis: A USER-ORIENTED ENTERPRISE PROCESS MODELING LANGUAGE

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Mumbai, India, on January 20, 1974, the son of Avinash and Archana Chaugule.

Education: Received the Bachelor of Engineering Degree in Automobile Engineering from M.H. Saboo Siddik College of Engineering, Mumbai, India, in May 1995; graduated with first class honours. Completed the requirements for the Master of Science degree in Industrial Engineering and Management at Oklahoma State University in August 2001; received the OSU Graduate Research Excellence Award for outstanding accomplishments in M.S. thesis research.

Experience: Worked as a Product Engineer with India's leading tractor manufacturing company, from July 1995 to July 1999. Gained experience in solving product and process problems. Hands-on experience in vendor development, value engineering, and documentation of processes and quality procedures for ISO 9001 quality standards. Worked as a graduate research assistant in the School of Industrial Engineering and Management at Oklahoma State University from August 2000 to August 2001, on a NSF funded project on the development of a user-oriented process and performance modeling framework for enterprise systems. Participated in preparing research proposals. Lead student researcher for the development of a process modeling language and associated modeling constructs. Worked for OSU Extension Services as a summer intern, from May 2000 to August 2000 and from May 2001 to June 2001, on solving industrial problems.

Professional Memberships: Alpha Pi Mu (National Honor Society for Industrial Engineers).