This is to certify that the

dissertation entitled

INTEGRATION OF ENVIRONMENT INTO PRODUCT DESIGN AND MANUFACTURING: THEORY AND IMPLEMENTATION

presented by

Youngsun Chun

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Electrical Eng

R. L. Tummala
Major professor

Date 3-7-2000

DATE DUE	DATE DUE	DATE DUE

INTEGRATION OF ENVIRONMENT INTO PRODUCT DESIGN AND MANUFACTURING: THEORY AND IMPLEMENTATION

By

Youngsun Chun

A DISSERTATION

Submitted to
Michigan State University
in partial fulfilment of the requirments
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering

2000

ABSTRACT

Integration of Environment

into

Product Design and Manufacturing:

Theory and Implementation

by

Youngsun Chun

Environmentally Conscious Manufacturing(ECM) has two major goals such as developing green technology and analytical modeling tool which can assess the consequences of different strategies. Overall, modeling interactions between manufacturing plants and environment is very huge and complex task.

Most quantitative tools are concerned with environmental accounting system without concerning of feasibility and impact of process network structure.

This disseratation is an attempt to develop a framework to study the impact of alternative technologies, strateges, and designs based on process network theory. As a result, a computer modeling and simulation tool, Mass-Energy Based Simulation(MEBS), instantiating an ECM tool is presented.

Dedicated
to
My Father and Mother

Seok Hyun Chun and *Hyung Ae Kim*

ACKNOWLEDGEMENTS

Looking upon the pathway that I have chosen, this is the moment to acknowledge those who shaped me and helped me for being where I am. My first thank goes to my beloved mother and father, and my family. They made me proud of what I am.

Also my gratitude and appreciation goes to my academic advisor Dr. Lal Tummala for his guidance and financial support during my research work. This dissertation could not be completed without his support. And this work is partially supported by NSF grant DMI-9528759.

I am also grateful to my guidance committee members: Dr. Hassan Khalil, Dr. Steven Melnyk, and Dr. Charles MacCluer for their interests, guidance, advice, and encouraging.

I am indebted to Ms. Mary and Mr. Norman Robison, Mr. Charles McNease, Mr. H. Kay for their encouragement and friendship. Ms. and Mr. Robison, taught me how to communicate in written English. Mr. McNease, beginning with one of my volunteer examiners for my amateur radio license, shared lots of his experiences and friendship.

Finally, I am asking forgiveness to my wife Kyounghwa, and to my dear children Sungah and Sungwoo for not sharing time together much.

Contents

List of Tables

List of Figures

Part I
INTRODUCTION

1 Introduction

Traditionally, products are designed for their appearance, technical (electrical, mechanical, and etc.) performance, and functionality. The environmental impact of this design on the manufacturing processes, product use and disposal are seldom considered. With the increasing demands on conservation of natural resources and environment, modern firms have begun to incorporate environmental concerns into product design and manufacturing. This process is variously called environmentally conscious product design, environmentallly responsible product design, design for environment(DfE) or green design[24, 10]. As an extension of existing DfX(Design for X) strategies, the DfE focus begins at the product development stage and runs all the way through the distribution[29]. It is important to recognize that decisions made during the design phase have a profound impact on the entire life cycle which involves the manufacture, product use, and product reuse or disposal [40]. Manufacturing systems that incorporate environmental considerations similarly are called Environmentally Conscious Manufacturing system(ECM) or green manufacturing systems. Billatos and Basaly [33] define the goals of *Eivironmentally Conscious Manufacturing(ECM)*, also coined as *green engineering* or *green technology*, as follows:

Waste reduction is justified based on financial analysis without concern for the added environmental benefits. Total Quality Management(TQM) and Just In-Time(JIT) manufacturing are example strategies for achieving this goal.

Materials management aims for economical recovery of materials or finished products for reuse. The three categories of strategies to achieve this goal are *Design for recycling(DfR)*, *Design for disassembly(DfD)*, and *toxic management.*

Pollution prevention has the goal of eliminating the use of manufacturing processes that generate pollution. This differs from *pollution control*, also known as *end-of-pipe(EOP)* solution, which refers to the treatment of harmful by-products after they have been produced.

Product enhancement is a design activity to reduce resource requirements, waste, and pollution during product's use through its operable life, usually motivated by regulations to control harmful by-products.

1.1 Why ECM?

Population grows in a geometric ratio in an environment which supplies unlimited resources and tolerates unlimited waste, unless the environment is managed. In reality this exponential growth is not true, because every population depends on others in one way or another, the earth has limited resources, and humans have limited tolerance of waste in the environment[12].

There are many signs of environmental stress indicating that the health of the environment today is worse than that of yesterday. If management or regulation toward sustainable products and services which can be produced indefinitely without adding any environmental stress are not done, the only one earth we share with others will be worse day by day until disaster may strike all of us. The importance of supporting the environment is increasing as both the products and services demanded by the human population grow at the cost of environment resources and continuous increase in the human population. While better technologies and more focused effort by individuals have increased the productivity and services, the pressure understanding interactions among different entities such as plants, economies, and environmental loads also have increased.

Considering industry is the major producer of the solid waste as in Figure 1, environmentally friendly design in manufacturing plants would greatly affect the rest of the life cycle of a product.

Due to increasing environmental awareness, the companies have recognized the economic and social advantages of designing and manufacturing environmentally responsible products, so called "green products" and placed greater emphasis on incorporating environmental concerns into product design and manufacturing. AT&T, Xerox, Intel, Hewlett Packard, Tektronix, 3M, and Texas Instruments corporations have integrated DfE concepts into their product development and design and the benefits of ECM strategies are shown in Table 1 from [29]. For example, Intel Corporation has retooled their

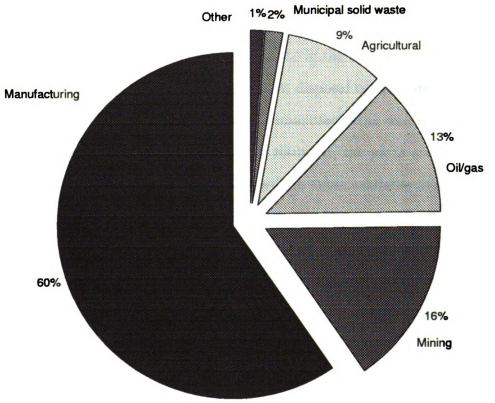


Figure 1: Sources of solid waste

Corporation	Estimated Savings	DfE Strategies
Intel	$1 Million	Recycling/Reuse
Xerox	$200 Million	Remanufacturing
Hewlett-Packard	$17 Million	Recycling
3M	$1 Billion	Recycling, Remanufacturing

Table 1: Benefits of DfE implementation and ECM strategies used by a sample of domestic manufacturers

product design so they do not generate waste in the first place. This lead to savings on chemical purchases, as well as on disposal costs. Similarly, Xerox and 3M Corporation have incorporated remanufacturing and recycling into their design. Once expired products are returned, the parts are segregated into reusable and unusable parts which will be made available as spare parts for newly manufactured products.

To minimize the impact on the environment, designers should took into consideration the materials used, energy efficiency of the processes used, wastes generated during manufacture, product use and disposal. In order to achieve these objectives, some guidelines are provided and are as follows [17]:

1. Choose abundant, nontoxic, nonregulated materials if possible. If toxic materials are required for a manufacturing process, try to generate them on site rather than by having them made elsewhere and shipped.

2. If possible, choose natural materials rather than synthetic materials.

3. Design for minimum use of materials in products, in processes, and in service.

4. Try to get most of the needed materials through recycling streams rather than through raw materials extraction.

Darnell et al[29] state that two essential future needs for successful ECM are 1) green technology development to minimize waste in processes and 2) the development of analytical modeling tools that can assess the environmental consequences of different design and managing strategies.

Sweatman and Simon [36] view green products as different from sustainable products which depend on what kinds of products are made in what quantity. In other words, the degree of sustainability – also known as eco-efficiency – is measured in terms of biodegradability, DfE emphasizing renewability, and consumption patterns. They made three categories of products by the degree of sustainability as follows:

100% eco–efficiency : sustainable products, those that can be produced in large quantities indefinitely.

high eco–efficiency : products having environmentally-conscious features but which can be produced eithter in limited quantity or for a limited time

low eco–efficiency : products which deplete non-renewable resources, damageshuman health, or pollute the environment.

This thesis presents a new methodology and computer aided tool which can assist in decision making and green technology assessment to realize those goals of ECM. The thesis is organized as follows:

1. Literature survey of the previous work
2. Problem definition
3. Problem statement
4. System and modeling
5. Methodology
6. Mass-Energy Based(MEB) model description
7. MEB network
8. MEB Language(MEBL)
9. MEB network execution model
10. Graphic user interface
11. Case studies
12. Survey
13. Conclusion

2 Previous Work

Computer tools are available to aid designers in analyzing the impacts of designs on the environment or providing guidelines of design strategies. They are either analysis tools based on Life Cycle Analysis(LCA)[32, 35, 4] or strategy and planning tools which are often linked with other Computer Aided Design(CAD) softwares[15] or handbooks. Generalized description of LCA is described in [30, 1, 9].

AT&T proposed a quick way to assess environmental impacts using an evaluation questionaire to be answered by people who are involved in the life cycle of a product and its alternative [18]. They are asked to specify the degree of environmental assessment using numbers between zero and four and they are asked to fill the product assessment of the 5x5 abridged matrix with rows describing five stages of a product life cycle stages as in Figure 2 and columns representing five categories of envirnonmental concern. Guided by checklists, DfE assessor assigns a number from 0(highest impact) to 4 (lowest impact) to each element of the matrix. Then the final 25 scoring elements are plotted on a *target plot* which is a polar form of a transformed bar graph to display environmental impact. The circumference of target plot is divided into 25 sections. The outermost circle represents the value 0 and innermost circle represents the value 4. Then the bull's-eye represents a product of the lowest environmental impact. Although this method is easy to apply, and may become a step toward DfE, this does not provide objective scoring nor

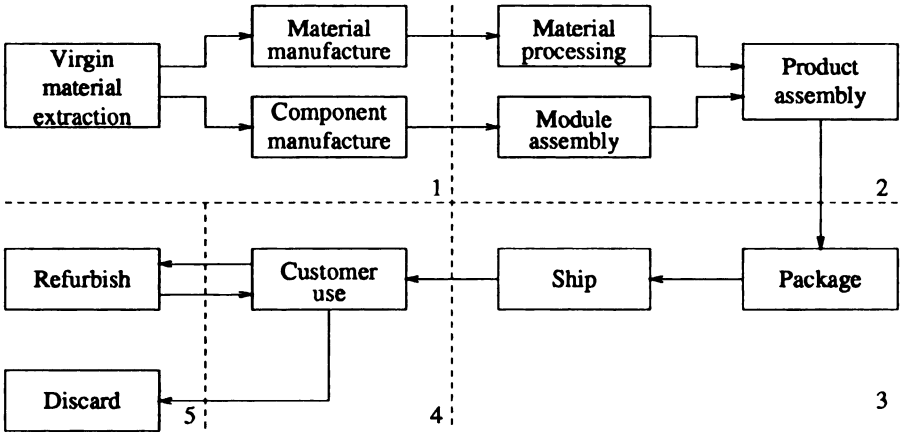


Figure 2: Activities in the five life-cycle stages of a product

guidance regarding the relative importance of different issues.

The abridged matrix proposed by AT&T[18] and ecoindicator[15] are shown in Figure 3 and Figure 6 as examples of qualitative and quantitative abridged life cycle assessment tools respectively.

This method was also applied to facilities design and planning and supply line analysis[28]. Examples are shown in Figure 4 and 5

The example of Figure 6 is rather a nice example of data visualization than a system modeling tool.

Sheldon[34] made an attempt to assess the environmental responsibility of an manufacturing process and propsoed *Environmental Quotient*(EQ) by

$$EQ = AU \times U$$

where AU(atom utilization) is calculated by dividing the molecular weight of the desired product by that of the sum total of all substances produced and

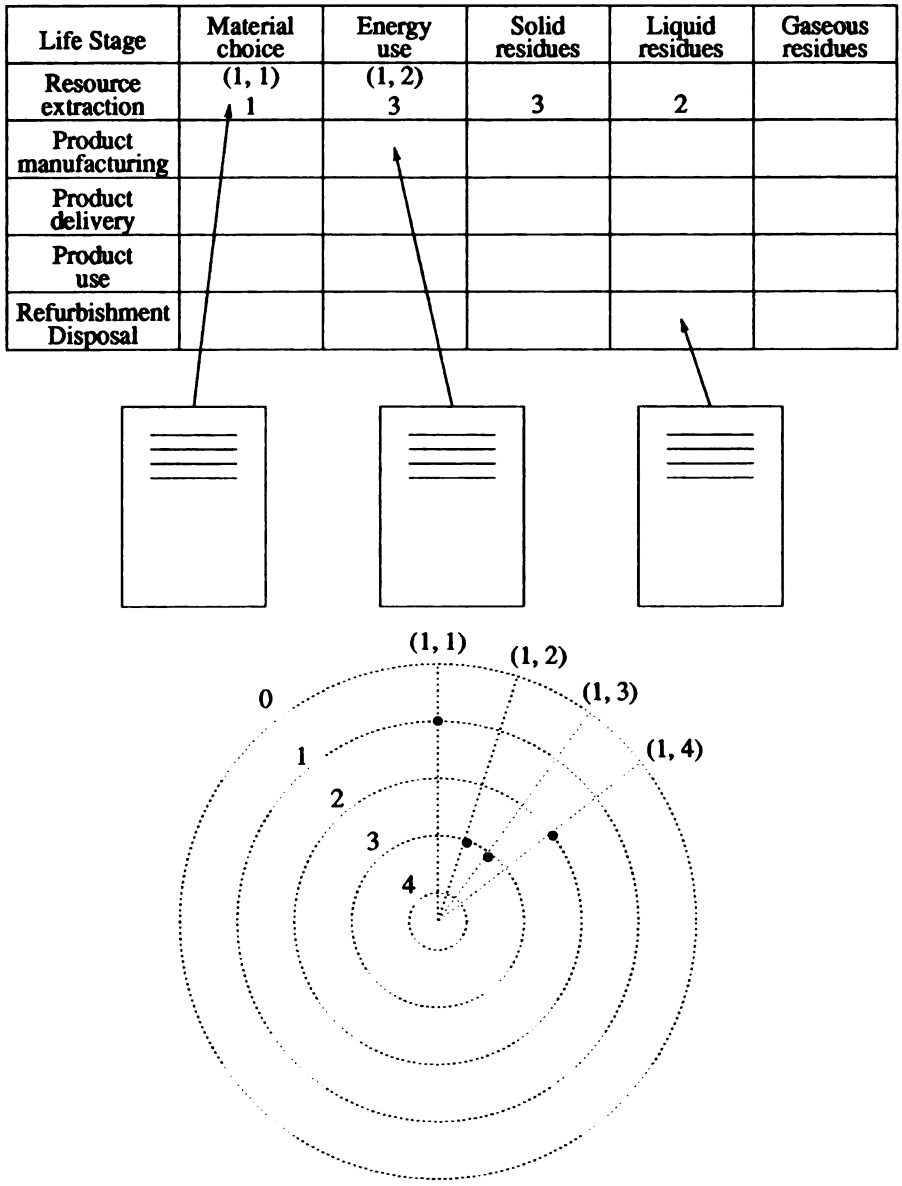


Figure 3: The environmentally responsible product assessment matrix and the target plot

	Ecosystem Disruption	Material Resources	Energy Considerations	Air Pollution	Water Pollution	Solid Waste
Site location, preparation and infrastructure						
Construction/Renovation						
Facility/utilities						
Installation of process equipment						
Facility Operation						
Use						
Maintenance						
Process activities						
Change of process activities						
Facility Closure						

Figure 4: Facilities LCA matrix

Facilities LCA Matrix(L.Weinberg, Environmental Program Administrator, Boeing Co., "The Development of a Streamlined, Environmental Lifecycle Analysis for Facilities", Int'l Symposium on Electronics & the Environment, 1998)

	Semi-conductors	PCB's	Connectors	Passive Components	Cable & Wires	Cabinets & Racks	Materials & Finishes	Joining Techniques	Hybrids & Assemblies
Material Substitution									
Substance Use Reduction									
Waste & Emission Source Reduction									
Energy Use Reduction									
Design for Re-usability									
Design for Disassembly									
Design for Recyclability									
Design for Separability									
Design for Energy Recovery									
Design for Disposability									

Figure 5: Matrix: Supply line to environmental design practices

Matrix: Supply Line to Environmental Design Practicies(M.H.Nagel, Lucent Technologies, "Environmental Quality in the Supply Line: a New Approach", Int'l Symposium on Electronics & the Environment, 1998)

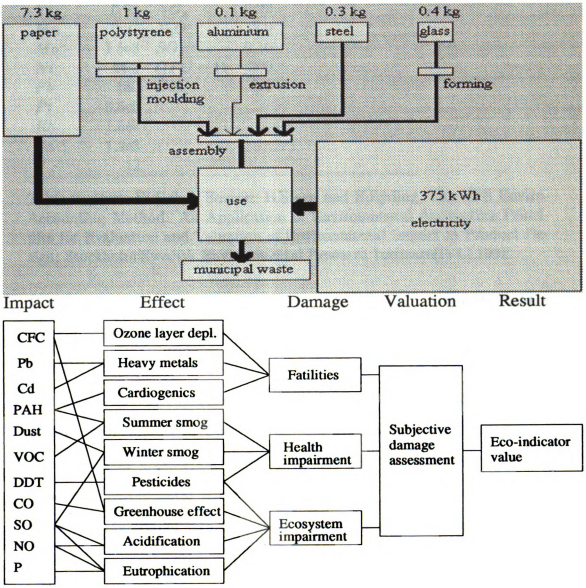


Figure 6: Application of the Eco-Indicator as a tool

Raw Materials		Emissions-Air		Emissions-Water	
Co	76	CO_2	0.09	*Nitrogen*	0.1
Cr	8.8	CO	0.27	*Phosphorus*	0.3
Fe	0.09	NO_x	0.22		
Mn	0.97	N_2O	7.0		
Mo	1.5e3	SO_x	0.10		
Ni	24.3	$CFC-11$	300		
Pb	180	CH_4	1.0		
Pt	3.5e5				
Rh	1.8e6				
Sn	1.2e3				
V	12				

Table 2: Units:ELU/kg. Source: B.Steen and S.Ryding, The EPS Enviro-Accounting Method: An Application of Environmental Accounting Principles for Evaluation and Valuation of Environmental Impact in Product Design, Stockholm:Swedish Environmental Research Institute(IVL),1992.

U is an environmental index, a measure of toxicity[34]. Although Sheldon did not suggest how to assign the index, the Swedish Environmental Institute(IVL) and Volvo Car Corporation have developed an analytic tool, the Environmental Priority Strategies(EPS) system. The index is represented in 'Environmental Load Units'(ELUs) per kilogram(ELU/kg), per square meter(ELU/m^2), per spots($ELU/spot$), and etc. Those indices are calculated by environmental scientists, ecologists, and materials specialists for every raw material[6]. Table 2 show for some examples of environmental indices.

An example of the use of EPS system to compare the front end made of GMT composite and galvanized steel is shown in Table 3.

The proposed ELU concept, an agreed set of environmental indices, has

Materials & processes	Production			Product Use			Waste Incineration			Total ELU
	ELU/kg	kg	ELU	ELU/kg	kg	ELU	ELU/kg	kg	ELU	
GMT composite										
Production:										
GMT material	0.58	4.0	2.32							2.32
Reused production scrap	-0.58	0.3	-0.17							-0.17
compression molding	0.03	4.0	0.12							0.12
Product Use:										
Petrol				0.82	29.6	24.27				24.27
Recycling:										
GMT material							-0.21	3.7	-0.78	-0.78
Total sum		2.27			24.27			-0.78		25.76
								Reuse		
Galvanized steel										
Production:										
Steel material	0.98	9.0	8.82							8.82
Steel stamping	0.06	9.0	0.54							0.54
Reused production scrap	-0.92	3.0	-2.76							-2.76
Spot welding(spots)	0.004	48	0.19							0.19
Painting(spots)	0.01	0.6	0.02							0.02
Product Use:										
Petrol				0.82	48.0	39.36				39.36
Recycling:										
Steel material							-0.92	6.0	-5.52	-5.52
Total sum		6.81			39.36				-5.52	40.65

Table 3: The product use is based on 1 year of use. Calculation of ELU for automobile front ends. Source: S.Ryding, B.Steen, A.Wenblad, and R.Karlson, The EPS system - An LCA concept for cleaner Technology and Product Development Strategies, and Design for the Environment, Paper presented at EPA Workshop on Identifying a Framework for Human Health and Environmental Risk Ranking, Washington, D.C., June 30-July 1, 1993.

great advantage of objective scoring system, flexibility of modeling a system, and capability to compare different DfE strategies. Still, it provides neither how processes interact together nor feasibility of implementing a DfE strategy. Matthews and Lave[21] proposed another method which accounts for costs in a manufacturing setting and shows the optimal price for all cases as in Figure 7.

In general the ECM tools range from simple to complex[32, 35, 4]. Fiskel [14] classified these tools into qualitative and quantitative methods and discussed the advantages and disadvantages of these mothods. [21] proposed a generalized system model which accounts for costs in a manufacturing setting and shows the optimal price for all cases as in Figure 7.

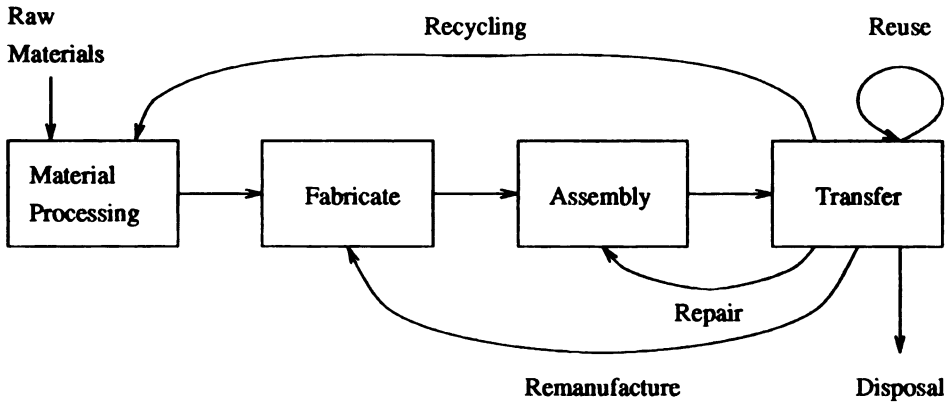
An expert system tool was discussed to determine improvements for easy assembly, disassembly, and material suggestions based on CAD software or input of a product specification[8].

2.1 Qualitative methods

Qualitative methods are further divided into two types of methods: *checklists* and *matrices*.

1. Advantages

    - Easy to apply
    - Minimal data is required
    - Not as expensive as quantitave method.



The Model

C1 = M1, M1 is the initial cost of manufacturing

Ci = (1-k)M1 + T, where k is the reusable product of a product and T is turnaround cost

Use	Man cost	Turnaround	Use disp	End disp	End toxic	Cost(i)	Life cost	Price
1	40	0	0	0	0	40	40	40
2	12	2	3.6	0	0	17.6	57.6	28.8
3	12	2	3.6	0	0	17.6	75.2	25.7
4	12	2	3.6	0	0	17.6	92.8	23.2
5	12	2	3.6	0	0	17.6	110.4	22.8
10	12	2	3.6	0	0	17.6	198.4	19.8
20	12	2	3.6	0	0	17.6	374.4	18.7
End	0	2	0	5	7	14	388.4	19.4

Assuming k = 0.7, M1= 40, T = 2, n = 20

Life Cycle Modeling of Manufacturing by H.S.Mathews and L.B.Lave(Carnegie Mellon University)

Figure 7: Life cycle modeling of manufacturing

Because of above characteristics, this method may be the first step in implementing DfE especially in identifying probable improvements.

2. Disadvantages

   - This method can show existence of performance improvement, but not how much improvement, even though using numerical scores.
   - This method provides no guidance regarding the relative importance of different issues. For example, is it more important to reduce source volume or to assure recyclability?
   - People may fail to become sufficiently involved in DfE issues and may overlook important opportunities or problems that are not covered on the list.

2.2 Quantitative methods

1. Advantages

   - Can develop an inventory of the environmental burdens associated with a product and process by identifying and quantifying energy and materials used and wastes released to the environment.
   - Assess the impact of those energy and material uses and releases on the environment.
   - Evaulate and implement opportunities to effect environmental improvements.

2. Disadvantages

- Defining system boundaries for LCA is controversial.
- LCA is data-intensive and expensive to conduct.
- Inventory assessment alone is inadequate for meaningful comparison, yet impact assessment is fraught with scientific difficulties.
- LCA does not account for other nonenvironmental aspects of product quality and cost.
- LCA cannot capture the dynamics of changing markets and technologies.
- LCA results may be inappropriate for use in eco-labeling.

In order to overcome these shortcomings, an ECM tool was developed based on PNT theory also developed at Michigan State University.

3 Problem Statement

Due to the complexity of LCA, the subjective, vague, inconsistent guidelines inherent in these semi-quantitative approaches are likely to lead to ad hoc evaulation. Its primary weakness is that results are often subject to individual interpretation[11]. And the major disadvantage of the above methods is that they provide very limited guidance for the improvement of the process.

My thesis will provide a systematic way to deal with this problem without sacrificing the detail necessary for environmental and economic impact of the various strategies used in product design and manufacturing.

Furthermore, we have developed a new paradigm for the improvement and management of the process using the technology and the tools developed in this thesis as shown in Figure 8.

More specifically, we will discuss a new quantitative tool which provides a systematic way to deal with this problem without sacrificing the detail necessary for evaluating environmental and economic impact of the various strategies used in product manufacture, use and disposal as follows:

1. It is comprehensive and thus can be used for the entire life cycle of the product. This is important because of conflicting requiremnets between different life stages.

2. It is isomorphic to the physical activities of the life cycle(material flows) that are responsible for pollution. Materials cause polution and the

tools should be able to provide this information as a function of management strategies used.

3. The models are based on fundamental principle of material energy and balance.

4. It allows the user the capability to perform sensitivity analysis. This will help to evaluate the impact of less accurate data on the outcome.

5. It allows "what-if" simulation capability.

6. Helps to evaluate the impact of changes in processes and/or technologies(for example, the impact of automation or recycling).

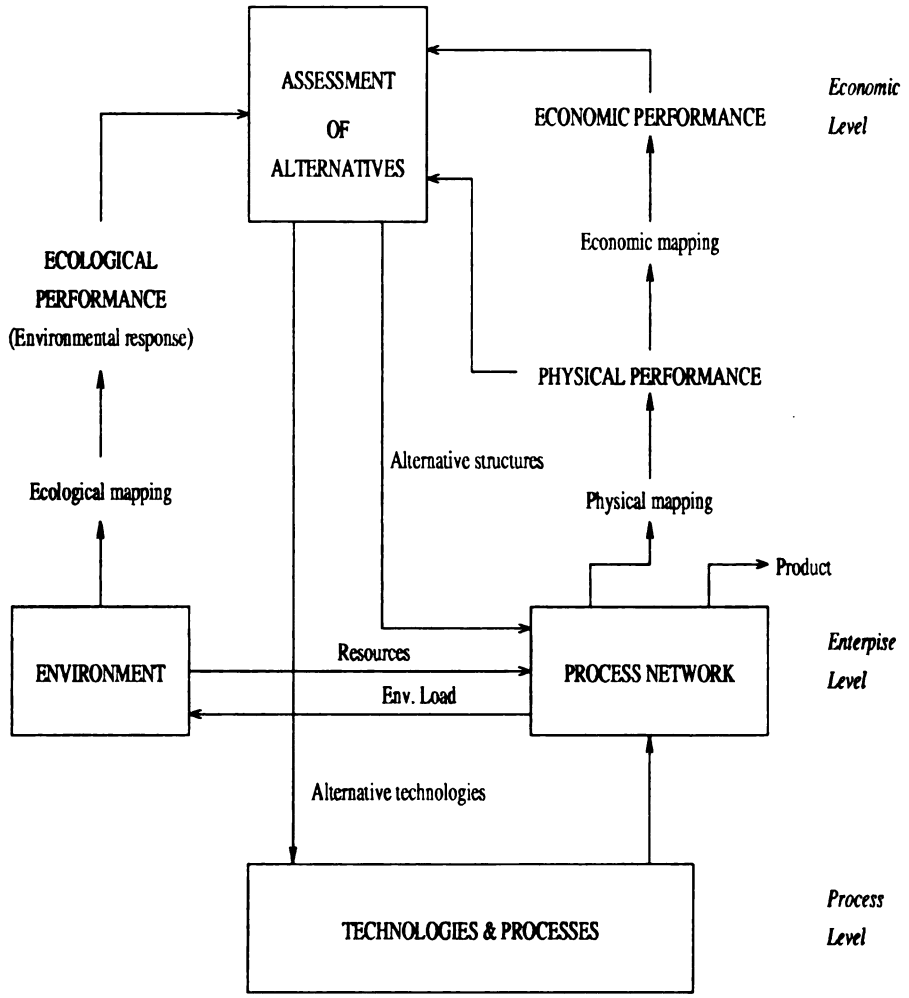


Figure 8: Integrated approach to manufacturing system analysis and design

Part II
Methodology

4 System and Modeling

4.1 System Definition

A system is defined as an aggregation or assemblage of objects joined in some regular interaction or interdependence, simply put, a set of interacting objects called subsystems[5]. And the system is often affected by changes occurring outside the system[7]. Some system activities may also produce changes that do not react on the system. Such changes occurring outside the system are said to occur in the *system environment* [16].

The definitions of system in Webster Dictionary are:

- structural design
- a usually miniature representation of something
- a pattern of something to be made
- an example for imitation or emulation
- a description or analogy used to help visualize something that cannot be directly observed
- a system of postulates, data, and inferences presented as a mathematical description of an entity or state of affairs.

Often, physical systems under studies are likely to be too large, too broad, or too complex to characterize as a whole. Many theoretical suggestions about how to partition such a system have been suggested in [20]. In order to circumvent such problems to get a satisfactory solution which might not be the best or exact solution, the large system needs to be broken down to a number of subsystem small enough to be tractable problems and then reduce the number of objective measurement parameters to a smaller number of parameters relevant to the study objectives.

Then the system can be represented as interconnection between each block which is an aggregation of entities. A model, whether it is physical or mathematical, is used to study a system as a substitute and in most case as a simplification of the system.

Based on assumptions on the physical system, this process of selecting parameters through system data gathering and data analysis chooses the system boundary and identifies its entities along with their relationships together.

4.2 Model

Simulation refers to a broad collection of methods and applications to mimic the behavior of real systems[25].

Simulation

Meriam Webster dictionary defines simulation as:

- The initiative representation of the functioning of one system or process by means of the functioning another

- Examination of a problem often not subject to direct experimentation by means of simulating.

With the exception of design by creation without any help of previous knowledge of similar problems before, most designs can be done from previous designs by modification or selection [3].

Measurability study in the modeling phase is one of the key issues in studying a system, whether the primary purpose of the study is looking for a new creation of a system, or enhancement of an exisiting system, or even controlling an existing real system. Even though most small store managers do not use computer simulation, they are making every effort to maximize their profits by constructing their own store models and by making "what-if" analysis in their minds.

Depending on the size of a system and the goal of extent of fine detail, an appropriate scale of modeling is required to meet the goal of a system study. While simulation can be a replica of a real physical system, or mathematical model, computer simulation refers to methods for studying a wide variety of models of real world systems by numerical evaluation using software designed to initiate the sytem's operations or characteristics, often over time.

It quantifies assumptions represented by conceptual maps and explores their impact on various "what-if" situations This is very valuable to managers who want to test new assumptions in new ways. One of the advantages of simulation is that it enables us to gain valuable insight into how their assumptions interact with each other. This insight offers an unprecedented

competitive advantage by improving decision-making and problem-solving skills.

It is especially effective when used as a scenario planning tool. The benefits of simulation are:

- Risk-free strategy experimentation
- Enables managers to explain their ideas more easily and insights to other people inside and outside the organization.
- "what-if" type questions in comparison to other alternatives

Modeling is a formal representation of a system followed by simulation which assigns semantic meanings for its formal representation.

5 Methodology

Top-down design methodology traditionally has been used to cope with design complexity[22]. Here, both top-down design which is obtained by goal oriented approach and bottom-up design which is used to estimate the costs of products are used together.

In order to represent a system, MEB DfE tool have both textual description and graphical description. The analytical description of each process in MEB DfE tool is based on both PNT and MEB economic model [38, 39, 37]. The textual description of a sytem, MEBL in section 8, has been designed for brief mathematical modeling of a process especially in dealing with matrix computation.

The MEB graph model is also used for better description of interprocess communication and interactions among processes. This modeling method hides the cryptic nature of textual description and provides a global conceptual map of a very large system.

Modeling a system begins with identifying every process and its output products, input materials, and byproducts(waste) of each process. Then with all the measurements available after construction of an MEB modeling and simulation, the next question is what to do with all those evidences. Any reasoning, validating, scientific judgement is based on those evidences which may lead to modification of a model, or different judgements. To make a judgement, possible decision categories need to be defined first. Then the decision problem would be assigning measurements to each of the categories. The next question is how each category is judged compared to other categories to quantify a global environmental burden. It seems to be next to impossible to find an unified formula to lead to an unique decision agreed upon by all the communities. Still, it would be nice having such a formula pleasing all the communities.

Here, the environmental impact is computed for each byproduct first and then summed up as an eco-indicator value of a whole system. The environmental impact metrics used in this tool came from the EPS Enviro-Accounting method by Swedish Environmental Researach Institute(IVL). The MEBL description of this metrics is in Section 8.6. To access such an interoperable database provided by environmental communities, the textual

description - MEBL - also understand SQL-like syntax.

6 What is MEB model ?

Nowadays, the importance of impacts on the environment by manufacturing processes motivates to evaluate environmental burdens associated with a product. The Mass-Energy Based Modeling System(MEBMS) is attempting to realize the evaluation of environmental burdens. This paper describes a tool based on the Mass-Energy based economic model[38, 39, 37].

The manufacturing environment consists of many processes and procedures applying to materials, and disposals associated with its energy, and cost.

Historically, the effectiveness of manufacturing has been evaluated by monetary accounting system. Trends in manufacturing towards decentralization and outsourcing of business requirements need an effective modeling tool to coordinate the business activities.

While the majority of financial accounting systems is powerful, this approach alone does not show environmental factors, technical factors, energy cost, and monetary factors easily due to the complexity of the interconnection between processes or between processes and environment.

As more information is flourishing from various disciplines and processes, the difficulties of system modeling increase in terms of creating a model, evaluation of the model, maintaining the model, and proficiencies in programming language skill.

This section proposes Mass-Energy Based Modeling System Tool(MEBMST)

based on the the Mass–Energy Based Economic Models[38, 39, 37]. MEBMST was developed to evaluate environmentally conscious product designs, management of manufacturing facilities to evaluate the strategies for reducing waste flows into the environment, and life cycle assesment.

Along with motivation of modeling environmental problems, the observation of similarities between physical laws of preserving material and energy and economic characteristics of physical production process, and the classic economic input-ouput analysis lead to proposing the Mass-Energy Based(MEB) economic model in [38, 39, 37]. The difference between the classic input-ouput analysis is how labor is formulated. In the MEB model, labor is formulated as an energy cost rather than as a flow of services as in classical input–output analysis. The MEB model views a production process as a sequence of transformations on the state of materials by energy. This enables one to break a large system into tractable smaller systems. Further, MEB model divides output as *useful product* and *by-product(waste)*.

6.1 Overview

The primary goals of MEBMST are as follows:

- To build a mathemtical model for processes which are used as building blocks of a plant.
- To implement a Graphical User Interface(GUI) which hides all the details of programming languages and visualizes the presentation of anal-

ysis results such as monetary factors, environmental factors given quantities of final products, and the unit cost of final products in addition to a report of the results.

A manufacturing plant is modularized with building blocks of several classes according to process flow until the desired detail description is reached. In order to contain a whole plant in a limited property of screen resource, certain blocks are described as a library which has a full description at somewhere else.

The MEBMST provides six basic kinds of process building blocks as follows:

- Production process
- Recycling process
- Storage process
- Junction process
- Goal process
- Wire object
- Library Process

6.1.1 Production Process

For example, in the model in Figure 9, the y_i represents the flow rates of materials, and x_i represents energy cost per each unit of material flow rate

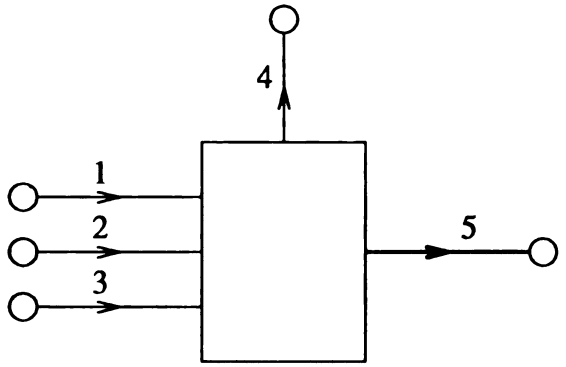


Figure 9: Abstraction of material transformation

where $i = 1, 2, \cdots, 5$. And assumptions are made that y_5 is the useful product and y_4 is the by-product.

Then the product $y_i x_i$ becomes the energy flow rate.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{bmatrix} y_5 \stackrel{\text{def}}{=} K \cdot y_5 \tag{1}$$

where the column vector

$$K = [k_1 k_2 k_3 k_4]^T$$

is called "technological coefficients of productions" following Leontief.

The law of conservation of mass requires that

$$y_1 + y_2 + y_3 - y_4 - y_5 = 0.$$

And applying the law of conservation of energy,

$$\text{output energy} + \text{input energy} + \text{processing energy} = 0.$$

or

$$x_5 y_5 = -\sum_{j=1}^{4} x_j y_j - f(y_5) y_5. \tag{2}$$

where $f(y_5)$ is the processing energy per unit of output y_5.

Substituting Equation 1 for Equation 2, we have, for $y_5 \neq 0$, the cost equation

$$x_5 = -\sum_{j=1}^{4} k_j x_j - f(y_5) \tag{3}$$

or

$$x_5 = -K^T X - f(y_5) \tag{4}$$

in vector form.

The Mass–Energy Based(MEB) Simulation tool is developed to evaluate environmentally conscious product designs, management of manufacturing facilities to evaluate the strategies for reducing waste flows into the environment, and life cycle analysis.

6.1.2 Recycling Process

Given the two choices of whether to produce the exact required input materials by recycling part of byproducts with possible leftovers and whether to recycle all amounts of byproducts and to postpone compensation of required input materials after recycling process, the latter is the philosophy behind the *recycle* class. Depending on the lack or excess of recycled products, the

difference of amount between the required input materials and recycled products may be brought from outside of the system or took out of the system with associated cost.

Considering that the goal of recycling process is to recycle all byproducts, the execution sequence of computing flow rates becomes opposite of the *production* class. Given the flow rates and the unit costs of the byproduct or a *production* class, the flow rates of the recycled products and its associated unit costs are described by

$$\begin{bmatrix} Y_o \\ Y_b \end{bmatrix} = K^T \begin{bmatrix} Y_i \\ Y_r \end{bmatrix} \tag{5}$$

and the cost of by-products and intermediate recycled products are described by

$$\begin{bmatrix} X_o \\ X_b \end{bmatrix} = K \cdot X_i - F_o(y_o) \tag{6}$$

6.1.3 Storage Process

The *Storage* and *Recycle* classes are somewhat different from other classes, while still being closely related to each other.

If the flow rate of a reprocessed end product to be recycled into a production line matches the exact flow rate requirement of a *production* process, then the system would form a perfect closed cycle system. But in reality, what if the flow rate of reprocessed materials does not match the flow rate required by a *Production* instance ? Or is it possible to design a plant which

exactly matches the amounts of needed materials ? Depending on the sufficiency or insufficiency of the recycled end product, the same kind of material - possibly with different unit prices - may need to be imported from outside a system.

To deal with such inconsistency, the *Storage* class comes to the rescue between a *Recycling* instance and a *Production* instance and behaves as a buffer between demand and service.

Given the recycled material flow Y_i, the required input material flow by a *production* class Y_o, the quantity of out-sourcing or surplus recycled material Y_b is determined by

$$Y_b = Y_o - Y_i \tag{7}$$

Similarly, given the recycled material unit cost X_i, and out-sourcing material unit cost X_b, the unit cost of input materials required by a *production* class is determined as follows:

$$X_o = \alpha \cdot X_i + \beta \cdot X_b. \tag{8}$$

where

$$\alpha = Y_o / Y_i$$

$$\beta = 1 - \alpha$$

6.1.4 Junction Process

This process is used to deliver an intermediate useful product to the next several production processes.

Let Y_p be the flow rate of intermediate products, Y_o^i, the flow rate of the next m production processes where $o \in [1, \cdots, m]$, with X_o and X_p, the unit energy cost for Y_o and Y_p respectively.

Two constraints which are met by the junction process are the continuity constraint

$$Y_p = \sum_{i=1}^{m} Y_i$$

and the compatibility constraint

$$X_o = X_p.$$

Figure 8.1 illustrates how *junction* class is used.

6.1.5 Goal Process

Before performing MEB simulation, *goals* need to be defined.

For example, "what-if" analysis of different flow rates of final products are specified as follows:

$$Y_o = \text{constant vector.}$$

6.1.6 Wire Object

This object is used to interconnect processes within a system carrying measures such as flow rate, unit engergy cost, name, and etc.

6.1.7 Library Process

Aggregation of processes is necessary in order to overcome the following problems:

- The screen size may be too small to describe a complex system.
- As processes clog together, readibility of processes deteriorates.
- Repeated modeling of frequent use of a system can be tedious, time consuming, and prone to errors.

A system is defined in terms of above processes. By defining the *Library Process* as a system recursively, system can be described hierarchically and structurally. The *library* class process is rather a simulation directive which manages what processes are to be simulated next. The larger a system becomes so is the degree of cluttering in a limited screen space. The *library* class is introduced as a building block to model a system which is too large to accomodate in the limited space of a drawing screen.

By its capability of encapsulation of complex processes with a simple representative object and its capability of expansion of a *Library* class instance into the full blown description of the system, not only does this class enhance readability but also it allows a user to choose the degree of detail description of the system.

As a *library* instance can have other *library* instances, a complicated system can be organized in a tree structure allowing a user the freedom of

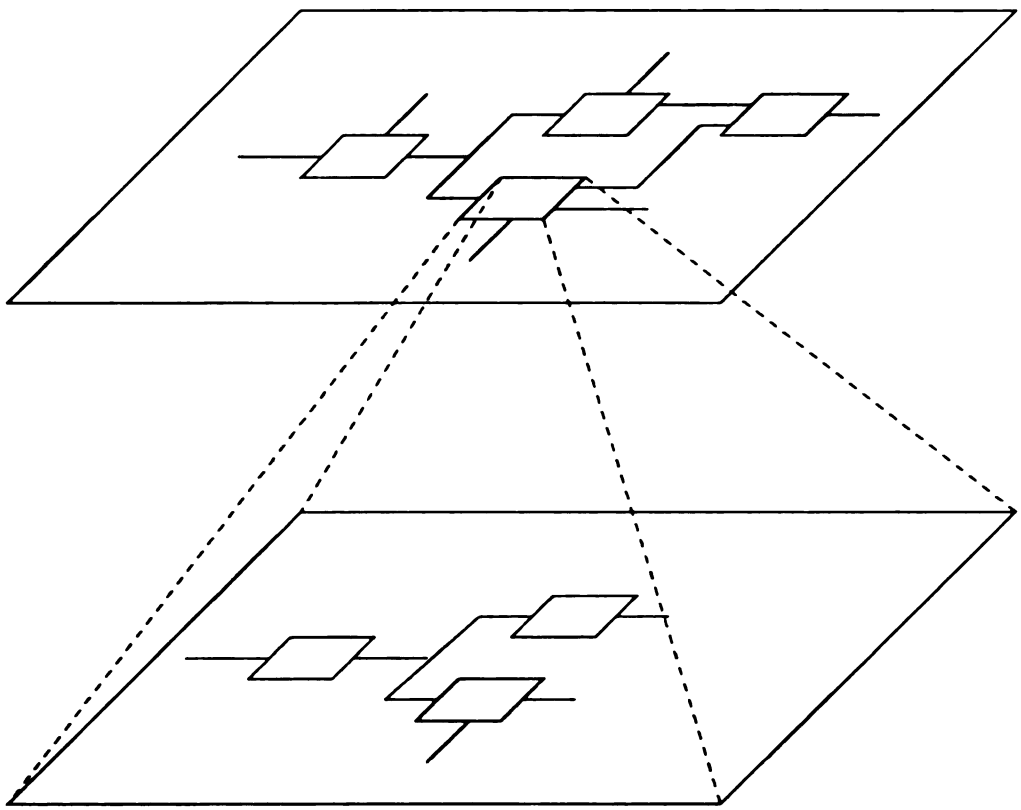


Figure 10: Multi layered tree structured description of a system

viewing any detailed level of a system as in Figure 10. And the reusability of a proven *library* class helps modeling with confidence and saves time.

A *library* class is created by adding its representative encapsulation shape to the existing system model. In normal classes other than the *library* class, the property of incident wire is determined by the neighboring process context. However, that is not the case for the *library* class.

The communication of one level of a system with the next level of a system occurs through the specific wiring of the next level system. Thus the encapsulation procedure involves defining the next level system boundary

and defining the location of points incident upon a library class. This correspondence in the interconnection of the *library* class and another class is determined by the incidence points alongside the enscapsulation shape. A *library* class process implicitly has a fixed number of incident points associated with *library* a labeled internal wire, besides the representative encapsulation shape.

By doing so, it is possible to to map the external wire into a certain internal wire in the next level system. The graphical user inteface(GUI) for this process is described in Appendix A. As an illustration, the system in Figure 8.1 can be simplified by creating a new *library* class `jj` in Figure 11.

The *Production Process* block, which describes a transformation process is based on the Mass Energy Based Economic Models[37]. The first part of the model, *transformation process*, has been explained in Section 6.1.1.

The production material flow equation Eq. 1 and the energy equation Eq. 3 in Sectionsec:wmeb can be partitioned and generalized as in Eq. 9 and Eq. 10 to view a system as a component at the next level as in Figure 11.

$$\begin{bmatrix} y_l \\ y_r \end{bmatrix} = \begin{bmatrix} K_{lb} & K_{lo} \\ K_{rb} & K_{ro} \end{bmatrix} \begin{bmatrix} y_b \\ y_o \end{bmatrix} \stackrel{\text{def}}{=} K \begin{bmatrix} y_b \\ y_o \end{bmatrix} \tag{9}$$

where y_l refers to the required input material flow supplied to the processes within a system boundary, y_b to intermediate products produced inside a system boundary, and y_r to the material flow response variables outside a system boundary which include both input supplies and production of byproduct or waste.

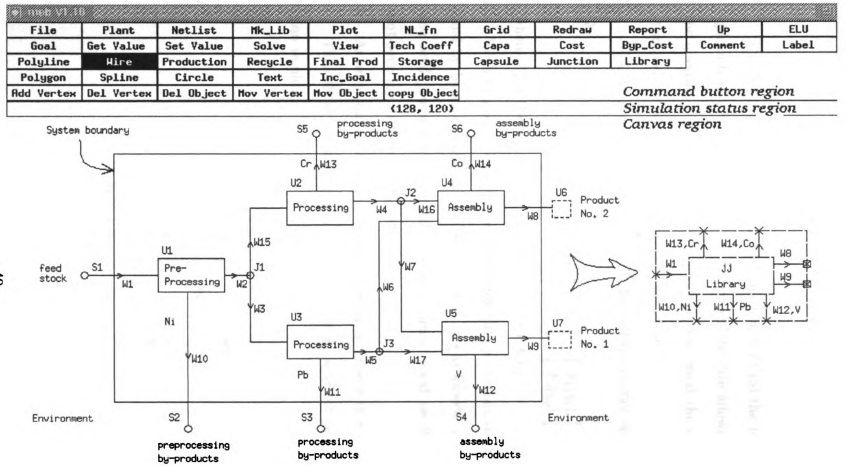


Figure 11: System as a component at the next level

The K, characterizing the transformation, is called the *technological coefficients* of production. This goal-driven architecture allows one to answer the question such as "given a final product flow, what the material flow of intermediate products and byproducts would be?"

And the second part of the model, *generalized energy equation*, is given by:

$$\begin{bmatrix} X_b \\ X_o \end{bmatrix} = \begin{bmatrix} K_{lb}^T & K_{rb}^T \\ K_{lo}^T & K_{ro}^T \end{bmatrix} \begin{bmatrix} X_l \\ X_r \end{bmatrix} - \begin{bmatrix} F_b(y_b) \\ F_o(y_o) \end{bmatrix} \tag{10}$$

where X_i is the cost related with y_i for $i \in \{l, r, b, o\}$.

With this equation, every material flow rate is computed to meet a final goal by back-propagation. Once every flow rate is known, the unit cost of the final product is computed in reverse order based on the unit cost of out-sourcing material unit costs.

As an example, consider a manufacturing enterprise in Figure 12.

Then those material flow equation and the generalized energy cost equation would be

$$\begin{aligned} y_l &= \begin{bmatrix} y_{31} \\ y_{41} \\ y_{32} \\ y_{42} \\ y_{53} \\ y_{54} \end{bmatrix}, \quad y_r = \begin{bmatrix} y_{r1} \\ y_{r2} \\ y_{r3} \\ y_{r4} \\ y_{r5} \\ y_{r6} \end{bmatrix} \\ y_b &= \begin{bmatrix} y_{o3} \\ y_{o4} \\ y_{o5} \end{bmatrix}, \quad y_o = \begin{bmatrix} y_{o1} \\ y_{o2} \end{bmatrix} \end{aligned} \tag{11}$$

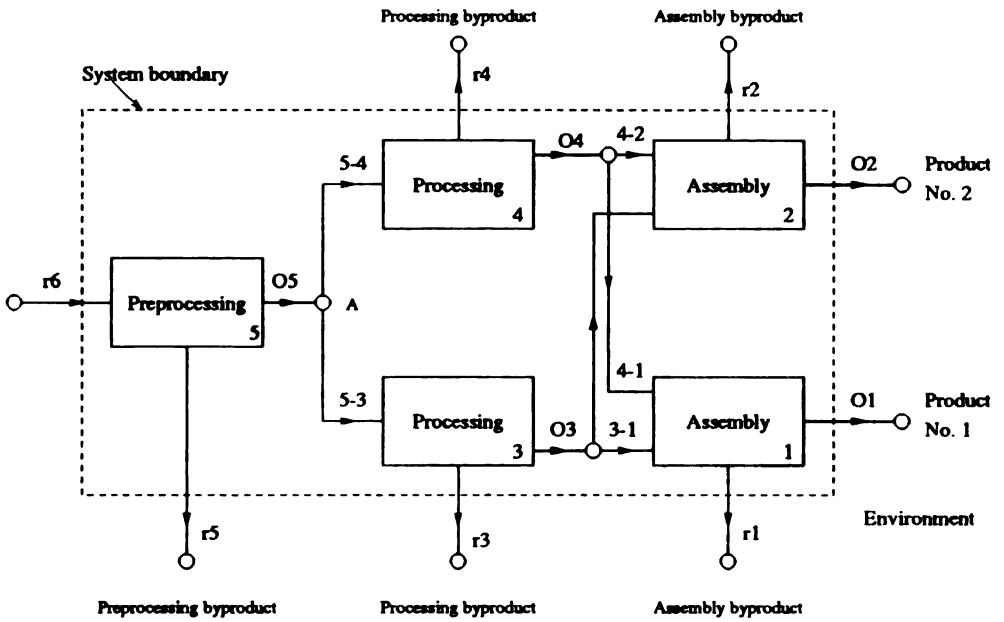


Figure 12: Illustrative process network diagram for a manufacturing enterprise

By the continuity constraints imposed by the *junction process*,

$$\begin{aligned} & y_{o5} - y_{54} - y_{53} = 0 \\ & y_{o3} - y_{31} - y_{32} = 0 \\ & y_{o4} - y_{42} - y_{41} = 0 \\ & \Rightarrow y_b = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} y_l \\ & \stackrel{\text{def}}{=} A y_l \end{aligned} \tag{12}$$

With the substitution of (12) for y_b in (9), the requirement of outside material and its associated cost is described in terms of the final product:

$$\begin{aligned} y_b &= (I - AK_{lb})^{-1} AK_{lo} \\ &\stackrel{\text{def}}{=} K_b y_o \\ y_r &= (K_{rb}(I - AK_{rb})^{-1} AK_{lo} + K_{ro}) y_o \end{aligned}$$

$$\begin{aligned} &\stackrel{\text{def}}{=} (K_{rb}K_b + K_{ro})y_o \\ &\stackrel{\text{def}}{=} K_s y_o \\ X_b &= -(I - AK_{lb}^T)^{-1}K_{rb}X_r - (I - AK_{lb}^T)^{-1}F_b(y_b) \end{aligned}$$

The system model given in Eq. 13 quantitatively establishes the relationship between the flow rates of the raw materials entering the system boundary and the products leaving the boundary along with the wastes released into the environment of the system. Furthermore, the model represents these flow rates as an explicit function of the process technologies incorporated within the system boundary. This provides us with the powerful method of handling complex system without losing logical or physical consistency [37].

Consider the extended system model as in Figure 13.

$$\begin{aligned} y_{fl} &= \begin{bmatrix} y_{67} \\ y_{78} \\ y_{82} \end{bmatrix}, \quad y_{fr} = \begin{bmatrix} y_{r7} \\ y_{r8} \\ y_{r9} \end{bmatrix} \\ y_{fb} &= \begin{bmatrix} y_{I6} \\ y_{I7_a} \\ y_{I7_b} \\ y_{I8} \\ y_{O8} \end{bmatrix}, \quad y_{fo} = \begin{bmatrix} y_{o3} \end{bmatrix} \end{aligned} \tag{13}$$

The technical coefficients of the *storage class* connecting i-th *production class* and the j-th *recycling class*, is determined by subtracting j-th row from i-th row of K.

$$\begin{bmatrix} Y_{fb} \\ Y_{fo} \end{bmatrix} = K \begin{bmatrix} Y_{fl} \\ Y_{fr} \end{bmatrix} \tag{14}$$

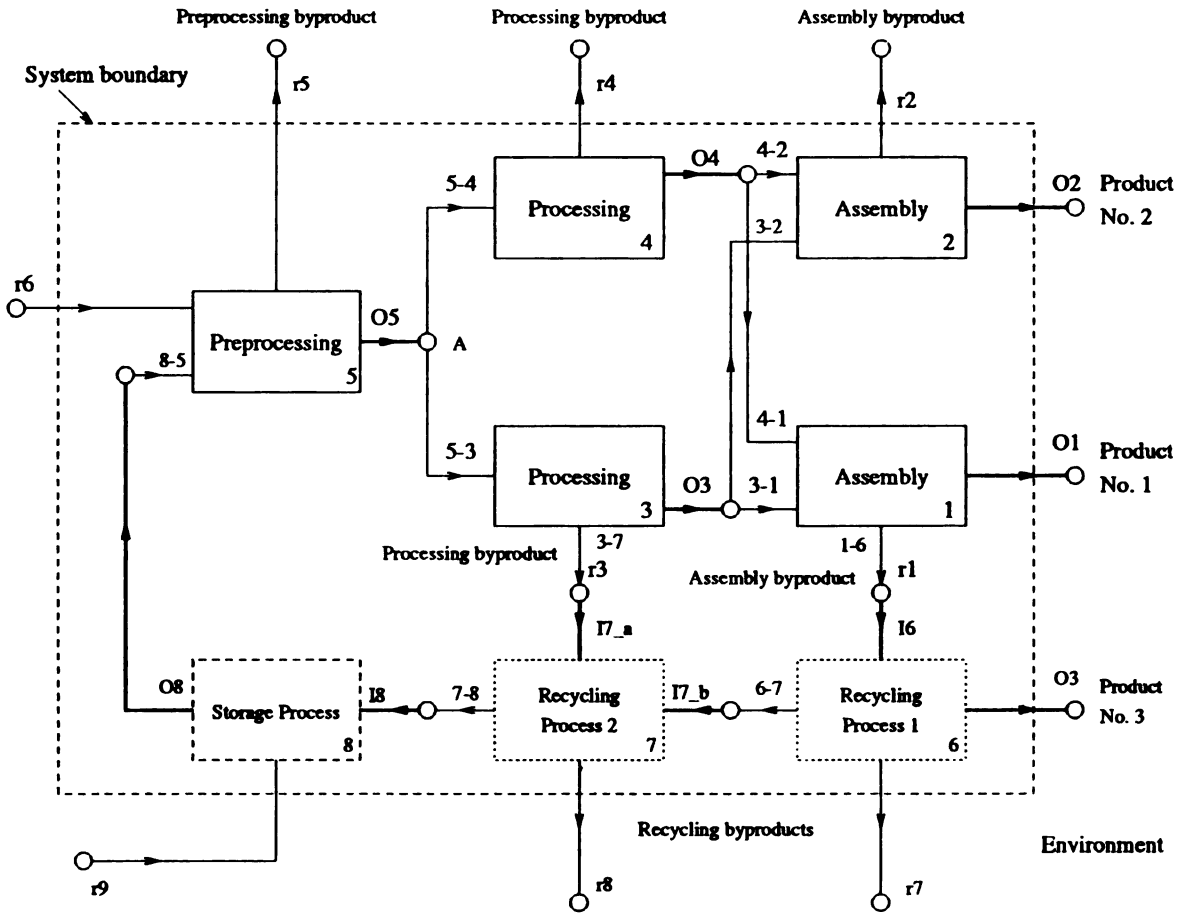


Figure 13: Extended process network diagram for a manufacturing enterprise

and the cost of byproducts, intermediate, or final products from the recycling processes are described by

$$\begin{bmatrix} X_{fb} \\ X_{fo} \end{bmatrix} = KC \begin{bmatrix} X_{flb} \\ X_{flo} \end{bmatrix} - \begin{bmatrix} F_b(y_{fb}) \\ F_o(y_{fo}) \end{bmatrix}. \tag{15}$$

Then overall extended system model could be constructed by augmenting the original system model with the extended system model as follows:

$$\begin{bmatrix} Y_l \\ Y_r \\ Y_{fb} \\ Y_{fo} \end{bmatrix} = \begin{bmatrix} K_{lb} & K_{lo} & * & \\ K_{rb} & K_{ro} & * & \\ * & & K_{fbl} & K_{fol} \\ * & & K_{fol} & K_{for} \end{bmatrix} \begin{bmatrix} Y_b \\ Y_o \\ Y_{fl} \\ Y_{fr} \end{bmatrix} \tag{16}$$

and the cost of byproducts, intermediate, or final products from the recycling processes are described by

$$\begin{bmatrix} X_b \\ X_o \\ X_{flb} \\ X_{flo} \end{bmatrix} = \begin{bmatrix} K_{lb}^T & K_{rb}^T & * & \\ K_{lo}^T & K_{ro}^T & * & \\ * & & KC_{fbl} & KC_{fbr} \\ * & & KC_{fol} & KC_{for} \end{bmatrix} \begin{bmatrix} X_l \\ X_r \\ X_{fl} \\ X_{fr} \end{bmatrix} - \begin{bmatrix} F_b(y_b) \\ F_o(y_o) \\ F_{fl}(y_{fl}) \\ F_{fr}(y_{fr}) \end{bmatrix}. \tag{17}$$

7 MEB NETWORK

The proposed Mass-Energy Based(MEB) System Modeling tool is composed of several main components: *Process and Network Representation*, *Environmental Load Unit Database*, *Goal Definition and System Environment Setup*, *MEB Language(MEBL) Execution Unit*, and *Data Visualization Unit.*

There are two extreme cases of how simulation can be accomplished. One approach is to design a simulation with a textual simulation language which can handle what general purpose language does. After all, it is the machine codified behavioral descriptions which make simulation possible by computer. And the capability of a simulation in fine detail is only limited by the capability of simulation language.

However, the disadvantage becomes obvious when a system grows larger. The larger the system grows, so does the size of the codified simulation program. Even the author of a simulation design is likely to become confused about what are the boundaries of subprocesses and how they interact together, as time goes by. Adding to that, the learning of a new simulation language may take long time. And the cost of initial learning, retaining that learning of the simulation language can be high.

At the other side of the first apporach stands the graphical representation of a system. Describing a system by only graphical objects greatly enhances the readibility of a system, helps the reader to grasp overview of the whole system, and to understand interactions between subprocesses more easily.

In reality, both methods should be mixed appropriately and manageably in performing simulation, as does the design process.

Despite the ease and other advantages of graphic modeling of a system, the graphic objects are not as flexible as simulation language itself. And there is some system behavior which can be described only by simulation language itself.

By embedding the conversion from visual semantics to simulation language into the MEB modeling, the learning time of using a new simulation tool is greatly reduced. I would like to recommend that the first approach of textual description to be confined in a process and the second approach of graphical description be used in defining system boundaries and interactions to accomplish a simulation.

From this discussion, I would like to assert that graphical modeling cannot replace simulation language itself, even though the reverse is true.

The proposed Mass-Energy Based System Modeling tool, combining both approaches, begins with dividing a large system into managegeable subprocesses with MEB building blocks and wires as shown in Figure 14.

This tool allows one to input the description of the main structure of a plant using a drawing pallet available to the user. This pallet contains built-in drawing buttons in a graphic user interface(GUI) implemented on the X-Window environment. The GUI relieves the user from knowing all the mathematical details of the models which describe each process within a plant and the interconnection constraints associated with the structure of

the plant or process.

Besides having features which represent network information succinctly, MEBS also introduces the Mass-Energy Based Simulation Language(MEBL) which borrows many aspects from C language, MATLAB[1], and SQL database language. A source program is automatically created by the user with the GUI. I will describe some of the details of the program with an example.

7.1 MEB Graph Representation

The process network is a mirrored acyclic data flow graph even though processes contains feedback loops.

A network is described with several types of building blocks such as `production`, `junction`, `library`, `goal` blocks, and wires which connect the blocks together. The forward connections are done by all the types of blocks except the `recycle` type block while backward feedback connection uses only the `recycle` type block as a subprocess. The special `storage` type block is used when the backward connection feeds to a forward connected process block to form a feedback connection.

The various values of intermediate products or byproducts are propagated through wires. The entities of wire, which might be a final product, byproduct, or intermediate product, propagate through wire communicating bidirectionally. And these wires determine preset nodes which current node is dependent on and a postset nodes which are to be exected next.

[1]MATLAB is a trademark of Math Works Inc.

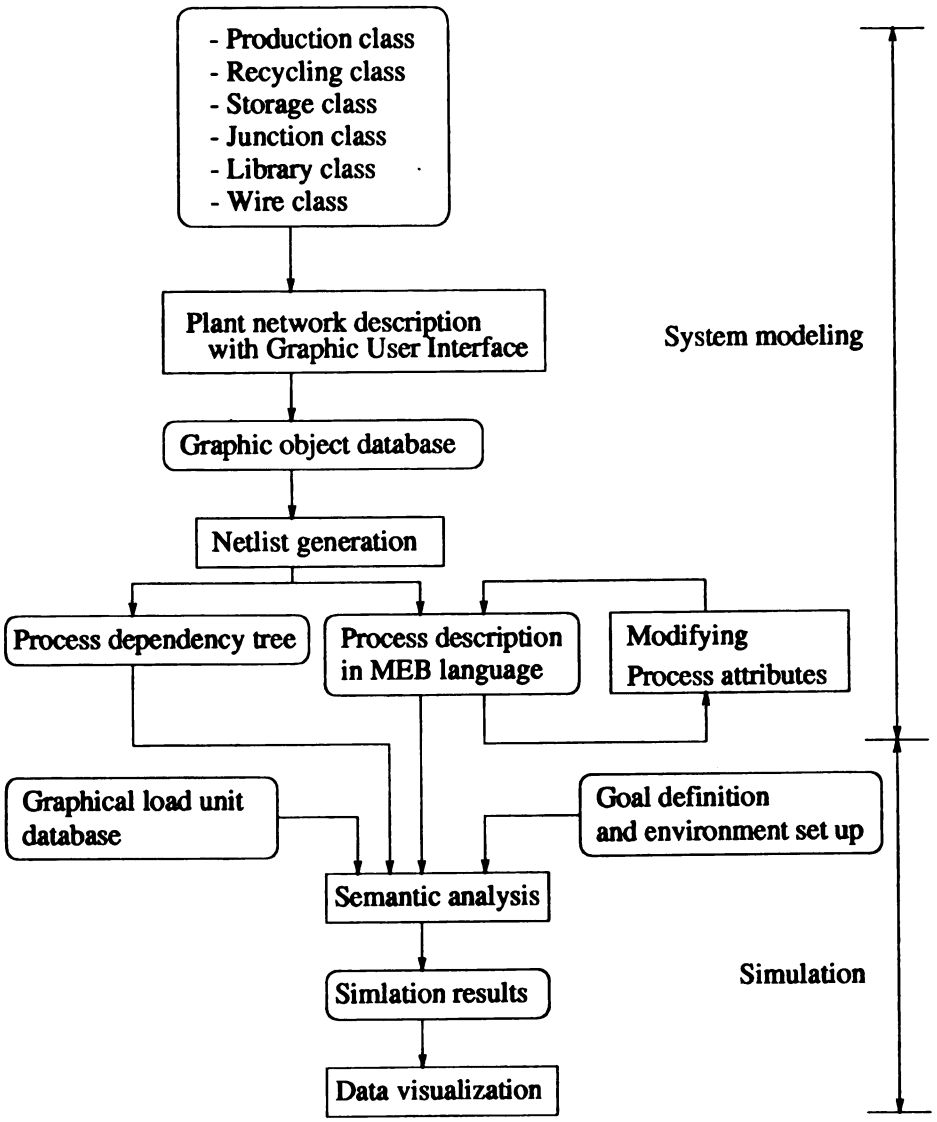


Figure 14: MEB simulation architecture

Considering that wire attributes might be changed only through execution of process modeling, this process execution becomes a transition to change a state of a process into the next state and to move on to other dependent processes to do the same.

Definition 1 *A process network, G, is a seven-tuple graph* $G = (V, T, W, \Delta, \delta, \tau, \beta)$; $V = \{v_1, v_2, \ldots, v_k\}$ *where* $k = |V|$, *is a finite set of processes shaped as rectangulars, and the attributes of processes are extended by five different classes of* {*PRODUCTION, JUNCTION, RECYCLE, STORAGE, and LIBRARY*}; $T = \{t_1, t_2, \ldots, t_m\}$ *where* $m = |T|$, *is a finite set of terminal nodes shaped as small circles, and the attribute of T are extended by two different types of* {*GOAL, SIGNAL*}; $W = \{w_1, w_2, \ldots, w_a\}$ *where* $a = |W|$, *is a finite set of ordered pair of different nodes such that* $w_i = (v_s, v_d)$, $s \neq d$ *where* w_i *is a wire from a node* v_s *to a node* v_d *represented by an arrow;* Δ *and* δ *is a top-down and bottom-up hook-up function mapping from W to W;* $\tau \subset V$ *and* $\beta \subset V$ *are initial set of nodes at which top-down or bottom-up execution sequences are to be originated.*

7.2 Inputs and Outputs

Wire,W, which connect processes together into a system is extended by its attributes as follows:

$$W = (Label, Capa, Cost)$$

Label Entity name; set of alphabets

Capa Flow rate of entities

Cost Unit cost of entities

Since Δ and δ are function of W, the distinction of inputs and outputs are necessary. Given a process with neighbor wires on it, the wires are fall into one of a catetory, input or output. The decision of input and output of a wire, w_i, is based on PNT mathematical model mentioned in Section 6. In PNT theory, wire has a context sensitive meaning in terms of their direction, types of its associated processes, and the direction of execution sequence i.e. top-down or bottom-up execution environment. Those resulting constraints to form a sensitive meaning of wire is described in Table 4 and Table 5 along with the constraint such that the number of inbound wires of a *JUNCTION*, *STORAGE*, and *GOODS* should be one.

This context sensitive meaning of a wire is used in several ways as follows:

1. To determine whether connection of a certain nature should be allowed.
2. To construct a set of functions through wire examination whether it is a stimulus or a response variable to form a function.
3. To determine the next execution sequence.

The pair of values in Tables 4 and 5 are used to construct the top-down function and the bottom-up function respectively. The meaning of the numbers are as follows:

	SIGNAL	JUNC.	PROD.	RECYCLE	GOODS	STORAGE	LIBRARY
SIGNAL	(-1,-1)	(3,3)	(3,3)	(3,3)	(3,3)	(3,3)	(-1,-1)
JUNCTION	(-1,-1)	(1,0)	(1,0)	(-1,-1)	(1,0)	(1,0)	(-1,-1))
PRODUCTION	(0,2)	(1,0)	(1,0)	(0,2)	(1,0)	(0,2)	(-1,-1)
RECYCLE	(0,0)	(-1,-1)	(-1,-1)	(0,0)	(-1,-1)	(0,0)	(-1,-1)
GOODS	(-1,-1)	(-1,-1)	(-1,0)	(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)
STORAGE	(-1,-1)	(1,0)	(1,0)	(-1,-1)	(1,0)	(-1,-1)	(-1,-1)
LIBRARY	(0,3)	(0,3)	(0,3)	(0,3)	(0,3)	(0,3)	(1,0)

Table 4: Inbound wire;$N_{row} \times N_{column}$ constraints in tabular form

	SIGNAL	JUNC.	PROD.	RECYCLE	GOODS	STORAGE	LIBRARY
SIGNAL	(-1,-1)	(0,1)	(0,1)	(1,1)	(-1,-1)	(-1,-1)	(0,1)
JUNCTION	(0,1)	(0,1)	(0,1)	(-1,-1)	(3,3)	(0,1)	(0,1))
PRODUCTION	(0,1)	(0,1)	(0,1)	(1,1)	(3,3)	(0,1)	(1,0))
RECYCLE	(0,1)	(-1,-1)	(1,1)	(1,1)	(-1,-1)	(-1,-1)	(1,1))
GOODS	(3,3)	(3,3)	(3,3)	(-1,-1)	(-1,-1)	(3,3)	(-1,-1))
STORAGE	(0,1)	(-1,-1)	(0,1)	(1,1)	(-1,-1)	(-1,-1)	(0,1))
LIBRARY	(0,1)	(-1,-1)	(-1,-1)	(1,1)	(-1,-1)	(0,1)	(0,1))

Table 5: Outbound wire; $N_{row} \times N_{column}$ constraints in tabular form

0 The wire is a dependent variable.

1 The wire is an independent variable.

2 The wire is an independent variable, but the cost of the wire needs to be initialized first in a process as part of the initialization function.

3 Connections are allowed, but the process have an empty function.

8 MEB Language(MEBL)

The graph model is a very good start to model a big system without delving into a mire of language quirks, and is quite a useful conceptual global map to show the extent of detail interactions between different processes and the data flow between processes.

Still, the mathematical modeling of each process is often based on a textual description for its brevity, simplicity, and flexibility. After all, the computer only understands numbers and alphabets at its core. Thus, every aspect of system modeling should be translated into a textual description describing how each process should evaluate and how each process should interact with one another. However, a computer code is not a good tool for designing a program, nor is it a good communication language for people[13].

In this context, our language MEBL is to be described in two parts, i.e. evaluation of processes and its execution environment.

The MEBL expression for evaulation uses a similar model of C language, MATLAB, and SQL.

In the next section how a system can be described in textual MEBL will be presented. The exact syntax and grammar of MEBL are presented in Appendix H.

8.1 System Structure

The system structure described in MEBL consists of two types of system descriptions; they are declarations and the series of process description blocks

some of which might be directly related to wire variables and some of which are related to execution environment for initialization or set-up for simulation or post–processing of simulation results.

The keywords to differentiate such types are as follows:

`wire`	Define a set of entities conveyed by wire
`netlist`	Build MEB graph and translate MEB graph into MEBL
`init`	Assign *name* entity of wire instances with label
`post`	Calculate overall environmental impact
`junction`	Specify *junction* process based on MEB theory
`signal`	Specify terminal node process
`block`	Specify process description

The first declaration is about defining entities of wire which become perfomance measures based on MEB theory described as

```
wire {
        capa;
        cost;
        name;
}
```

A wire class is a composite object consisting of flow rate *capa*, energy cost *cost*, and a label *name* of wire itself. A wire instance name followed by a period and one of three entity names is the way of referring to the individual wire data object.

8.2 Netlist

Again considering the above example, the `netlist` body consists of the list of the wires with its source node to the left and its destination node to the right. For example,

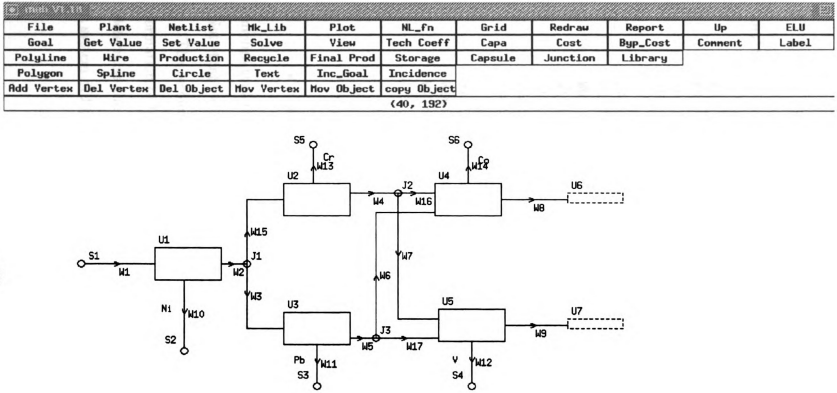


Figure 15: A system with junction between processes

```
netlist {
        W1: S1 -> U1;
        W2: U1 -> J1;
        W3: J1 -> U3;
        W4: U2 -> J2;
        W5: U3 -> J3;
        W6: J3 -> U4;
        W7: J2 -> U5;
        W8: U4 -> U6;
        W9: U5 -> U7;
        W10: U1 -> S2;
        W11: U3 -> S3;
        W12: U5 -> S4;
        W13: U2 -> S5;
        W14: U4 -> S6;
        W15: J1 -> U2;
        W16: J2 -> U4;
        W17: J3 -> U5;
}
```

Figure 16: Automatically generated netlist in MEBL

```
W1: S1 -> J1;
```

implies that the wire W1 goes from the signal node S1 to junction node J1. Note that -> has a different meaning from the meaning of the C language.

The second declaration consists of lists defining all the MEB graph information of how processes are connected together. This netlist will partly determine the sequence of process execution in simulation. For example, in a system of Figure 8.1, the *netlist* delclaration will look like

Block type	Subclass No.
Production	0
Recycle	1
Goods	2
Storage	3
Libray	4

Table 6: Block node subclass number used in MEBL

The remaining part of MEBL describes either the process itself or initialization before simulation begins and post-processing after the completion of simulation.

A `block` node represents a subprocess and contains appropriate statements in its body. Accordingly, there are two `block` nodes corresponding to two subprocesses in this example. `init` node is a special kind of block node which is done first before execution of functions associated with each block. The subclass number of the block used in MEBL statement as in "`block` subclass_number *block_name* { $\cdots$ }" is shown in Table 6.

Table 6 shows five kinds of process types used to build a system.

The automatically generated example of an MEBL system description corresponding to Figure 8.1 would be translated as:

```
junction J1 {
        shape     J1;
        out       W15;
        in        W2;
        out       W3;

        k = [ 1, 1; ];

        if (backward) {
                [ W2.capa; ] = k * [ W15.capa; W3.capa; ];
```

```
        } else {
                [ W15.cost; W3.cost; ] = k' * [ W2.cost; ];
        }
}

junction J2 {
        shape     J2;
        out       W16;
        in        W4;
        out       W7;

        k = [ 1, 1; ];

        if (backward) {
                [ W4.capa; ] = k * [ W16.capa; W7.capa; ];
        } else {
                [ W16.cost; W7.cost; ] = k' * [ W4.cost; ];
        }
}

junction J3 {
        shape     J3;
        out       W17;
        in        W5;
        out       W6;

        k = [ 1, 1; ];

        if (backward) {
                [ W5.capa; ] = k * [ W17.capa; W6.capa; ];
        } else {
                [ W17.cost; W6.cost; ] = k' * [ W5.cost; ];
        }
}

signal S1 {
        shape     none;
        out       W1;
}

signal S2 {
        shape     none;
        in        W10;
}
```

```
signal S3 {
        shape     none;
        in        W11;
}

signal S4 {
        shape     none;
        in        W12;
}

signal S5 {
        shape     none;
        in        W13;
}

signal S6 {
        shape     none;
        in        W14;
}

block 0 U1 {
        shape     U1;
        in        W1;
        out       W10;
        out       W2;

        W10.cost = zeros(size(W2.capa));

        k = [ 1; 1; ];

        if (backward) {
                [ W1.capa; W10.capa; ] = k * [ W2.capa; ];
        } else {
                [ W2.cost; ] = k' * [ W1.cost; W10.cost; ];
        }
}

block 0 U2 {
        shape     U2;


        out       W13;
        in        W15;
        out       W4;
```

```
        W13.cost = zeros(size(W4.capa));

        k = [ 1; 1; ];

        if (backward) {
                [ W13.capa; W15.capa; ] = k * [ W4.capa; ];
        } else {
                [ W4.cost; ] = k' * [ W13.cost; W15.cost; ];
        }
}

block 0 U3 {
        shape     U3;
        out       W11;
        in        W3;
        out       W5;

        W11.cost = zeros(size(W5.capa));

        k = [ 1; 1; ];

        if (backward) {
                [ W11.capa; W3.capa; ] = k * [ W5.capa; ];
        } else {
                [ W5.cost; ] = k' * [ W11.cost; W3.cost; ];
        }
}

block 0 U4 {
        shape     U4;
        out       W14;
        in        W16;
        in        W6;
        out       W8;

        W14.cost = zeros(size(W8.capa));

        k = [ 1; 1; 1; ];

        if (backward) {
                [ W14.capa; W16.capa; W6.capa; ] = k * [ W8.capa; ];
        } else {
                [ W8.cost; ] = k' * [ W14.cost; W16.cost; W6.cost; ];
        }
}
```

```
block 0 U5 {
        shape     U5;
        out       W12;
        in        W17;
        in        W7;
        out       W9;

        W12.cost = zeros(size(W9.capa));

        k = [ 1; 1; 1; ];

        if (backward) {
                [ W12.capa; W17.capa; W7.capa; ] = k * [ W9.capa; ];
        } else {
                [ W9.cost; ] = k' * [ W12.cost; W17.cost; W7.cost; ];
        }
}

block 2 U6 {
        shape     U6;
        in        W8;
}

block 2 U7 {
        shape     U7;
        in        W9;
}
```

The `junction` node abides by special constraints. In the framework of the MEB theory, the sum of output flow rates are the same as the input flow rate and the unit costs are the same for all wires.

A `signal` node is created automatically either at the beginning or at the end of a wire which is not connected to any other process. Normally, there is nothing to compute in the `signal` body except the initial cost provided in the phase of environmental setup to initiate simulation.

Finally, the special **post** body which may or may not be executed is available to evaluate environmental impact along with environmental index database as in Section 8.6 after all other processes have completed their simulation.

General building block classes as in Table 6 are specified by different shape of geometrics in drawing. For the sake of readability, the declaration part associates a specific block class with a graphic object in a drawing and indicates which wires are coming in and going out of a process. Then, a specific MEB evaulation statement of a block class follows the declaration part.

It is worth mentioning that any **wire** variables can be accessed in any nodes. However, considering that the messages coming in and going out of a process are highly correlated with the network of processes, it seems to be a good practice to access only the **wire** variables which are in contact with the node. With this limitation of choice in the practice, it helps the program to be modular and structured.

8.3 Variables

There are two execution environmental reserved read-only global variables such as **backward** and **forward**. Simulation goes through the phases of backward and forward computation; in other words, top–down or bottom–up computation.

Those special variables shows the direction of traverse during simulation

so that each process can define its own segment of a program in MEBL inside the same block depending on the state of traversal. Users can read those values but users are not allowed to set the values of those special variables.

8.4 Constants

To accommodate the frequently used symbol π in trigonometry function, a specific symbol `PI` is reserved for a constant. And the character constant is a character between single quoatation marks while the string constant like `"this is string"` is a list of characters within double quotation marks. Both the character and the string constants are accepted as in C language. Some invisible characters are represented by escaping as follows:

`\n`	newline
`\t`	tab
`\f`	form feed
`\\`	back slash

For the numerical representation, both decimal and hexa-representation are accepted for an integer value. For example, the decimal number `10` is equal to the hexa-number `0xa`. For the floating number representation, only decimal numbers are allowed, as in the following examples:

$$.1234 \quad 1.234 \quad 12.34E5 \quad 123.4e-5$$

8.5 Control Flow

The control flow is similar to that of the C language except that the expression body between control flow keywords to be selected or iterated should

be enclosed by { and } even though the expression body contains only one statement.

if else These non-iterative conditional selective control keywords associate exclusive statements to be executed with dynamic expressions while simulation is going on. As a result, this control flow selects a specific part of the body separated by `if` and `else` keywords for computation.

Example)

```
if (expr1) {
        statements;
} else if (expr2) {
        statements;
        :
} else (expr3) {
        statements;
}
```

Every expression body between `if` and `else` should be within { and }, even though the body has only one line statement.

for This iterative conditional control key word is a very concise iterative statement especially if initial statement before the iterative body or/and the post statement after the iterative body is/are necessary.

Example)

```
for (expr1; expr2; expr3) {
        statements;
}
```

expr1 is the initial condition before any other iterative statements of `for` is considered for computation. If *expr2* is true, the main body within { and } is computed followed by computation of *expr3* and *expr2* to make a full cycle again for next iteration. Otherwise, the `for` statement is completed without computation of the main iterative statements.

while The statements in the `while` body are computed repeatedly as long as the *expr1* is true before the execution of the `while` body. This is another iterative selective control flow simpler than the `for` key word.

Example)

```
while (expr1) {
      statements;
}
```

As a note the control flow `do ... while` in C language is not available.

break This control flow key word terminates the innermost enclosing loop by `for` and `while`.

continue returns the next computation program pointer immediately to the innermost enclosing `while` or `for` control statement.

goto `goto` *identifier;* renders the next computation to be the statement of the label *identifier*. An identifier followed by : is considered as an address label within the scope of a process.

Example)

```
goto label1;
```

8.6 Database

MEBL has a statement similar to Structured Query Language(SQL) for simple database manipulation. And for the compatibility with other databases, the database table can be managed by a normal text editor because it contains only the plain ASCII text.

Four keywords used in database file are:

version requires only one argument telling a version number.

delim describes the delimiter between fields.

field requires three arguments. The first argument is the field name and the second argument describes the type of field. There are only two types; *c* implies the field is character type and *f* implies the type of floating number. The third argument is the maximum field size.

Record delimiter is set to the new line character.

end implies the end of head information.

To understand the database structure an example of Environmental Loading Unit(ELU) database file is shown as follows:

The syntax for the database statement is of the form:

```
version   1
delim             ,
field             material    c         20
field             ec          c         4
field             elu         f         10
end

Co,               RM,         76
Cr,               RM,         8.8
Fe,               RM,         0.09
Mn,               RM,         0.97
Mo,               RM,         1.5E3

CO2,              EA,         0.09
CO,               EA,         0.27
SOx,              EA,         0.10
CFC-11,           EA,         300
CH4,              EA,         1.0

Nitrogen,         EW,         0.1
Phosphorus,       EW,         0.3
```

Figure 17: Example of a database file "*elu.db*" to show database structure

`select` *field_name* `from` *database* `where` *query*

The *database* uses the name omitting the suffix `.db` from the database file name.

To specify a field *material* in the database *elu*,

`elu#material`

is allowed in the *query* statement.

8.7 Expression

8.7.1 Matrix

The elements of the matrix are either separated by a comma or a semicolon. A semicolon is for the change to the next row of a matrix while a comma is for delimiting elements column-wise. For example, `a = [ 1, 2, 3; 4, 5, 6] ;` implies two by three matrix.

Larger matrices can be generated by using variables as shown in the following example:

```
a = [1, 2; 3, 4];
b = [5; 6];
c = [7, 8, 9];
```

A 3x3 matrix d is generated by using a, b, c as follows:

```
d = [a, b; c];
```

will construct d matrix to be three by three matrix resulting in

$$\begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

8.7.2 Vector

Two or three elements are needed to represent a range of values as follows:

`[expr1: expr2]` or `[expr1: expr2: expr3]`.

The first element **expr1** is the value to start from and the second value **expr2** is the final value of a vector. The third element determines the step size for the next element to generate. If the third element is missing, one is used for the default step size. For example `[0:10:2]` will generate a vector [0, 2, 4, 6, 8, 10].

8.8 Operators

Operators and their precedences are shown in Table 7.

8.9 Output Functions

prval(a) Displays the value of a in the message window.

plot(x, y) Draws x-y plot on a pop-up window.

title(s) Sets the title message of a pop-up drawing window to s.

8.10 Math Functions

det(a) Determinant of the square matrix a.

inv(a) Inverse of the square matrix a.

`+a`	Positive of a
`-a`	Negative of a
`!a`	Negation of a
`a'`	Transpose of a matrix a
`a * b`	Multiplication of a and b
`a / b`	Division of a by b
`a .* b`	Element-wise multiplication
`a ./ b`	Element-wise division
`a + b`	Sum of a and b
`a - b`	Subtraction of a by b
`a > b`	Greater than
`a >= b`	Greater than or equal to
`a < b`	Less than
`a <= b`	Less than or equal to
`a == b`	Equal to
`a != b`	Not equal to
`a && b`	a and b
`a || b`	a or b
`a = b`	Assignment

Table 7: Precedences of operators

diag(a) If a is a matrix, diag(a) is the main diagonal matrix. Or if a is a vector diag(a), creates a square matrix with the diagonal elements the same as a and off-diagonal elements zeros.

size(a) Returns the number of rows and the number of columns.

eye(a) Returns an identity matrix with the same size of a.

zeros(a) Returns a matrix of the same size of a with its elements zeros.

ones(a) Returns a matrix of the same size of a with its elements ones.

exp(a) Returns a matrix of the same size of a with its elements exponential of the elements of a.

ln(a) Returns a matrix of the same size of a with its elements natural logarithms of the elements of a.

log(a) Returns a matrix of the same size of a with its elements base ten logarithms of the elements of a.

cos(a) Returns a matrix of the same size of a with its elements cosine of the elements of a.

sin(a) Returns a matrix of the same size of a with its elements sine of the elements of a.

tan(a) Returns a matrix of the same size of a with its elements tangent of the elements of a.

acos(a) Returns a matrix of the same size of a with its elements inverse cosine of the elements of a.

asin(a) Returns a matrix of the same size of a with its elements inverse sine of the elements of a.

atan(a) Returns a matrix of the same size of a with its elements inverse tangent of the elements of a.

cosh(a) Returns a matrix of the same size of a with its elements hyperbolic cosine of the elements of a.

sinh(a) Returns a matrix of the same size of a with its elements hyperbolic sine of the elements of a.

tanh(a) Returns a matrix of the same size of a with its elements hyberbolic tangent of the elements of a.

9 Process Network Execution Model

A process network graph also can be represented by Petri Net, a bipartite graph, to aid a visual communication among processes similar to flowgraphs with concurrent processes capability.

Recalling the nature of a system, it has separate processes or components some of which interact independently and some of which have to wait until the required entites are available to begin its own processing.

Petri net is a tool for the study of systems [31].

And a definition for the Petri net is as follows:

Definition 2 *A marked Petri Net Structure, C, is a five-tuple, $C = (P, T, I, O, \mu)$; $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places; $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of transitions where $m, n > 0$ such that $P \cap T = \phi$; $I \subseteq \{P \times T\}$ is a input function of $t_i, i = 0, 1, \ldots, n$; $O \subseteq \{T \times P\}$ is a output function of $p_j, j = 0, 1, \ldots, m$; and the vector $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ is the marking where $\mu_i \in N$ is the number of tokens in place p_i.*

The state of a Petri net is represented by marking μ, an assignment of some number of tokens, represented by large black dots, to places.

A transition is able to fire in marking μ if each input place to that transition has at least one token in it. A transition is fired by removing one token from each input place and adding one token to each output places, resulting in a new marking.

The Petri net can be extended by transition labeling with the addition of top-down function or bottom-up function as in Equation 1-8 to each transition depending on the direction of traversing.

By the way of constructing an MEB graph, every place of the equivalent Petri net is reachable either in top-down or bottom-up approach.

Although the exact execution order of Petri net is not predetermined, due to its concurrency and asynchronous nature, some sequences by which nodes are invoked depend on the goal setup and environmental cost setting. The desired goal of the final product flow rate initiates backward propagation to figure out how much material is needed or how much by-product is produced.

The simulation by forward propagation computes every unit cost in a system with the unit costs of materials which are provided from outside of a system.

Having defined both the process network and the PN, it is helpful to define the process network in terms of PN for analyzing characteristics of the process network. Another definition of process network graph can be achieved by translation of process network graph into two kinds of extended Petri nets, one for the top-down simulation PN, P_t and the other for the bottom-up simulation PN, P_b, as follows:

For each instance of building block process, a pair of place and transition is created as a way to deal with the process synchronization problem. For each independent variable of a building block determined by Tables 4 and 5, Petri edge E is created so that each independent wire variable becomes

incoming Petri edges of $P \times T$, incident upon a transition node, and each dependent wire variable becomes outgoing Petri edges from a transition $t \in T$ to a place $p \in P$, incident upon a place. Again whether a wire variable is dependent or not is determined by Tables 4 and 5. The initial marking of P_t and P_b is determined by series of goals to achieve and those unit costs of materials provided from outside of a modeled system respectively.

Given a process network $G = (V, T, W, \Delta, \delta, \tau, \beta)$, the algorithm to translate the process network into marked top-down PN, P_t, using the Table 8 is as follows:

$P, T, I, O \leftarrow \phi$;
$\mu_0 \leftarrow \tau$;
for each $n_i \in V, T$
    $P \leftarrow P \cup \{n_i\}$
    unless the class of n_i.type is SIGNAL type
        $T \leftarrow T \cup \{n_i\}$;
for each $\omega = \{(n_s, n_d)\} \in W$
    if $(n_s = n_i)$ then
        $inbound = 0$;
    else
        $inbound = 1$;
    if $(g(ns.type, nd.type, inbound) == 1)$
        $I \leftarrow I \cup \{n_s\}$
        $O \leftarrow O \cup \{n_d\}$

Similar algorithm can be constructed to translate the process network into marked bottom-up PN, P_b, using the Table 9.

As an example, two translations from the process network graph of Figure 8.1 consisting of *Junction*, *Production*, and *Signal* class instances into a

pair of extended Petri net are shown in Figures 18 and 19. Note that two ordering of nodes in the resulting two PNs are opposite against each other.

In either simulation, whether it is top-down or bottom-up, the solvability of a process creates a token necessary to fire a transition for each input Petri edge in $P \times T$. By the proposed way of constructing an MEB graph, every place is guaranteed to be a reachable place, in other words, every process is guaranteed to have its own solution.

9.1 Execution of a Process

The execution environment is responsible for execution of a process and determines the Petri edges coming into a transition. Depending on the solvability of a place which is the source of incoming Petri edges, a token may or may not be assigned for each incoming Petri edge to the transition.

If every incoming edge to a transition has a token, then the top-down or bottom-up associated with the transition is ready to be fired; otherwise the current process will wait in the queue until all the required tokens are available.

If the *library* class process is encountered, after creation of the related process context and allocation of the required resources, process is switched to the new libray class process. The number of process contexts in a system will be the same as the number of *library* class instances.

The execution environment converts the MEB graph into the MEB Petri net and then, if a transition is ready to be fired, the associated function will

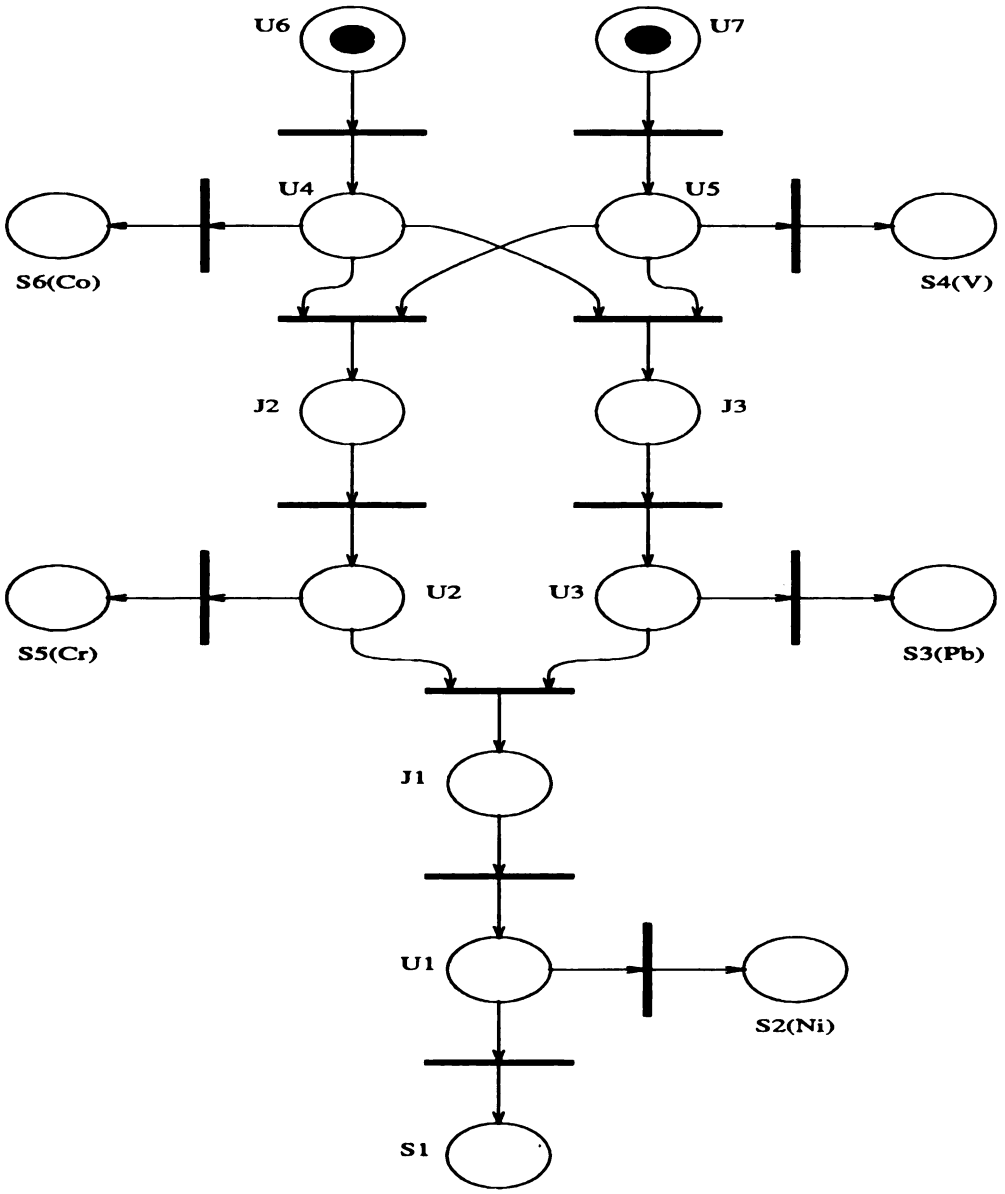


Figure 18: Translation of the MEB graph figure 8.1 into top-down Petri net

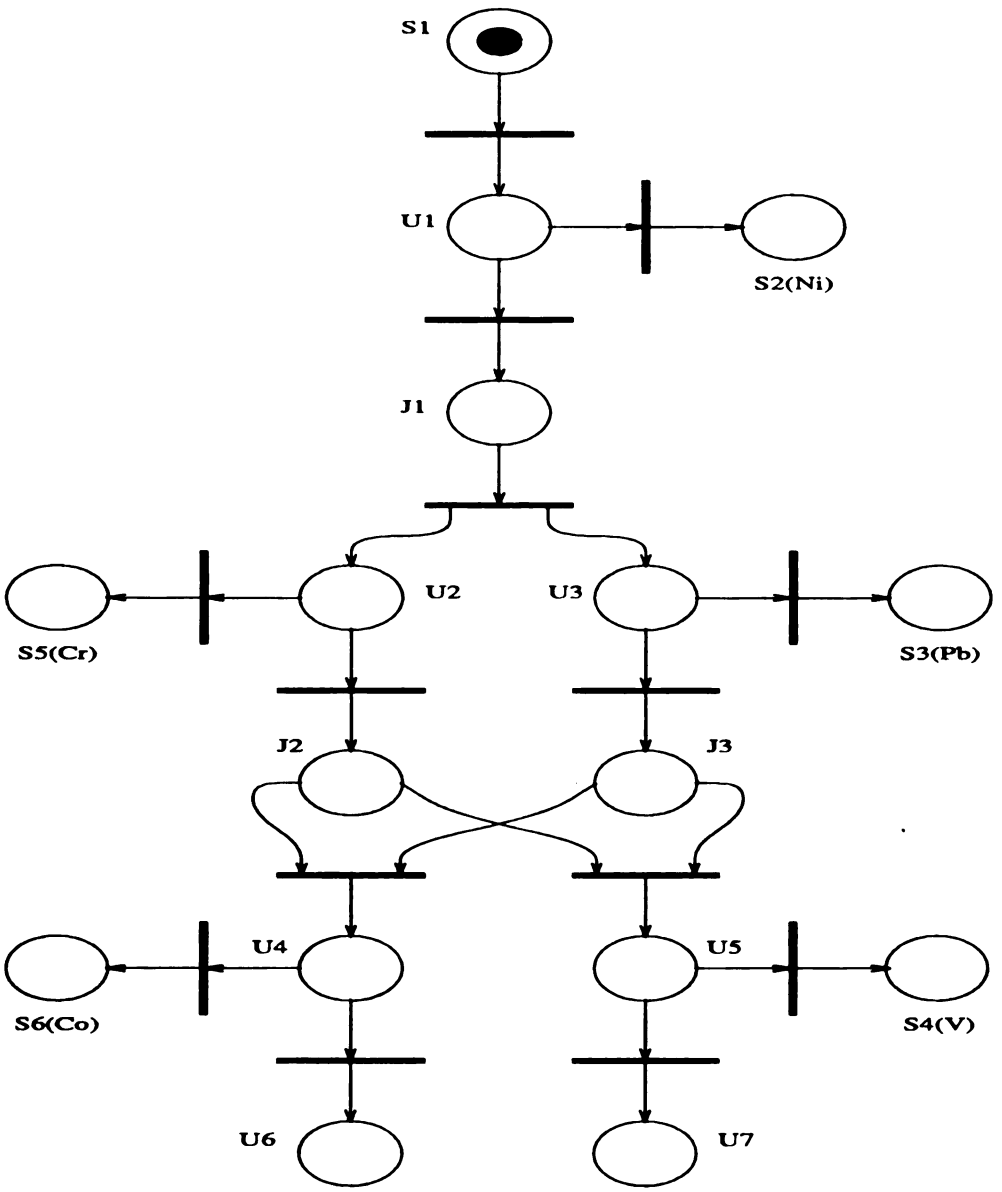


Figure 19: Translation of the MEB graph figure 8.1 into bottom-up Petri net

be executed.

The conversion of the MEB graph into the MEB Petri net is made possible by Table 8 which is used in top-down approach and Table 9 which is used in bottom-up approach.

And if a transition corresponding to a MEB building block is ready to be fired, meaning all the required tokens are available, then the associated top-down or bottom-up function will be executed. Depending on the solvability of the associated function, a new token may or may not be placed on the next place.

The corresponding algorithm is shown in Figure 20.

Theorem 1 *The the process network PN is safe and bounded.*

Proof: An enabled transition leads to execution of either top-down and bottom-up function associated with the place. Since the number of output function $|O| = 1$, there is only one place from an enabled transition. As a result only one token is deposited into the place at most. Also there is no loop in the process network PN. Therefore, The PN translated from a process network is always safe and bounded. □

Theorem 2 *There exists a firing sequence to reach every place of the process network PN.*

Proof: The process network PN is acyclic directed graph with root places given by τ or β. The root places have level of 0 and its children have level of 1.

Base step: The tokens in the root places are initialized by τ or β. For the first level transition, $|I| = 1$ and $|O| = 1$. These conditions satisify the firing condition of the first level transition leading to emptying tokens from root places and passing those tokens to its direct descendant places.

Hypothesis step: Assume that for all n-th level of places, there exists a firing sequence to have tokens in n-th level places.

Induction step: Since the process network PN is an acyclic graph, the level of places needed by $(n+1)$-th places are less than $(n+1)$ which were already deposited from previous steps. Having all the required tokens needed to fire $(n+1)$-th transitions are available, the $(n+1)$-th transitions are fired transfering tokens to $(n+1)$-th places or descendant transitions.

Thus there exists a firing sequence to reach every place in the process network PN. □.

Noting that tokens represent solvability of processes, the above theorem guarantees every process in the process network have a solution, if each subprocess can be solved.

The translation of the two-tuple elements in Tables 8 and 9 is as follows:

-1	Connection itself is not allowed
0	Connection is allowed but excluded from being the next place
1	Connection is allowed and should be the next place to be exeucted

Consider Figure 9.1 to show how to model a system having comprehensive use of all kinds of classes. The *library* class used in this example encapsulates the system in Figure 8.1.

```
Execute()
{
        IF the next process is a library class, THEN {
                IF the current process context is not the same as
                   the next one, THEN {
                        Set up enviroments for parameter passing;
                        Switch to a new process context;
                        Execute();
                        Switch back to the parent process;
                        Post-evaluate the wire variables of a child;
                        process to synchronize with those of a
                                    parent process;
        }

        IF the relevant top-down or bottom-up function associated
           with this transition is not solvable, THEN {
                Do not generate a token for this Petri edge going
                out of this transition;
        } ELSE {
                Generate a token for this Petri edge;
        }

        FOR each outgoing Petri edge of the current process {
                Find the next place to execute;
                If the transition is not ready to be fired, then
                        wait until all the tokens are ready;

                Execute();
        }
}
```

Figure 20: Algorithm for execution of a Petri net

	PROD.	RECY.	GOODS	STOR.	LIB.	JUNC.	SIGNAL
PRODUCTION	(1, 0)	(-1, 1)	(-1, 0)	(1, 0)	(1, 0)	(1, 0)	(1, 0)
RECYCLE	(0,-1)	(0, 1)	(-1,-1)	(-1,1)	(0,-1)	(-1,-1)	(-1,0))
GOODS	(1,-1)	(-1,-1)	(-1,-1)	(1,-1)	(1,-1)	(1,-1)	(-1,0)
STORAGE	(1,0)	(0,-1)	(-1,0)	(1,0)	(1,0)	(1,0)	(1,-1)
LIBRARY	(1,0)	(-1,0)	(-1,-1)	(-1,-1)	(1,-1)	(1,-1)	(1,-1)
JUNCTION	(1,0)	(-1,-1)	(-1,0)	(-1,0)	(1,0)	(1,0)	(1,-1)
SIGNAL	(0,0)	(0,0)	(-1,0)	(-1,0)	(-1,0)	(-1,0)	(-1,-1)

Table 8: Conversion table to transform MEB graph into MEB Petri net in top-down approach; row and column are current or next MEB block class respectively and the first and second element of a two-tuple is for inbound and outbound wire respectively

	PROD.	RECY.	GOODS	STOR.	LIB.	JUNC.	SIGNAL
PRODUCTION	(0, 1)	(-1, 0)	(-1, 0)	(0, 1)	(0, 1)	(0, 1)	(0, 0)
RECYCLE	(0,-1)	(0, 0)	(-1,-1)	(-1,0)	(0,-1)	(-1,-1)	(0,0))
GOODS	(0,-1)	(-1,-1)	(-1,-1)	(0,-1)	(0,-1)	(0,-1)	(0,-1)
STORAGE	(1,1)	(0,-1)	(-1,0)	(0,1)	(0,1)	(0,1)	(0,-1)
LIBRARY	(1,0)	(-1,0)	(-1,-1)	(-1,-1)	(1,-1)	(1,-1)	(1,-1)
JUNCTION	(0,1)	(-1,-1)	(-1,0)	(0, 1)	(0, 1)	(0, 1)	(0,-1)
SIGNAL	(0,1)	(0,0)	(-1,0)	(-1,1)	(0, 1)	(-1,1)	(-1,-1)

Table 9: Conversion table to transform MEB graph into MEB Petri net in bottom-up approach; row and column are current or next MEB block class respectively and the first and second element of a two-tuple is for inbound and outbound wire respectively

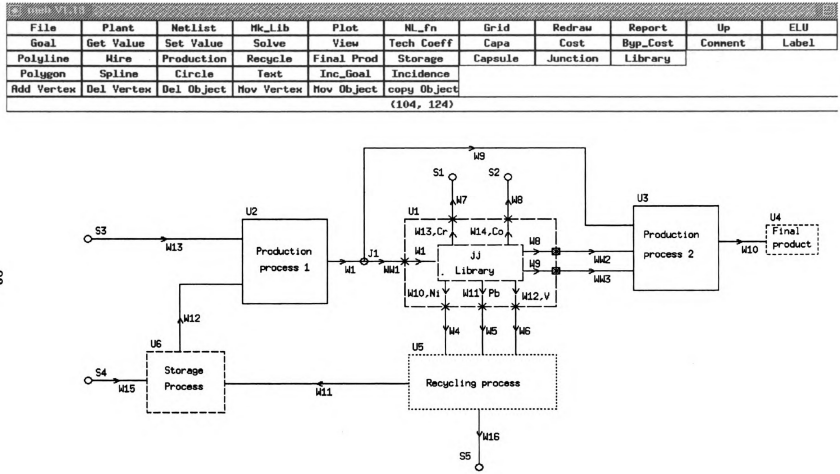


Figure 21: A system model having all classes

The translation of the MEB graph Figure 9.1 into MEBL is shown in Appendix section IV. And the correspnding translations of the MEB graph into MEB Petri net are shown in Figures 22 and 23.

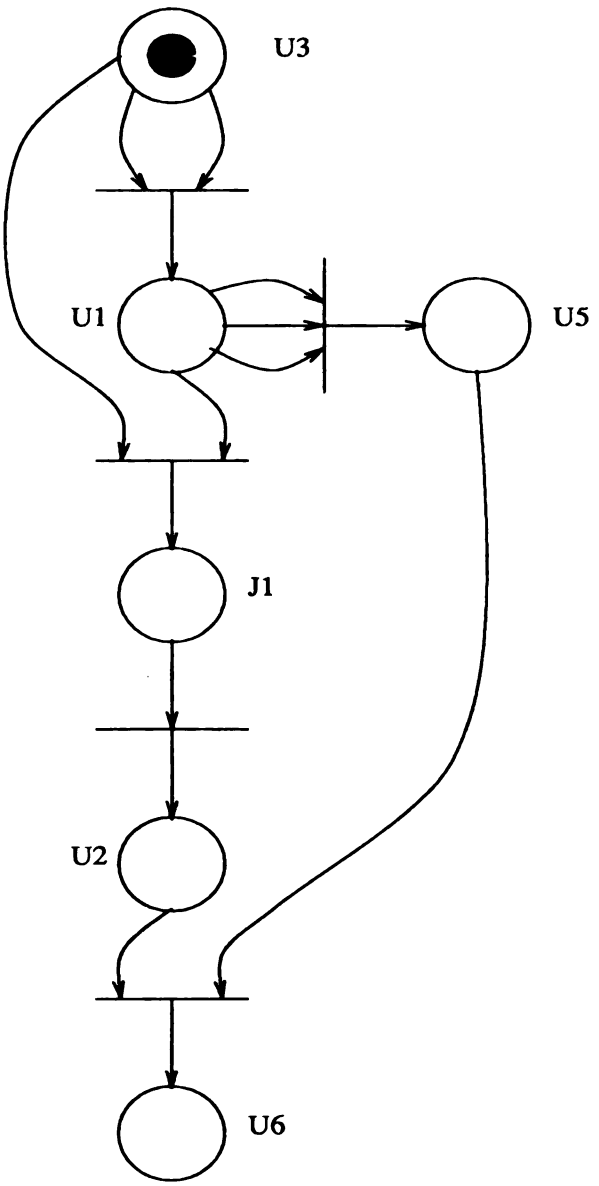


Figure 22: Translation of the MEB graph figure 9.1 into top-down Petri net

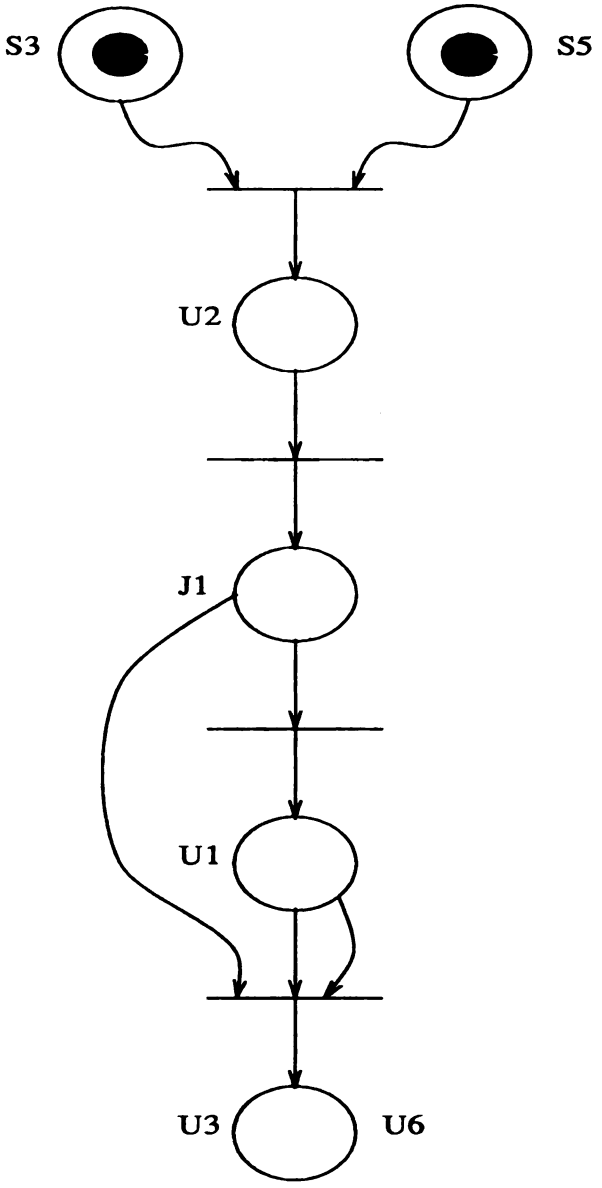


Figure 23: Translation of the MEB graph figure 9.1 into bottom-up Petri net

10 Graphic User Interface

Because MEB modeling partitions an overall system into a tractable amount of processes and MEB standardized modules, MEB theory is capable of succinct and crisp modeling of a complicated, very large system. As a system becomes larger and complex, the more importance of designing user interface should be emphasized for handling large system comfortably, safely, and efficiently[27].

In the drawing course to partition a system, graphical interactions between the user and the MEB simulation system play an important role more crucial than any other phase of simulation.

With the goal and scope definition of simulation in mind, a satisfactory modeling comes out of numerous repetitive corrections of models based on its resulting interpretation and its validity check.

As is one of MEB simulation characteristics, the mix of top-down and bottom-up approaches makes modeling look a lot more like a real world system because that is the basic nature involved in many design, analysis, and synthesis processes, though such characteristics might add one more complexity to a system.

The ease of drawing graphical objects embedding MEB theory determines smooth riding over the important phase of modeling with less pain. With the MEB GUI, from partitioning a model to seeing the results are but a few clicks of a mouse button away. The easiness and the simplicity of interactions make

it possible for a user to focus only on defining a system or process boundaries in this phase of partitioning a large system.

And this easiness of MEB GUI simulation helps to make a system more understandable. The GUI buttons are used to accomplish and control every aspect of system simlation activities.

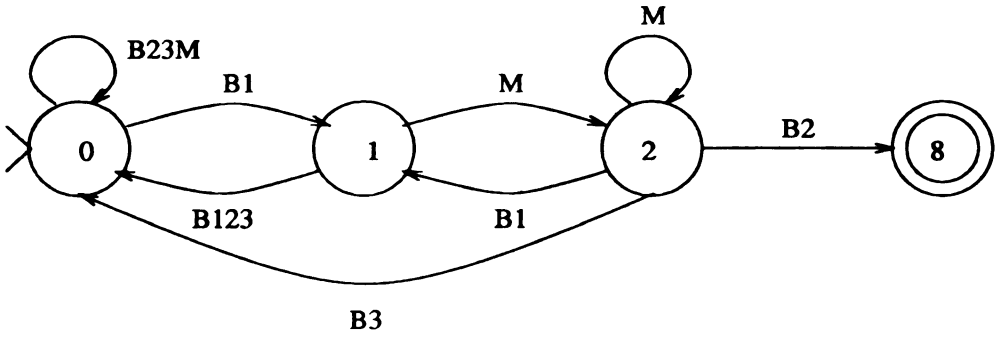
The interactions to accomplish most of the GDB related activities are designed with the Moore machine[23]. The literal `B` followed by numerals are the set of mouse buttons and `M` is the event of mouse movement.

Some Moore machines are shown in Figure 24 and Most drawing command buttons have their own automaton. Creation of rectangle shape by **<Production, Recycle, Final Prod, Storage, Capsule>** command buttons have Moore machines as in Figure 24 (a). The automata (b) and (c) in Figure 24 are for the **<Add Vertex>** and **<Del Vertex>** command buttons respectively.

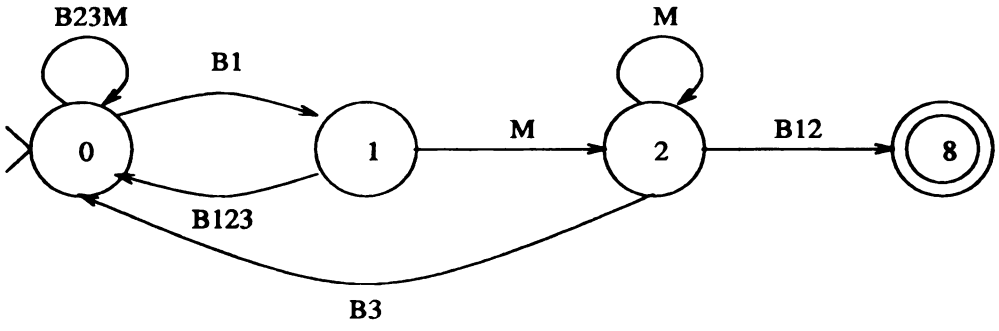
As an example, in the GUI of the button **< Wire >**, activities associated with its state diagram (a) in Figure 24 are shown as follows:

0	Clear GDB temporary buffer.
1	Store a vertex which forms a corner of rectangle.
2	Display rubber band rectangle.
8	Add a new instance to GDB.

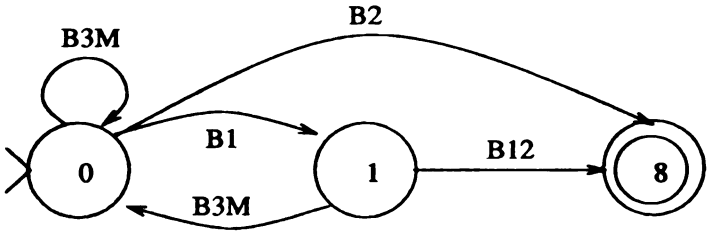
In the next section, MEB GUI will be introduced in terms of screen property composition, and interactions to accomplish simulation activities will be described.



(a)



(b)



(c)

Figure 24: Moore machines for GUI interactions

File	Save/load GDB to/from file, Save/load simulation envrionment to/from file
Plant	Save/load system description in MEBL to/from file
Netlist	Translate MEB graph into MEBL
Mk_lib	Use current drawing objects for *Library* shape definition
Plot	Plot 2-D graph for a visualization of simulation result
Nl_fn	Define different non–linear function in MEB theory
Grid	Define grid size for easy selection of GDB
Redraw	Redraw GDB on canvas
Report	Dump all the simulation results to a file
Up	Process context switch to parent
ELU	Evaulate total Environmental Load Unit(ELU)
Goal	Simulate after Setting up simulation environment
Get Value	Measure system performance
Solve	Simulate for an individual process
View	Observe resources for a process
Tech Coeff	Modify Technical Coefficient in a process
Capa	Modify top–down approach function
Cost	Modify bottom–up approach function
Byp_Cost	Change a byproduct cost
Comment	Assign a comment to a wire variable
Label	Change label of building blocks or wire

Table 10: Inbound wire;$N_{row} \times N_{column}$ constraints in tabular form

10.1 Screen Property Composition

The screen property is divided horizontally into three regions as in Figure 11.

The first region contains command buttons specifying one of the simulation activities. The functionality of the activities is summarized in Tables 10 and 11. The second region is used as the window to show the simulation status Lastly, the third one is the canvans window having the coordinate system with the origin in the upper left-hand corner.

By simulation environment I mean the list of goals to be achieved in

Polyline	Place polyline
Wire	Place staircase polyline
Production	Place *Production* class object
Recycle	Place *polyline* class object
Final Prod	Place *Goods* class object
Storage	Place *Storage* class object
Capsule	Define *Library* class object boundary
Junction	Place *Junction* class object
Library	Place *Library* class object
Polygon	Place polygons
Spline	Place splines
Circle	Place circle
Text	Place Text
Inc_Goal	Define incident points having independent wire variables in creating a new *Library* class
Incidence	Define incident points having dependent wire variables in creating a new *Library* class
Add Vertex	Add vertex
Del Vertex	Delete vertex
Del Object	Delete object
Mov Vertex	Move vertex
Mov Object	Move object
copy Object	Copy object

Table 11: Inbound wire;$N_{row} \times N_{column}$ constraints in tabular form

```
goal U6 W8.capa = [1:10:1];
goal U7 W9.capa = [1:10:1];
cost S1 W1.cost = 1*ones(size(W1.capa)) + exp(-0.5*abs(W1.capa));
```

Figure 25: Simulation environment set-up for the system in Table 8.1.

terms of flow rate of final products and the unit cost of materials supplied from outside of a system. Sometimes if a sytem gets larger, setting up those simulation environments by hand becomes very cumbersome. If those simulation environmental setups just were saved for later use or for changing part of them, it would save a lot of time or effort doing tedious interactions in setting up the environment every time a simulation is about to be performed. An example of such a simulation environment setup of the system in Table 8.1 is shown in Figure 25.

The expression of Figure 25 is automatically generated as default template statements. By changing part of the templates, it is not necessary to remember the whole exact syntax or to enter all of the expressions.

10.2 Graphic Database(GDB)

As noted in Figure 14, MEB simulation begins with drawing a system - MEB graph - using MEB building blocks to make a graphic database. MEB GDB is the resultant collection of such graphical objects.

An example of a GDB table of the system Figure 8.1 will look as follows:

```
 U1  rect 2 0     164 168     236 208
 U2  rect 2 0     304 88     376 128
 U3  rect 2 0     304 248     376 288
 U4  rect 2 0     468 88     540 128
 U5  rect 2 0     472 244     544 284
 U6  rect 2 2     612 100     672 112
 U7  rect 2 2     612 256     672 268
 W1  polyline 2     88 188     164 188
 W2  polyline 2     236 188     264 188
 W3  polyline 3     264 188     264 268     304 268
 W4  polyline 2     376 100     428 100
 W5  polyline 2     376 280     404 280
 W6  polyline 3     404 280     404 124     468 124
 W7  polyline 3     428 100     428 256     472 256
 W8  polyline 2     540 108     612 108
 W9  polyline 2     544 264     612 264
W10  polyline 2     196 208     196 292
W11  polyline 2     340 288     340 336
W12  polyline 2     508 284     508 336
W13  polyline 2     336 88     336 44
W14  polyline 2     504 88     504 44
 J1  junction 1     264 188
 J2  junction 1     428 100
 J3  junction 1     404 280
W15  polyline 3     264 188     264 108     304 108
W16  polyline 2     428 100     468 100
W17  polyline 2     404 280     472 280
TE1  text 1     348 60     Cr
TE2  text 1     516 64     Co
TE3  text 1     172 248     Ni
TE4  text 1     316 312     Pb
TE5  text 1     488 312     V
```

Figure 26: MEB GDB table of the system in Figure 8.1

11 Case Studies

11.1 Swine/Crop System

MEB theory is applied to a pasture-based farrow-to-finish swine production system. Three crops(wheat, soybeans, and corn) are grown on the farm to provide the bulk of the feed ration for the swine. The swine/crop agroecosystem is partitioned into physical and biological production processes[2].

The process flow diagram of a agroecosystem are depicted in Figure 11.1. Based on the data used in [2], the technical coefficients of the complex network of paper manufacturing are described in Tables 13,14, 14, and 15.

Given the unit costs of input material flows in 12, the unit costs of intermediate product and the final product are shown in Tables 18 and 19 along with the amount of each material flow.

11.2 Paper Cup LCA

The process flow diagram of a paper manufacturing plant and that of paper use and disposal are depicted in Figures 28 and 29 respectively. To enhance the view of overall LCA of paper cup, the whole LCA is described in Figure 30. This hierarchical modeling not only gives a bird's eye view of LCA but also gives detail system description as the system is further explored.

Based on the data used in [26], the technical coefficients of the complex network of paper manufacturing are described in Table 21 where response variables on the left column are expressed in terms of stimulus variables. Also the technical coefficients of the use and disposal phase of LCA is described

Variable	Description	Unit cost
W1	wheat seed	0.12500
W2	soybean seed	0.18750
W3	corn seed	1.18750
W4	grass seed	1.57500
W30	NPK fertilizer	0.35000
W31	pesticides	2.27000
W28	NPK fertilizer	0.35000
W29	pesticides	4.76000
W44	NPK fertilizer	0.35000
W45	pesticides	1.57500
W70	manure	0.00000
W20	water	0.00000
W21	medication	0.05000
W59	boars	233.33333
W60	sows	0.00000
W49	water	0.00000
W50	medication	0.05000
W67	gilts	155.000
W46	water	0.00000
W47	medication	0.05000
W26	purchased feed	0.04464
W27	supplement	0.15500

Table 12: Unit cost of material flux

Res. var.	Description	Tech. coeff.	Unit
U1(SOIL/PROD Wheat)			
W1	seed	2.5	lb/bu
W30	NPK	1.14286	lb/bu
W31	pesticides	0.12114	lb/bu
W32	leaching	0.00000	lb/bu
W5	biomass	stimulus	bu
U2(HARVESTING Wheat)			
W33	losses	0.05263	bu/bu
W5	biomass	1.05263	bu/bu
W6	grain	stimulus	bu
U3(STORAGE Wheat)			
W34	losses	0.01010	bu/bu
W6	biomass	1.01010	bu/bu
W7	grain	stimulus	bu
U4(Transport Wheat)			
W35	losses	0.00017	bu/lb
W7	grainss	0.01684	bu/lb
W14	wheat	stimulus	lb
U5(SOIL/PROD Soybeans)			
W2	seed	1.77778	lb/bu
W28	NPK	1.03333	lb/bu
W29	pesticides	0.07189	lb/bu
W39	leaching	0.00000	lb/bu
W8	biomass	stimulus	bu
U6(MARKETING Soybeans)			
W38	losses	0.05263	bu/bu
W8	biomass	1.05263	bu/bu
W10	soybeans	stimulus	bu

Table 13: The technical coefficients of swine/crop agroecosystem 1

Res. var.	Description	Tech. coeff.	Unit
U7(STORAGE Soybeans)			
W10	soybeans	1.01010	bu/bu
W37	lossess	0.01010	bu/bu
W12	soybeans	stimulus	bu
U8(TRANSPORT soybeans)			
W12	soybeans	0.01684	bu/lb
W36	losses	0.00017	bu/lb
W15	soybeans	stimulus	lb
U9(SOIL/PROD Corn)			
W3	seed	0.09836	lb/bu
W40	leaching	0.00000	lb/bu
W44	NPK	1.17612	lb/bu
W45	pesticides	0.07297	lb/bu
W10	biomass	stimulus	bu
U10(HARVESTING Corn)			
W41	losses	0.05263	bu/bu
W9	biomass	1.05263	bu/bu
W11	grain	stimulus	bu
U11(STORAGE Corn)			
W11	grain	1.01010	bu/bu
W42	losses	0.01010	bu/bu
W13	grain	stimulus	bu
U12(TRANSPORT Corn)			
W13	losses	0.01804	bu/lb
W43	biomass	0.00018	bu/lb
W16	soybeans	stimulus	lb
U13(SOIL/PROD Pasture)			
W100	leaching	0.00000	lb/lb
W4	seed	0.01400	lb/lb
W69	manure	61.11110	lb/lb
W17	biomass	stimulus	lb

Table 14: The technical coefficients of swine/crop agroecosystem 2

Res. var.	Description	Tech. coeff.	Unit
U14(GRAZING Pasture)			
W17	biomass	8.69565	lb/lb
W18	grass feed	stimulus	lb
U15(Manure Transport)			
W68	manure	1.00000	lb/lb
W51	manure	stimulus	lb
U16(Breeding and Farrowing)			
W19	grass feed	0.44190	lb/piglet
W20	water	147.780	gal/piglet
W21	medication	25.0000	mg/piglet
W24	feed	134.270	lb/piglet
W52	manure	311.116	lb/piglet
W54	gilts	0.03000	hd/piglet
W57	cull	0.03400	hd/piglet
W59	boars	0.00600	hd/piglet
W60	sows	0.04200	hd/piglet
W61	piglets	stimulus	piglet
U17(Nursery)			
W49	water	21.0000	gal/piglet
W50	medication	2.00000	mg/piglet
W53	manure	48.29974	lb/piglet
W56	dead animals	0.29070	hd/piglet
W61	piglets	1.29199	hd/piglet
W64	feed	15.0000	lb/piglet
W62	piglets	stimulus	piglet

Table 15: The technical coefficients of swine/crop agroecosystem 3

Res. var.	Description	Tech. coeff.	Unit
	U18(GROWING/FINISHING)		
W46	water	452.08333	gal/S.H.
W47	medication	1.00000	mg/S.H.
W48	gilts	0.04167	hd/S.H.
W58	dead animals	0.03225	hd/S.H.
W62	piglets	1.07500	hd/S.H.
W63	pasture	2.58056	lb/S.H.
W65	feed	784.19444	lb/S.H.
W66	manure	1043.75	lb/S.H.
W55	slaughter hogs	stimulus	S.H.(Slaughter Hog)
	U19(FEED/MILL)		
W14	wheat	0.36293	lb/lb
W15	soybeans	0.14282	lb/lb
W16	corn	0.47274	lb/lb
W25	losses	0.02041	lb/lb
W26	purchased feed	0.00000	lb/lb
W27	supplement	0.04192	lb/lb
W22	feed	stimulus	lb

Table 16: The technical coefficients of swine/crop agroecosystem 3

Process name	Description	Fixed cost	Variable cost	Unit cost
U1	SOIL/PROD Wheat	716.05	0.32143	$/bu
U2	HARVESTING Wheat	741.81	0.30827	$/bu
U3	STORAGE Wheat	356.59	0.08069	$/bu
U4	Transport Wheat	0.00	0.00008	$/lb
U5	SOIL/PROD Soybeans	453.33	0.97778	$/bu
U6	MARKETING Soybeans	306.51	0.48655	$/bu
U7	STORAGE Soybeans	192.26	0.08664	$/bu
U8	TRANSPORT soybeans	0.00	0.00008	$/lb
U9	SOIL/PROD Corn	503.87	0.32836	$/bu
U10	HARVESTING Corn	387.59	0.22938	$/bu
U11	STORAGE Corn	854.53	0.23011	$/bu
U12	TRANSPORT Corn	0.00	0.00008	$/lb
U13	SOIL/PROD Pasture	0.00	0.06000	$/lb
U14	GRAZING Pasture	0.00	0.02609	$/lb
U15	Manure Transport	183.75	0.00005	$/lb
U16	Breeding and Farrowing	1,001.84	3.54662	$/piglet
U17	Nursery	1,732.79	3.71150	$/piglet
U18	GROWING/FINISHING	3,170.35	11.08271	$/S.H.
U19	FEED/MILL	2,805.67	0.00200	$/lb

Table 17: Unit input costs

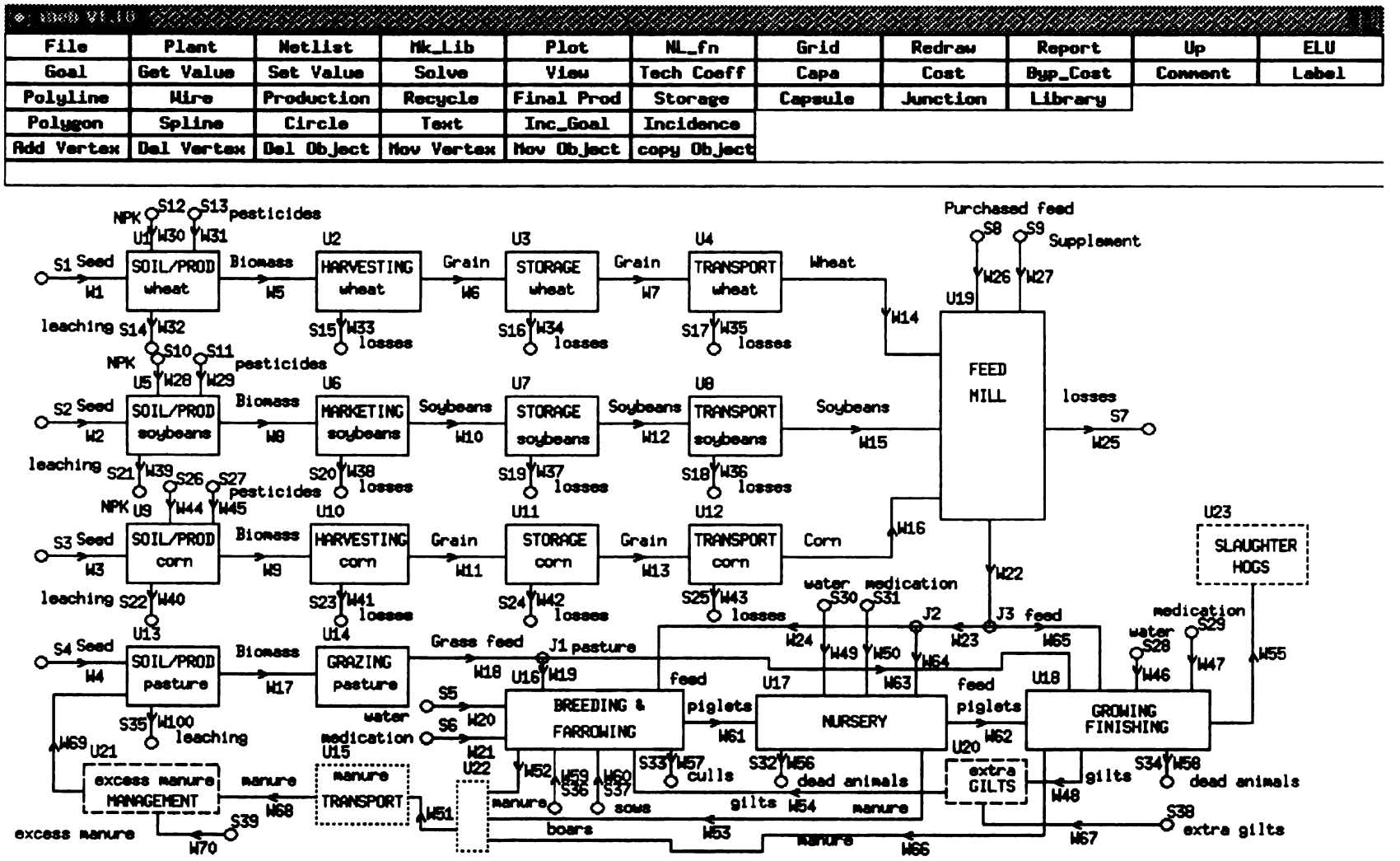


Figure 27: MEB network swine/crop agroecosystem

Variable	Description	Material outputs		Cost per unit of output	
W55	HOGS(goal)	360	370	101.74	100.7
W1	wheat seed	5771.4	5931.7	0.125	0.125
W2	soybean seed	1615	1659.9	0.1875	0.1875
W3	corn seed	316.85	325.65	1.1875	1.1875
W4	grass seed	139.99	143.88	1.575	1.575
W5	biomass	2308.5	2372.7	1.6191	1.6107
W6	wheat grain	2193.1	2254	2.3508	2.3329
W7	wheat grain	2171.2	2231.5	2.6195	2.5969
W8	biomass	908.46	933.69	2.514	2.5005
W9	biomass	3221.3	3310.8	1.1281	1.1239
W10	soybeans	863.04	887.01	3.488	3.4642
W11	grain	3060.3	3145.3	1.5436	1.5357
W12	soybeans	854.41	878.14	3.8349	3.8048
W13	grain	3029.7	3113.8	2.0548	2.0397
W14	wheat	1.2893e+05	1.3251e+05	0.044192	0.043812
W15	soybeans	50737	52146	0.06466	0.064152
W16	corn	1.6794e+05	1.7261e+05	0.037149	0.036876
W17	biomass	9999.6	10277	0.10318	0.10268
W18	pasture	1150	1181.9	0.92327	0.91895
W19	pasture	220.95	227.09	0.92327	0.91895
W20	water	73890	75943	0	0
W21	medication	12500	12847	0.05	0.05
W22	feed	3.5525e+05	3.6512e+05	0.05723	0.056677
W23	feed	72940	74966	0.05723	0.056677
W24	feed	67135	69000	0.05723	0.056677
W25	losses	7250.7	7452.1	0	0
W26	purchased feed	0	0	0.04464	0.04464
W27	supplement	14892	15306	0.155	0.155
W28	NPK	938.74	964.81	0.35	0.35
W29	pesticides	65.309	67.123	4.76	4.76
W30	NPK	2638.3	2711.6	0.35	0.35

Table 18: Swine/Crop agroecosystem 1

Variable	Description	Material outputs		Cost per unit of output	
W55	HOGS(goal)	360	370	101.74	100.7
W31	pesticides	279.66	287.43	2.27	2.27
W32	leaching	0	0	0	0
W33	losses	115.42	118.63	0	0
W34	losses	21.929	22.538	0	0
W35	losses	21.918	22.527	0	0
W36	losses	8.6253	8.8648	0	0
W37	losses	8.6295	8.8692	0	0
W38	losses	45.422	46.683	0	0
W39	leaching	0	0	0	0
W40	leaching	0	0	0	0
W41	losses	161.06	165.54	0	0
W42	losses	30.6	31.449	0	0
W43	losses	30.229	31.069	0	0
W44	NPK	3788.7	3893.9	0.35	0.35
W45	pesticides	235.06	241.59	1.575	1.575
W46	water	1.6275e+05	1.6727e+05	0	0
W47	medication	360	370	0.05	0.05
W48	gilts	15.001	15.418	0	0
W49	water	8127	8352.8	0	0
W50	medication	774	795.5	0.05	0.05
W51	manure	5.5e+05	5.6528e+05	0.00038	0.00038
W52	manure	1.5556e+05	1.5988e+05	0.00038	0.00038
W53	manure	18692	19211	0.00038	0.00038
W54	gilts	15	15.417	-0.01236	-0.01236
W56	dead animals	112.5	115.63	0	0
W57	culls	17	17.472	0	0
W58	dead animals	11.61	11.933	0	0
W59	boars	3	3.0833	233.33	233.33

Table 19: Swine/Crop agroecosystem 2

Variable	Description	Material outputs		Cost per unit of output	
W55	HOGS(goal)	360	370	101.74	100.7
W60	sows	21	21.583	0	0
W61	piglets	500	513.89	16.41	16.28
W62	piglets	387	397.75	30.368	30.07
W63	pasture	929	954.81	0.92327	0.91895
W64	feed	5805	5966.2	0.05723	0.056677
W65	feed	2.8231e+05	2.9015e+05	0.05723	0.056677
W66	manure	3.7575e+05	3.8619e+05	0.00038	0.00038
W67	extra gilts	-0.0011961	-0.0012293	155	155
W68	manure	5.5e+05	5.6528e+05	0.00038409	0.00037506
W69	manure	6.1109e+05	6.2806e+05	0.0003457	0.00033757
W70	excess manure	61085	62782	0	0

Table 20: Swine/Crop agroecosystem 3

in Table 22 and 23.

With the unit cost of system resourses in Table 24, some of the simulation results, for brievity, are shown as follows:

```
PROCESS(0): plca
==========================
W1.capa: [ 0.001 0.002 0.003 0.004 0.005 ; ]
W1.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

W2.capa: [ 0.06 0.12 0.18 0.24 0.3 ; ]
W2.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

W3.capa: [ 0.02 0.04 0.06 0.08 0.1 ; ]
W3.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

W4.capa: [ 0.03 0.06 0.09 0.12 0.15 ; ]
W4.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

W5.capa: [ 0.01 0.02 0.03 0.04 0.05 ; ]
W5.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

W6.capa: [ 0.01 0.02 0.03 0.04 0.05 ; ]
```

Variable	Description	Tech. Coeff.
	U3(Pulp Manufacturing)	
W12	r3(chlorine)	0.06
W49	r4(sodium hydroxide)	0.02
W54	r5(sodium chlorate)	0.03
W55	r6(sulfuric acid)	0.01
W56	r7(sulfur dioxide)	0.01
W57	r8(calcium oxide)	0.01
W58	r2(water)	0.10
W39	wood chip	2.2
W1	sodium sulfate(s.s.)	0.009
W11	recycled s.s.	0.01
W16	r22(H2O)	0.07
W17	r23(suspended solids)	0.01
W18	r24(BOD)	0.005
W19	r25(organochlorides)	0.003
W20	r26(cellulosic fiber)	0.001
W21	r27(inorganic salts)	0.06
W22	r19(chlorine)	0.0002
W23	r20(chlorine dioxide)	0.0002
W24	r21(reduced sulfides)	0.0015
W46	black liquor	1.2
W24	pulp	stimulus
	U6(Waste Wood & Black Liquor Combustion)	
W41	CO2	0.0
W42	CO	0.028
W43	NOx	0.046
W44	SO2	0.100
W45	particulates	0.015
W2	bark and waste	stimulus
W59	smelt	0.17
W46	black liquor	stimulus

Table 21: The technical coefficients of paper manufacturing

Variable	Description	Tech. Coeff.
	U4(Waste Paper Re-pulping)	
W29	r3(chlorine)	0.06
W28	r4(sodium hydroxide)	0.02
W27	r5(sodium chlorate)	0.03
W26	r6(sulfuric acid)	0.01
W25	r7(sulfur dioxide)	0.01
W13	r8(calcium oxide)	0.01
W14	r2(water)	0.10
W53	waste pulp	1.0
W30	r22(H2)	0.04
W31	r23(suspended solids)	0.005
W32	r24(BOD)	0.003
W33	r25(organochlorides)	0.002
W34	r26(cellulosic fiber)	0.001
W35	r27(inorganic salts)	0.03
W36	r19(chlorine)	0.001
W37	r20(chlorine dioxide)	0.001
W38	r21(reduced sulfides)	0.0008
W48	recycled pulp	stimulus
	U5(Wood Processing)	
W2	bark and waste	0.06
W40	wood logs	1.06
W39	wood chips	stimulus
	U7(Paper Manufacturing)	
W47	pulp	1.0
W48	recycled pulp	0.0
W51	H2O	0.01
W50	paper	stimulus
	U2(Cup Manufacturing)	
W2	paper	1.0
W6	adhesive	0.0
W8	waste paper	0.03
W3	cups	stimulus

Table 22: The technical coefficients of cup use and disposal

Variable	Description	Tech. coeff.
	U2(Cup Use)	
W4	cups	1.0
W7	beverages	1.0
W9	used cups	1.0
W5	beverages	stimulus
	U6(Waste Paper Transport)	
W10	incin. wasted paper	0.0
W12	landfill wasted paper	1.0
W8	waste paper	stimulus
	U7(Used Cup Transport)	
W11	landfill used cups	1.0
W13	incin. used cups	0.0
W9	used cups	stimulus
	U8(Used Cup Transport)	
W14	r14(CO2)	0.0
W15	r15(CO)	0.028
W16	r16(NOx)	0.046
W17	r17(SO2)	0.100
W18	r18(particulates)	0.015
W19	fuel needed	0.0
W20	ash	0.03
W13	used cups	stimulus
	U9(Landfill)	
W21	r14(CO2)	0.0
W22	r32(methane)	0.0
W23	r33(leachate)	0.0
W24	r34(cellulosic fiber)	1.0
W25	r24(BOD)	0.0
W11	used cups	stimulus
W12	wasted paper	stimulus

Table 23: The technical coefficients of cup use and disposal

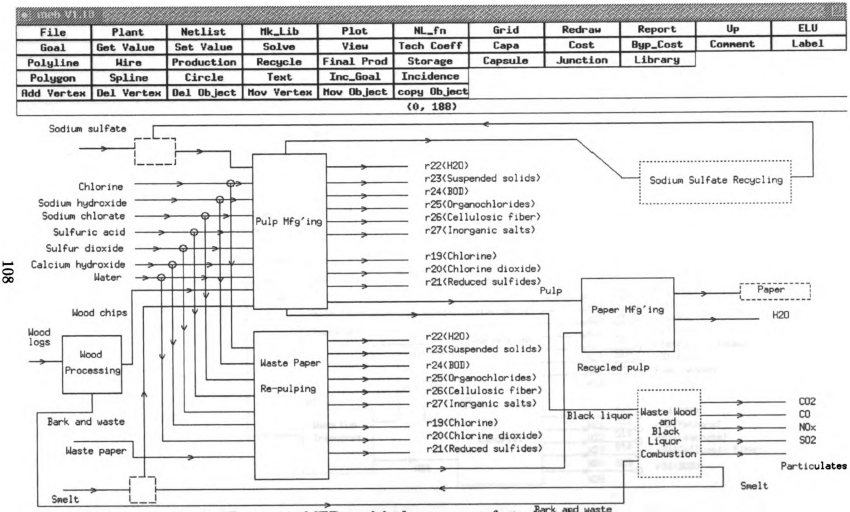


Figure 28: MEB model of paper manufacturing plant

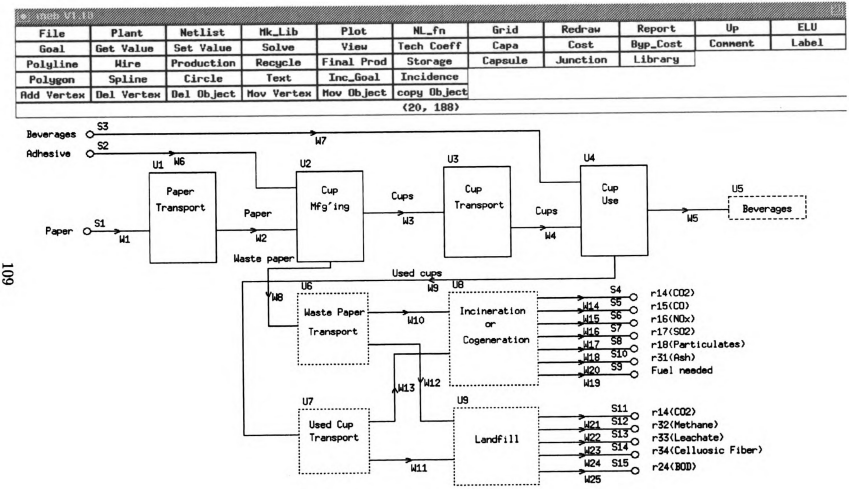


Figure 29: MEB model of paper use and disposal system

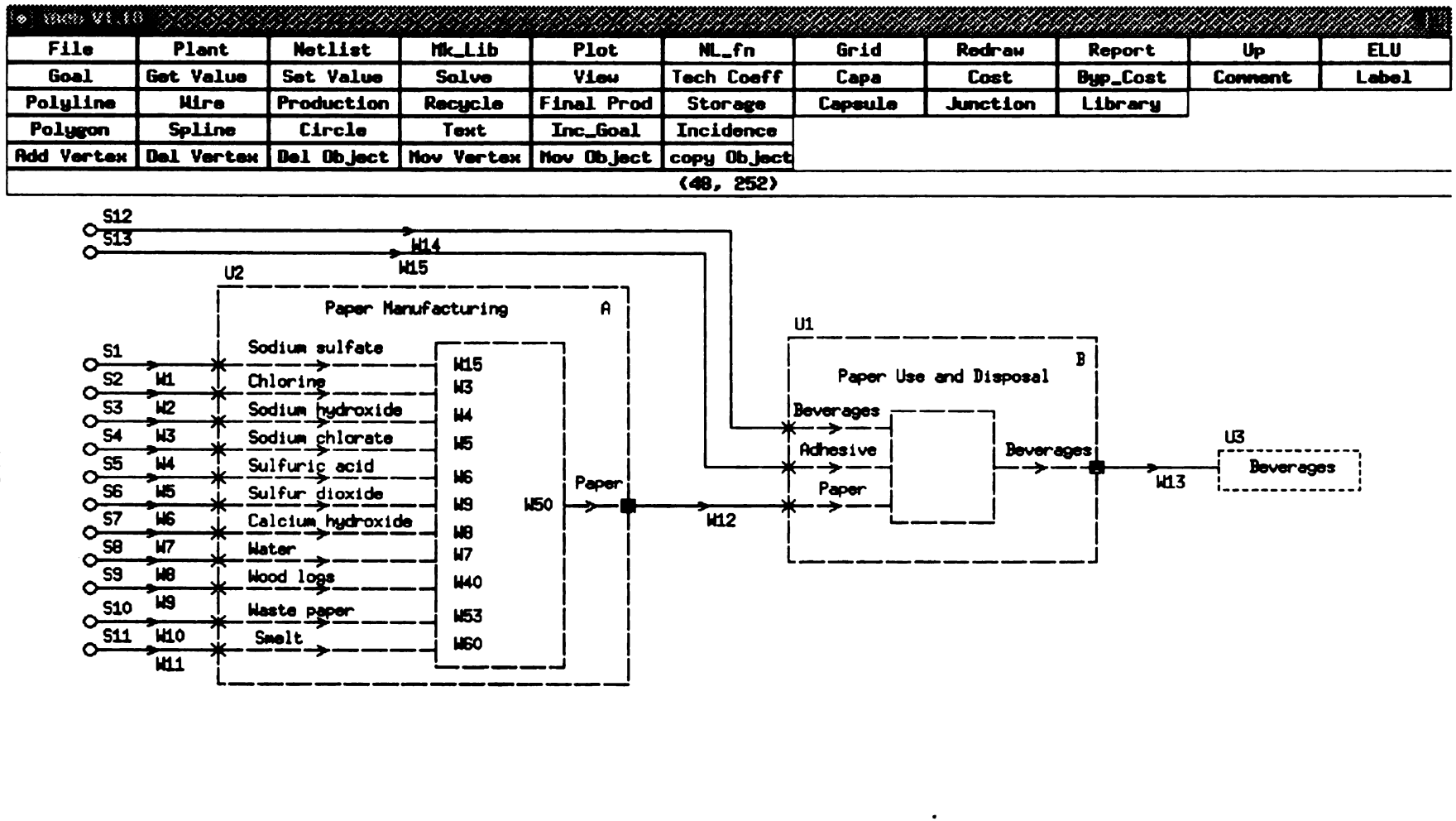


Figure 30: MEB model for paper cup life cycle analysis

Res. var.	Description	Energetic cost
W1	Sodium sulfate	$0.25
W2	Chlorine	$0.25
W3	Sodium hydroxide	$0.25
W4	Sodium chlorate	$0.25
W5	Sulfuric acid	$0.25
W6	Sulfur dioxide	$0.25
W7	Calcium hydroxide	$0.25
W8	Water	$0.00
W9	Wood logs	$0.10
W10	Waste paper	$0.00
W11	smelt	$0.00
W14	Beverage	$0.00
W15	Adhesive	$0.25

Table 24: The unit costs of system resources for paper cup use and disposal

```
W6.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

W7.capa: [ 0.01 0.02 0.03 0.04 0.05 ; ]
W7.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

W8.capa: [ 0.1 0.2 0.3 0.4 0.5 ; ]
W8.cost: [ 4.54e-05 2.0612e-09 9.3576e-14 4.2484e-18 1.9287e-22 ; ]

W9.capa: [ 2.332 4.664 6.996 9.328 11.66 ; ]
W9.cost: [ 0.08 0.08 0.08 0.08 0.08 ; ]

W10.capa: [ 0 0 0 0 0 ; ]
W10.cost: [ 1 1 1 1 1 ; ]

W11.capa: 0 ;
W11.cost: 1 ;

W12.capa: [ 1 2 3 4 5 ; ]
W12.cost: [ 0.22181 0.22181 0.22181 0.22181 0.22181 ; ]

W13.capa: [ 1 2 3 4 5 ; ]
W13.cost: [ 0.22181 0.22181 0.22181 0.22181 0.22181 ; ]
```

```
W14.capa: [ 1 2 3 4 5 ; ]
W14.cost: [ 3.7201e-44 1.3839e-87 5.1482e-131 1.9152e-174 7.1246e-218 ; ]

W15.capa: [ 0 0 0 0 0 ; ]
W15.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

PROCESS(1): puse
=========================
W1.capa: [ 1 2 3 4 5 ; ]
W1.cost: [ 0.22181 0.22181 0.22181 0.22181 0.22181 ; ]

W2.capa: [ 1 2 3 4 5 ; ]
W2.cost: [ 0.22181 0.22181 0.22181 0.22181 0.22181 ; ]

W3.capa: [ 1 2 3 4 5 ; ]
W3.cost: [ 0.22181 0.22181 0.22181 0.22181 0.22181 ; ]

W4.capa: [ 1 2 3 4 5 ; ]
W4.cost: [ 0.22181 0.22181 0.22181 0.22181 0.22181 ; ]

W5.capa: [ 1 2 3 4 5 ; ]
W5.cost: [ 0.22181 0.22181 0.22181 0.22181 0.22181 ; ]

W6.capa: [ 0 0 0 0 0 ; ]
W6.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

            :
            :
            :

PROCESS(2): pmfg
=========================
W1.capa: [ 0.009 0.018 0.027 0.036 0.045 ; ]
W1.cost: [ 0 0 0 0 0 ; ]

W2.capa: [ 0.132 0.264 0.396 0.528 0.66 ; ]
W2.cost: [ 0 0 0 0 0 ; ]

W3.capa: [ 0.06 0.12 0.18 0.24 0.3 ; ]
W3.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

W4.capa: [ 0.02 0.04 0.06 0.08 0.1 ; ]
W4.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]
```

```
W5.capa: [ 0.03 0.06 0.09 0.12 0.15 ; ]
W5.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

W6.capa: [ 0.01 0.02 0.03 0.04 0.05 ; ]
W6.cost: [ 0.25 0.25 0.25 0.25 0.25 ; ]

                :
                :
                :
```

11.3 Ford F150 Truck Tail Light Assembly

The tail light assembly structure plant is shown in Figure 31.

The technical coefficients of the Ford F150 truck tail light assembly plant are give in Table 25 and Table 26.

11.4 Water Plant Modelling

The water plant is modelled in Figure 32. And all the byproduct costs are assumed to be zero dollar. The electricity cost of pump with efficiency of 0.85 is computed as follows:

$$\begin{aligned} x_{elec} &= \$0.05 \cdot 0.73 kwh/1 KGD/100 PSI \cdot kgd \cdot psi/0.85 \\ &= 0.042941 \cdot kgd \cdot psi \end{aligned}$$

Given the plant water demand, the problem is to find all the costs of products in the water plant. The result is summarized in Section 11.4.

The plant water demands are as follows:

Res. var.	Description	Tech. coeff.	unit
	U1(Body molding)		
W1	plastic	0.551	oz.
W2	gates	0.05	unit
W3	scrap	0.05	unit
W4	body	stimulus	unit
	U2(Body metallized)		
W15	metal waste	0.0	mg
W19	metal	75	mg
W4	body	1	unit
W5	metalized body	stimulus	unit
	U3(Glued body)		
W16	purge	0.0088	oz.
W17	plastic	0.0088	oz.
W20	glue	0.013	oz.
W5	body	1	unit
W6	glued body	stimulus	unit
	U4(Lens mating)		
W30	lens	1	unit
W6	glued body	1	unit
W7	body	stimulus	unit
	U5(Dry on rack)		
W7	body	1	unit
W14	body	stimulus	unit

Table 25: F150 tail light assembly modeling 1

Res. var.	Description	Tech. coeff.	unit
U6(Drive studs)			
W14	dried body	1	unit
W31	studs	2	unit
W6	body	stimulus	
U7(Leak test)			
W18	scrap	0.0045	unit
W8	body	0.9955	unit
W9	body passed	stimulus	unit
U8(Put bulbs in)			
W21	body	1	unit
W9	socket	1	unit
W10	body	stimulus	unit
U9(Inspection)			
W10	body	0.95	unit
W12	scrap	0.05	unit
W11	body	stimulus	unit
U13(Socket assembly)			
W32	bulb	2	unit
W33	socket	2	unit
W21	socket	stimulus	unit

Table 26: F150 tail light assembly modeling 2

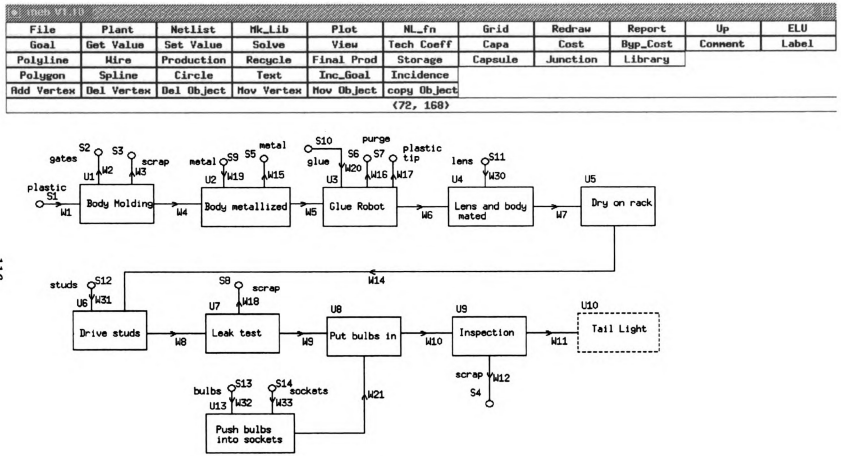


Figure 31: Ford F150 tail light assembly plant

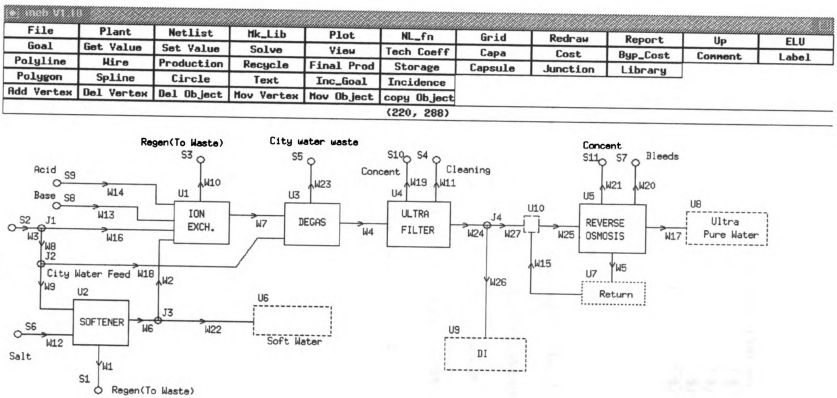


Figure 32: Recycle option I

Product	Amount of Flow	Cost per Unit($)	Unit
Soft Water(W26)	500	? $/yr	KGD
DI(W26)	300	? $/yr	KGD
UPW(W17)	800	? $/yr	KGD
	Total Cost	? $/yr	

- REVERSE OSMOSIS(U5)

  1. Pump dP = 300 psi, Eff. = 85%
  2. Rejection 95% of TDS ppm, 100% of RO Feed TSS
  3. Concentrate Flow(W21) = 5% of RO Feed(W25)
  4. Bleed Flow(W20) = 5% of RO Product(W_f)
  5. Return Flow(W5) = 5% of RO Product(W_f)

From (3), (4), (5), and by the conservation of mass respectively,

$$y_{21} = 0.05 y_{25}$$

$$y_{20} = 0.05 y_f$$

$$y_5 = 0.05 y_f$$

$$y_{25} = y_5 + y_{20} + y_{21} + y_f$$

$$y_f = y_{17} + y_{20} + y_5$$

$$\Rightarrow \quad y_5 = y_{20} = 0.05 y_f \stackrel{def}{=} a$$

$$\Rightarrow \quad y_{25} = (a) + (a) + y_{21} + (20a)$$

$$\Rightarrow \quad 20 y_{21} = 22a + y_{21}$$

$$\Rightarrow \quad y_{21} = \frac{22a}{19} = 1.1579a$$

$$\Rightarrow \quad y_{25} = (a) + (a) + (1.1579a) + (20a) = 23.1579a$$

$$\Rightarrow \quad y_f = y_{17} + (a) + (a)$$

$$\Rightarrow \quad 20a = y_{17} + 2a$$

$$\Rightarrow \quad a = \frac{y_{17}}{18}$$

$$\Rightarrow \quad \begin{bmatrix} y_{20} \\ y_{21} \\ y_{25} \\ y_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1.1579 \\ 23.1579 \\ 1 \end{bmatrix} a = \begin{bmatrix} 0.05555 \\ 0.06433 \\ 1.28655 \\ 0.05555 \end{bmatrix} y_{17} \stackrel{def}{=} k y_{17}$$

$$x_{17} = k' \begin{bmatrix} x_{20} \\ x_{21} \\ x_{25} + 0.042941 * 3 \\ x_5 \end{bmatrix}$$

- ULTRA FILTER(U4)

  1. Pump dP = 100 psi, Eff. = 85%
  2. Concentrate Flow(W19) = 10% of UF Feed(W4).
  3. Cleaning Flow(W11) = 1% of UF Feed(W4).
  4. Rejection = 0% of TDS ppm, 95% of TSS ppm

From (2), (3), and by the conservation of mass respectively,

$$y_{19} = 0.1 y_4$$

$$y_{11} = 0.01 y_4$$

$$y_4 = y_5 + y_{11} + y_{19}$$

$$\Rightarrow \quad y_4 = y_5 + (0.01y_4) + (0.1y_4)$$

$$\Rightarrow \quad (1 - 0.11)y_4 = y_5$$

$$\Rightarrow \quad y_4 = 1.1236y_5$$

$$\Rightarrow \quad y_{19} = 0.1(1.1236y_5) = 0.11236y_5$$

$$\Rightarrow \quad y_{11} = 0.01(1.1236y_5) = 0.011236y_5$$

$$\Rightarrow \quad \begin{bmatrix} y_{11} \\ y_{19} \\ y_4 \end{bmatrix} = \begin{bmatrix} 0.011236 \\ 0.11236 \\ 1.1236 \end{bmatrix} y_5 \stackrel{def}{=} ky_5$$

$$x_24 = k' \begin{bmatrix} x_{11} \\ x_{19} \\ x_4 + 0.042941 \end{bmatrix}$$

- DEGAS(U3)

1. Cooling Water (City) Usage(W18,W23) = 0.5 gpm/KGD = 0.0005 kgpm/KGD
2. Degas Water Feed(W7) = Degas Water Product(W4)

From (1), (2), and by the conservation of mass respectively,

$$y_{18} = y_{23} = 0.5y_4$$

$$y_7 = y_4$$

$$\Rightarrow \quad y_5 = 0.05y_{17} + 0.05y_{17} + y_{17}$$

$$\Rightarrow \qquad y_5 = 1.1 y_{17}$$

$$\Rightarrow \quad \begin{bmatrix} y_{18} \\ y_{23} \\ y_7 \end{bmatrix} = \begin{bmatrix} 0.0005 \\ 0.0005 \\ 1 \end{bmatrix} y_4 \stackrel{def}{=} k y_4$$

- ION EXCH.(U1)

  All the by-product costs are assumed to be zero dollars.

  1. Pump dP = 100 psi, Eff. = 85%
  2. Regen. Flow(W10) = 35% of IX Product(W7) (50% Feed(W16), 50% Soft(W2), + Acid(W14) & Base(W13))
  3. Effectiveness = 100% removal of IX Feed TDS Ion
  4. Acid Usage = 1 gal/kgal/100ppm ion Acid Cost = \$0.30/gal = \$300/kgal
  5. Base Usage = 0.5 gal/kgal/100ppm ion Base Cost = \$1.50/gal = \$1500/kgal
  6. TDS in City Feed 300 ppm

  From (2), (4), (5), (6), and by the conservation of mass respectively,

$$y_{10} = 0.35 y_7$$

$$y_{16} = y_2 \stackrel{def}{=} a$$

$$y_{14} = (0.001)(300 ppm/100 ppm)(y_{16} + y_2) = 0.006a$$

$$y_{13} = (0.0005)(300 ppm/100 ppm)(y_{16} + y_2) = 0.003a$$

$$y_7 + y_{10} = y_{16} + y_2 + y_{13} + y_{14}$$

$$\Rightarrow \quad y_7 + y_{10} = (a) + (a) + (0.006a) + (0.003a) = 2.009a$$

$$\Rightarrow \quad (\tfrac{1}{0.35} + 1)y_{10} = 2.009a$$

$$\Rightarrow \quad y_{10} = 0.520852a$$

$$\Rightarrow \quad (0.520852a) = 0.35y_7$$

$$\Rightarrow \quad a = 0.671976y_7$$

$$\Rightarrow \quad \begin{bmatrix} y_{10} \\ y_{13} \\ y_{14} \\ y_{16} \\ y_2 \end{bmatrix} = \begin{bmatrix} 0.35y_7 \\ 0.003a \\ 0.006a \\ a \\ a \end{bmatrix} = \begin{bmatrix} 0.35 \\ 0.002016 \\ 0.004032 \\ 0.671976 \\ 0.671976 \end{bmatrix} y_7 \stackrel{def}{=} ky_7$$

$$x_7 = k' \begin{bmatrix} x_{10} \\ x_{13} + 0.042941 \\ x_{14} + 0.042941 \\ x_{16} + 0.042941 \\ x_2 + 0.042941 \end{bmatrix} = k' \begin{bmatrix} 0 \\ 1500 + 0.042941 \\ 300 + 0.042941 \\ 2 + 0.042941 \\ x_2 + 0.042941 \end{bmatrix}$$

- SOFTENER(U2)

  All the by-product costs are assumed to be zero dollars.

  1. Pump dP = 100 psi, Eff. = 85%
  2. Regen. Flow(W1) = 1% of Soft Product(W6)
  3. Salt Usage(W12) = 2lb/kgal/100 ppm hard, Salt Cost = \$.03/lb
  4. Hard Ion in City Feed(W9) = 200 ppm
  5. City Feed Cost(W9) = \$2.00/kgal

From (2), (3), (4), and by the conservation of mass respectively,

$$y_1 = 0.01y_6$$

$$y_{12} = (2)(200ppm/100ppm)y_9 = 4y_9$$

$$y_6 + y_1 = y_9$$

$$\Rightarrow \quad y_6 + (0.01y_6) = y_9$$

$$\Rightarrow \quad y_9 = 1.01y_6$$

$$\Rightarrow \quad y_{12} = 4(1.01y_6) = 4.04y_6$$

$$\Rightarrow \quad \begin{bmatrix} y_1 \\ y_{12} \\ y_9 \end{bmatrix} = \begin{bmatrix} 0.01 \\ 4.04 \\ 1.01 \end{bmatrix} y_6 \stackrel{def}{=} ky_6$$

$$x_6 = k' \begin{bmatrix} x_1 \\ x_{12} \\ x_9 + 0.042941 \end{bmatrix} = \quad = k' \begin{bmatrix} 0 \\ 0.03 \\ 2 + 0.042941 \end{bmatrix}$$

- RESULT

  All the amounts of material flows and the related costs are in the *luc.rpt* report file.

  The Cost($/yr) is computed as follows:

$$\text{Cost(\$/yr)} = \text{Flow Rate} \cdot \text{Unit Cost} \cdot 365$$

  Important measures are tabulated as follows:

Product	Flow Rate(KGD)	Unit Cost($/Day/KGD)	Cost($/yr)
Soft Water(W6)	500	2.1846	398,690
DI(W26)	300	8.0309	879,384
UPW(W17)	800	10.052	2,935,184
		Total Cost	4,213,258

Part III
CONCLUSION

12 Conclusion

Good environmental performance measures are crucial to good environmental decision making. The measurability makes it possible to make the public be aware of environmental consequences, to set up the environmental goals to achieve, and to enforce such commitments. The environmental performance measure is the cornerstone of all environmental management systems(EMS).

Because the environmental systems are likely to be very large, complex, multi-disciplined, and often conflicting multi-objective, there is no unique method on how LCA should be done. Those difficulties are mentioned in [19]. Especially in the phase of interpretation, the judgement may be political.

Any quantitive environmental impact assessment can be used to measure the eco-efficiency in deciding whether a product, service, or process is greener than other alternatives. Greenness is a subjective term, however, identifying greenness may help to produce environmentally compatible products.

The goal of ECM is waste minimization through pollution prevention. Two components of Environmentally Conscious Manufacturing(ECM) are design and analysis and design of manufacturing strategies. Design of ECM systems requires quantitative tools to study the impact of alternative technologies, schedulers, materials and designs used.

Furthermore, to assist the environmental decision making, analysis, and

understanding of EMS, we need an effective modeling method to deal with large scale EMS while preserving the overall structure.

This dissertation is an attempt to develop an ECM tool based on MEB theory[38, 39, 37], to design a simulation language, to present a computer program instantiating a DfE tool called Mass-Energy Based Simulation Tool(MEBST).

Most quantitative DfE tools are concerned with environmental accounting system without concerning of feasibility and impact of process network structure. This dissertation is an attempt to answer such questions and to present a computer modeling and simulation tool, Mass-Energy Based Simulation(MEBS), instantiating an ECM tool(MEBST).

The unique features of this aproach are:

1. The MEBST is logical, mathematical, and has an expandable structure to model a system of various size and scale. Above all, processes are modeled based on physical parameters which does not change in terms of geographic location or different time, such as materials, energy cost(i.e. land, labor, and energy). Based on the sound physical and mathematical modeling, the MEBST can objectively assess environmental, economical, technological, network performances.

2. It is comprehensive and thus can be used for the entire life cycle of the product. This is important because of conflicting requirements between different life stages.

3. The models are based on fundamental principle of material energy and

balance.

4. It is computer based and is easy to use.

5. It allows the user the capability to perform sensitivity analysis. This will help to evaluate the impact of less accurate data on the outcome.

6. It allows "what-if" simulation capability

7. It helps to evaluate the impact of changes in processes and/or technologies(for example, the impact of automation or recycling).

All of these measures can be used for process improvement and management as shown in Figure 33.

Even if a small store managers do not use a simulation program, they are always drawing pictures in their minds, to maximize their profits using their best knowledge. This tool provides a graphical interface to evaluate these options rather easily.

One of the conclusions is that modeling is a formal representation of a system followed by simulation which assigns semantic meanings to its formal representation.

Because the framework of MEB modeling partitions an overall system into a tractable amount of processes and MEB standardized modules, MEB theory is capable of succinct, crisp, and structural modeling of complicated, very large system. By allowing the *library* class, a system can be constructed

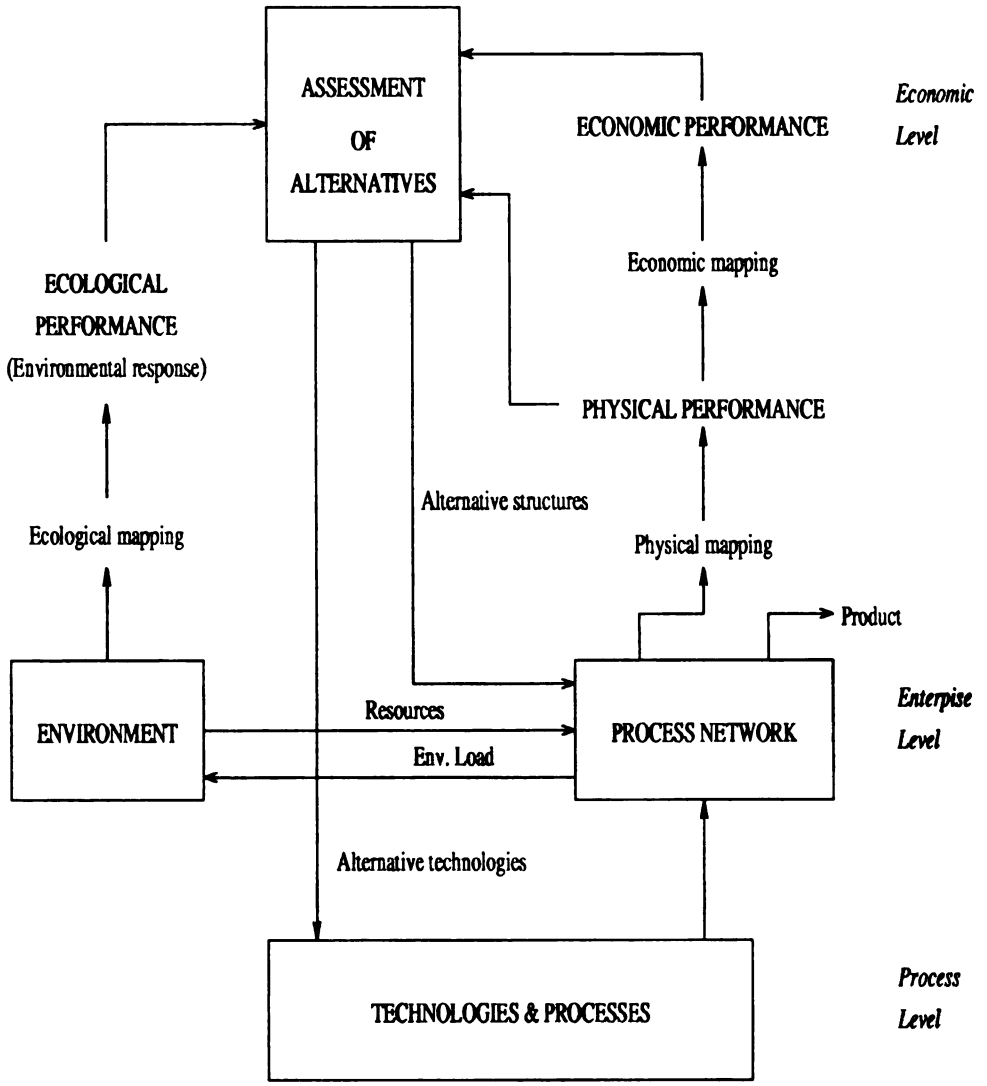


Figure 33: Integrated approach to manufacturing system analysis and design

in multi-layered structures along with a GUI capable of zoom-in and zoom-out presentation.

In the drawing course to partition a system, graphical interactions between the user and the MEB simulation system play an important role, more crucial than any other phase of simulation for a system to be understandable.

As is the one of a MEB simulation characteristics, the mix of top-down and bottom-up approaches makes modeling look a lot more like real world system because that is the basic nature involved in many design, analysis, and synthesis processes, though such characteristics might add more complexity to a system.

The ease of drawing graphical objects embedding MEB theory determines smooth riding over the important phase of modeling with less pain. With the MEB GUI, from partitioning a model and to seeing the results are but a few clicks of mouse button away. The ease and the simplicity of interactions make it possible for a user to focus only on defining a system or process boundaries in this phase of partitioning a large system.

After successful construction of the MEB graph, the semantics of the MEB graph are done through translating into MEBL and MEB Petri net. Each process solves its own problem using MEBL which can handle not only vector and matrix object expression but also interoperable database object succinctly with control flow statements.

As a results, this MEB tool can be used as highly complex information management system.

12.1 Contributions

For the realization of ECM, the proposed framework of MEB research and its methodology provides the following contributions:

1. Most importantly, the framework of MEB modeling and simulation tool which identifies related measures and processes to accomplish goals or specification of a system has been designed and implemented.

   With the proposed framework, measurement can range from a physically detailed description of raw material flow to an empirical view of environmental impacts of each life cycle stage.

   The framework is also useful for on-line evaluation of process improvement and management(Figure 33).

2. In order to express MEB theory, a powerful language - MEBL - which can deal with concurrent processes with composite data models such as vectors, matrices, interoperable database, and control flows has been designed and implemented.

3. A formal representation of MEB execution environment which inherently contains concurrency has been defined by using MEB Petri net.

4. MEB graph grammar providing a hierarchically structured multi-layered system modeling tool has been designed and implemented.

5. Grammar which translates MEB graphs into MEBL to express formulas

and to compute wire variables and MEB Petri net has been defined and implemented.

6. In order to determine the sequence of process execution due to parallelism embedded in a system, an algorithm has been developed to select a process to execute in the MEB Petri net execution environment.

7. Assuming user's minimal knowledge of MEBL, a GUI which painlessly guides the user through complex system modeling processes from drawing MEB graphs to viewing simulation results has been meticulously designed and implemented.

8. Reporting of MEB simulation results and 2-D data visualization have been implemented.

9. A variant of SQL which can query interoperable databases has been developed. Those databases may contain not only the environmental burden by each byproduct but also environmental impact categories and associated weight to aid in computing eco-indicator value considering the lack of unique measure of "how clean is green ?".

12.2 Future Direction

This research can be further improved in the following issues:

1. The framework provides the necessary information regarding the waste flows as a function of technology. The environmental performance of

these waste flow is heavily dependent upon the measures of impact. More research needs to be done to incorporate this incomplete and sometimes conflicting information to determine the environmental impact of a given technology.

2. With all the measurements available after construction of an MEB modeling and simulation, the next question is what to do with all those evidences.

   Any reasoning, validating, scientific judgement is based on those evidences which may lead to modification of a model, or different judgements.

   To make a judgement, possible decision categories need to be defined first. Then the decision problem would be assigning measurements to each of the categories. The next question is how each category is judged compared to other categories to quantify a global environmental burden.

   Combined with different weights for each categories, it seems to be next to impossible to find a unified formula to lead to a unique decision agreed upon by all the communities. Still, it would be nice having such a formula pleasing all the communities.

3. The database containing environmental load units needs to be filled with meaningful values agreed upon by environmental communities and scientists.

4. The framework does not provide any means of automatically selecting the optimal strategy from many of the strategies available. A feedback mechanism needs to be incorporated for this purpose. It may take a shape of an expert system.

5. The framework is implemented using Linux operating system. Transforming this to other operating system platform would be helpful.

Part IV
APPENDICES

LISTING

The translation of the MEB graph Figure 9.1 into MEBL is as follows:

```
wire {
        capa;
        cost;
        name;
}

netlist {
        W1: U2 -> J1;
        WW2: U1(W8) -> U3;
        WW3: U1(W9) -> U3;
        W4: U1(W10) -> U5;
        W5: U1(W11) -> U5;
        W6: U1(W12) -> U5;
        W7: U1(W13) -> S1;
        W8: U1(W14) -> S2;
        W9: J1 -> U3;
        W10: U3 -> U4;
        W11: U5 -> U6;
        W12: U6 -> U2;
        W13: S3 -> U2;
        W15: S4 -> U6;
        W16: U5 -> S5;
        WW1: J1 -> U1(W1);
}

init comment {
}

post elu {
        total_elu = 0;
        b = select elu from elu where (elu#material == W7.name);
        total_elu = total_elu + W7.capa*b;
        b = select elu from elu where (elu#material == W8.name);
        total_elu = total_elu + W8.capa*b;
        b = select elu from elu where (elu#material == W16.name);
        total_elu = total_elu + W16.capa*b;
        title("x: W10 vs. y: total_elu");
        plot(W10.capa, total_elu);
}

junction J1 {
```

```
        shape     J1;
        in        W1;
        out       W9;
        out       WW1;

        k = [ 1, 1; ];

        if (backward) {
                [ W1.capa; ] = k * [ W9.capa; WW1.capa; ];
        } else {
                [ W9.cost; WW1.cost; ] = k' * [ W1.cost; ];
        }
}

signal S1 {
        shape     none;
        in        W7;
}

signal S2 {
        shape     none;
        in        W8;
}

signal S3 {
        shape     none;
        out       W13;
}

signal S4 {
        shape     none;
        out       W15;
}

signal S5 {
        shape     none;
        in        W16;
}

class jj U1 {
        shape     U1;
        out       W4;
        out       W5;
        out       W6;
        out       W7;
```

```
        out         W8;
        in          WW1;
        out         WW2;
        out         WW3;
}

block 0 U2 {
        shape       U2;
        out         W1;
        in          W12;
        in          W13;

        k = [ 1; 1; ];

        if (backward) {
                [ W12.capa; W13.capa; ] = k * [ W1.capa; ];
        } else {
                [ W1.cost; ] = k' * [ W12.cost; W13.cost; ];
        }
}

block 0 U3 {
        shape       U3;
        out         W10;
        in          W9;
        in          WW2;
        in          WW3;

        k = [ 1; 1; 1; ];

        if (backward) {
                [ W9.capa; WW2.capa; WW3.capa; ] = k * [ W10.capa; ];
        } else {
                [ W10.cost; ] = k' * [ W9.cost; WW2.cost; WW3.cost; ];
        }
}

block 2 U4 {
        shape       U4;

        in          W10;
}

block 1 U5 {
        shape       U5;
```

```
        out         W11;
        out         W16;
        in          W4;
        in          W5;
        in          W6;


        kcost = [ 1, 1, 1; 1, 1, 1; ];
        k = [ 1, 1, 1; 1, 1, 1; ];

        if (backward) {
                [ W11.capa; W16.capa; ] = k * [ W4.capa; W5.capa;
                                                W6.capa; ];
                [ W11.cost; W16.cost; ] = kcost * [ W4.cost;
                                                    W5.cost;
                                                    W6.cost; ];
        }
}

block 3 U6 {
        shape       U6;
        in          W11;
        out         W12;
        in          W15;

        if (backward) {
                W15.capa = W12.capa - W11.capa;
                alpha1 = W11.capa./W12.capa;
                alpha2 = ones(size(alpha1)) - alpha1;
        } else {
                W12.cost = alpha1.*W11.cost + alpha2.*W15.cost;
        }
}
```

Mass-Energy Based Simulation User's Guide

A INTRODUCTION

The Mass-Energy Based Simulation(MEBS) tool is developed to evaluate environmentally conscious product designs, management of manufacturing facilities to evaluate the strategies for reducing waste flows into the environment, and life cycle analysis on the Linux platform.

This tool allows to input the description of the main structure of a plant using a drawing pallet. This pallet contains built-in drawing buttons in a graphic user interface(GUI) implemented on the X-Window environment with X11. The GUI relieves the user from having to know all the mathematical details of the models which describe each process within a plant and the interconnection constraints associated with the structure of the plant or a process.

Besides having features to represent network information succinctly, MEBS also introduces the Mass-Energy Based Simulation Language(MEBL) which borrows many aspects from C language, MATLAB[2], and SQL database language. A source program is automatically created by the user with the GUI. Followings are case-by-case examples which will illustrate the details of the program.

[2]MATLAB is a trademark of Math Works Inc.

B GETTING STARTED

B.1 Convention

<name> denotes the command button with the name embedded between angled brackets and executed by clicking the left mouse button once. Similarly, **<name1>, <name2>,** ... denotes a sequence of command buttons.

Usually the left mouse button is interpreted as selection operation and the right button as ESC key. The double click of the left mouse button is interpreted as the RETURN key or equivalent to a click of the middle button on a three button mouse.

B.2 Overview

A network is described with several types of building blocks such as *production*, *junction*, *library*, and *goal* blocks, and wires which connect the blocks together. The forward connections are done by all the types of blocks except the *recycle* type block while backward feedback connection uses only the *recycle* type block as a subprocess. The special *storage* type block is used when the backward connection feeds to a forward connected process block to form a feedback connection.

MEBS has a command name of *meb*. The detail command line options are described in Section G. There are buttons in the GUI of *meb* for drawing a plant, modification of the subprocess description, simulation, and the report of simulation results.

With the specific command **meb**, a large empty canvas shows up for draw-

ing subprocesses connected by directed lines or wires. The basic building blocks to describe a plant are rectangles of type **<Production>**, **<Recycle>**, **<Storage>**, **<Junction>**, **<Library>**, and **<Goods>** for a subprocess which are interconnected by **<Polyline>** and **<Wire>**. It should be noted that at least a block of type **Goods** be included in a network to specify the desired flow rate of final goods. **<Text>**, **<Comment>**, and **<Label>** buttons together are used to label the input and output of the subprocesses and later used for the report of simulation result. All other drawing primitives such as **<Polygon>**, **<Spline>**, and **<Circle>** are available.

There are four kinds of files used to model a plant. The first one is the figure file which describes the network of the plant. The figure file is later used to provide the relationships of subprocesses and it contains all geometric information of the subprocesses on the canvas. The second file is the plant file which has all the information to describe a plant except the geometric information to draw on the canvas and it is created by the **<Netlist>** button. The suffix of the file explicitly explains what kinds of files they are; i.e. a file name with the suffix ".fig" implies that the file contains figures, and the file name with the suffix ".pl" implies that the file is a type of plant file. The file name with the suffix ".rpt" may also be created to store the simulation results with the **<Report>** button.

To address environmental impact assessment, an environmental index is assigned to each type of material used in a plant. And all of the indices used in a plant are stored into the default database named "elu.db". A quantitive

measure of greeness of a plant is computed with the **<ELU>** button. **<File>**, **<Save>** will save a figure file after drawing a graph while **<Plant>**, **<Save>** will save a plant file generated by **<Netlist>** button. The plant file contains MEBL to simulate a plant.

After a network description is completed, we begin the simulation with the **<Netlist>** command button. This will generate a sample source program in MEBL in the framework of the MEB model. For the modification of the program in any subprocess, click the left mouse button on the subprocess which needs to be modified. There are four parameters which can be modified through GUI. They are *Technical Coefficients*, *Capa*, *Cost*, and *By-product cost*.

Once a modeling by MEB graph is done, the next step is to simulate by specifying **<Goal>**. The goal is specified by selecting the output nodes associated with *Goods* and providing input costs. If the beginning of the terminal node is selected, **meb** will ask you to input the unit cost. When the goals and the material unit costs have been established, simulation is initiated with a press of the mouse middle button.

The simulation results may be visualized with the **<Get Value>** button. This will plot a 2-D graph with the flow rate on the X-axis and the unit cost on the Y-axis. There is also a **<Plot>** button to obtain an x-y plot of any two arbitrary data. Finally all the simulation results can be stored in *file.rpt* by pressing the **<Report>** button.

B.3 Example

As an illustration, consider a plant with three *Production* blocks and two *Goods* block as shown in Figure 34.

B.3.1 Drawing an Example Plant

- Invoke simulation program **meb**.

  1. **meb**

- Use **<Production>** to create rectangular type production blocks

  1. Production
  2. Select two points to determine the size and position of rectangular production block with mouse.
  3. Copy Object
  4. The above button is used to duplicate the rectangular boxes that represent production processes.
  5. Place them at different positions on the canvas to create three more production blocks.

- Create **<Goods>** type rectangular blocks

  1. Final Prod
  2. Similarly create two *Goods* blocks with mouse.

- Connect the blocks with the **<Wire>** button.

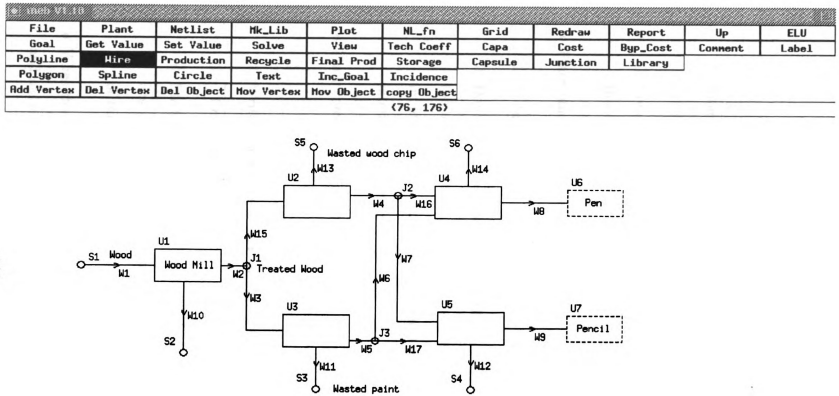


Figure 34: An example plant

1. Wire or Polyline

2. Interconnect blocks with mouse. To complete a line segment, click the middle mouse button or double click the left mouse button.

3. Complete the rest of interconnections between blocks following the same method.

- Place junctions on lines if necessary.

1. Junction

2. Double click the left mouse button or click the middle mouse button at the junction of lines. Note that there should be only one input line into a junction. This junction represents the points of interaction between the processes.

- Place comments on lines and blocks.

1. Text

2. Write description of lines or blocks if necessary.

3. Comment

4. First, select a line to be associated with comments followed by a description with double click of the left mouse button or a click of the middle mouse button.

- When the drawing is done, save the drawing using File Save as and name the file as *ex1.fig*.

Figure 35 describes a procedure to draw the plant where the numbers beside the boxes of the process represent the sequence of operations in drawing the plant. The corresponding mouse operation for each user input number is also shown at the bottom of Figure 35.

B.3.2 Simulation of an Example Plant

- Generate a simulation program.

  1. Netlist

  2. The result would look like Figure 34.

- Provide information for the good.

  1. Goal

  2. Select blocks of *Goal* type to specify the desired flow rate of final goods. Pop–up window will appear for input like Figure 36

  3. Select beginning of lines to provide unit cost of materials which are fed into the production blocks. Similarly, a pop–up window will appear for input like Figure 37

  4. Move the mouse on empty space and double click the left mouse button or click the middle mouse button.

  5. If everything goes right, The "*All Solved*" message will appear on the status window above the canvas. Otherwise, the blocks which are not solved will be displayed on the status window.

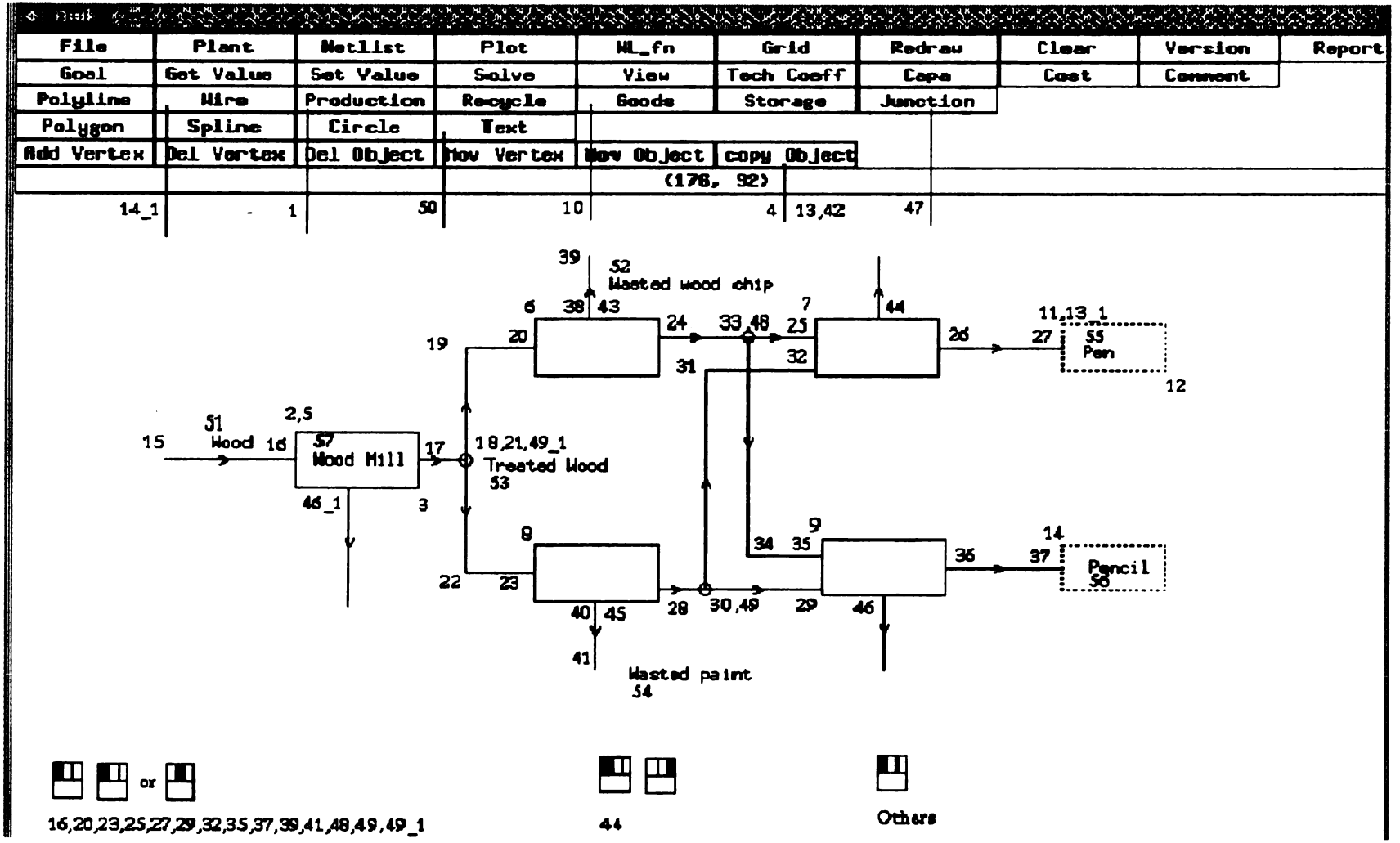


Figure 35: A Procedure to draw a plant

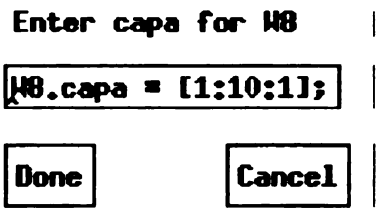


Figure 36: Pop–up capacity query window

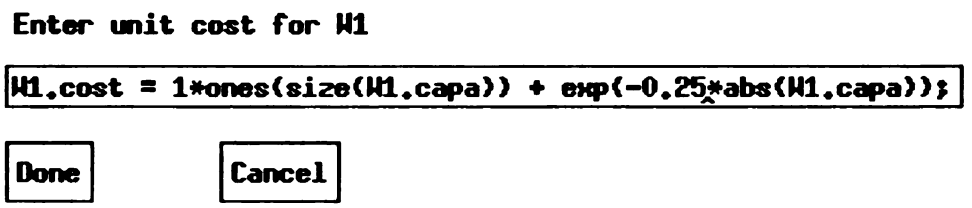


Figure 37: Pop–up cost query window

Figure 38 describes a procedure to simulate the plant.

- To change values of technical coefficients push [Tech Coeff] and select a subprocess. Then a query will appear to facilitate the modification of technical coefficients. If we want to save modifications for later reference, save the results with the drag submenu button [Plant] [Save]. The file will be stored as *ex1.pl*

- Examination of simulation results

  1. [Get Value]
  2. Select variables to visualize the simulation result. An example 2D plot is shown in Figure 39.
  3. Or use [Plot] button.

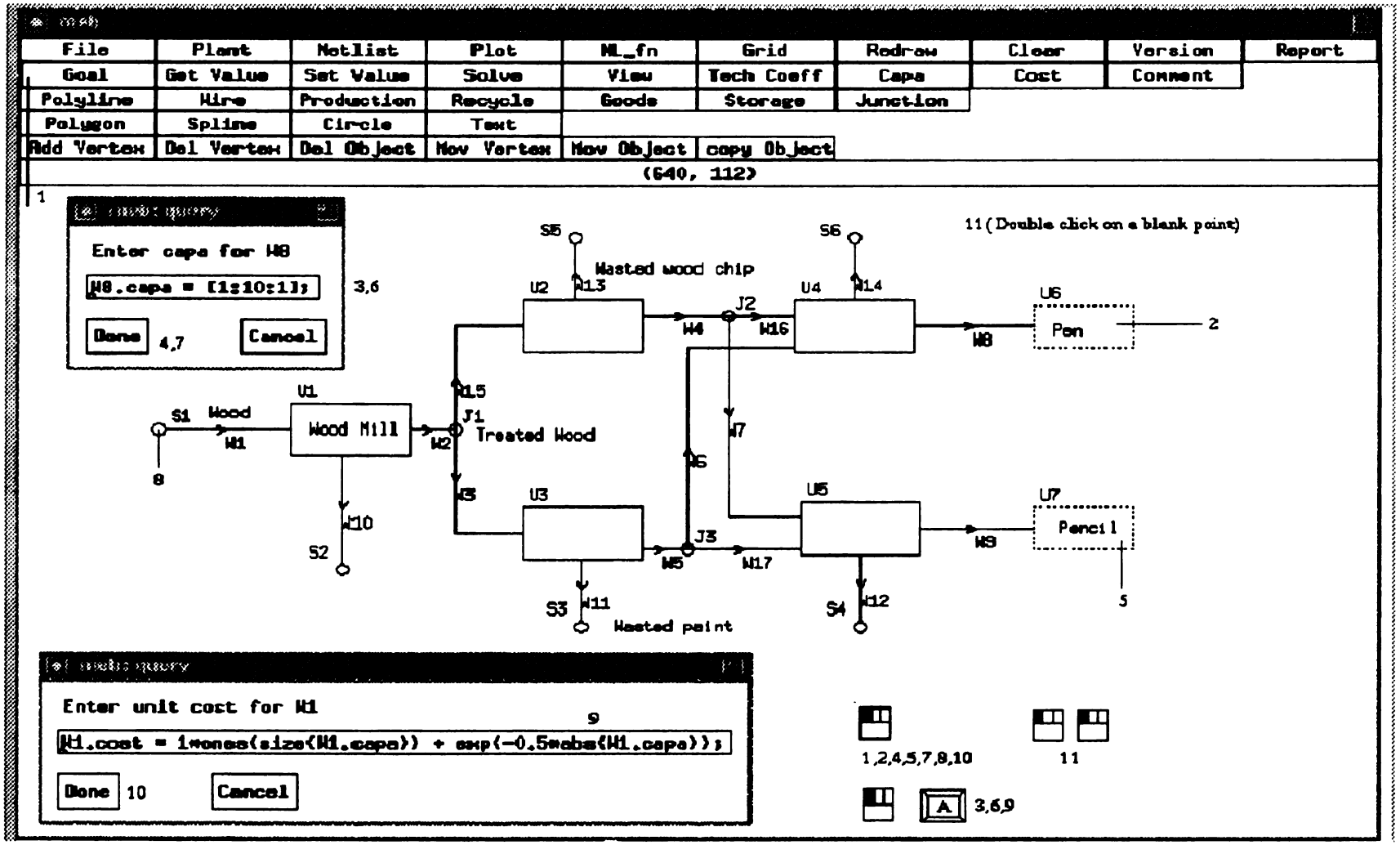


Figure 38: A procedure to simulate the plant

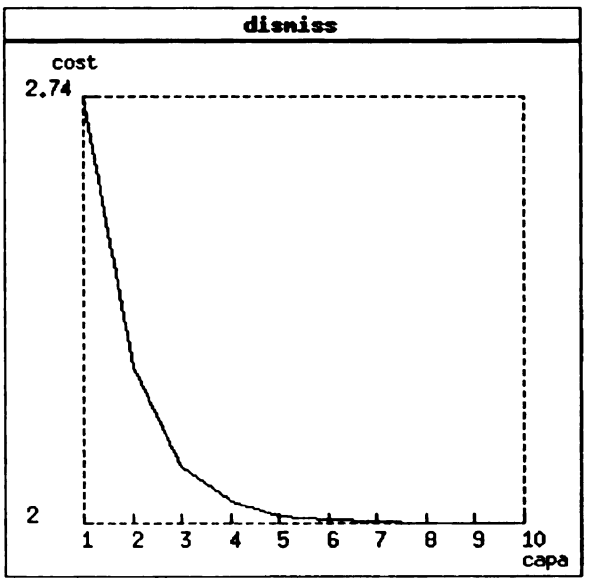


Figure 39: Capacity vs. cost 2-D plot

4. [Report] button will save all the simulation results in *ex1.rpt* file.

B.3.3 Defining Library

A *library* convention is used to represent a big plant in the limited size of a GUI screen. Assuming that a plant description is done and saved as *part2.fig* and *part2.pl*, a library part is saved as *lib_part2.fig* and takes the following steps:

- Change the GUI's current mode into the library creation mode by selecting the [Make Lib.] submenu button in the [Mk_Lib] menu.
- Define the shape of the library part and associate *wire* variables with its names.

1. [Capsule] allows the user to draw the boundary of a library part for interaction with the rest of the plant.
2. Select two points to determine the size and position of the *Capsule* type rectangle block with mouse.
3. Draw smaller *Production* type rectangular inside the *Capsule* type block.
4. Draw wires between the *Capsule* and *Production* type blocks which are used to interact with the rest of a plant.
5. Associate the newly created wire with the same name used in *par2.pl* with the [Label] button.

- Define the incident points where interaction occurs between the library part and the rest of a plant.
  1. The incident points are defined by the wire direction which is drawn above and the incident markers of [Inc_Goal] and [Incidence].
  2. If the incident point is either an input to the library part or a byproduct of a library part, choose the [Incidence] button, and click the left mouse button at the incident point. Otherwise, choose the [Inc_Goal] button and and click the left mouse button at the incident point.

- Save the resultant drawing with the name em lib_part2.fig by selecting the [Save As] submenu button in the [Mk_Lib] menu.

- Finish the defining library part by selecting the [Use Lib.] submenu button in the [Mk_Lib] menu.

- Using a library block is similar to using the normal building block. Select the [Library] button and select a position where the library is to be located. When asked to enter a library name, just enter *part2* omitting the prefix *lib_* and the suffix.*fig*.

Figure 40 describes a procedure to define the library part *lib_part2.fig* where the numbers beside the boxes of the process represent the sequence of operations in drawing the plant.

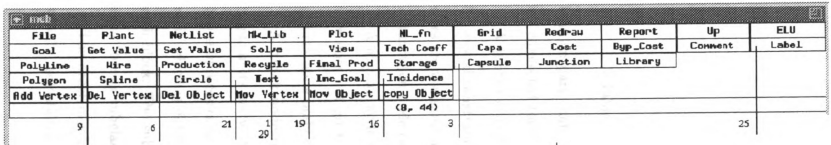
C MEBL SYNTAX

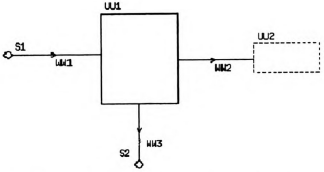
C.1 Plant Structure

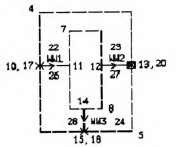
The plant structure consists of two types of declarations and a series of the node blocks. There are five kinds of visible node blocks: **junction** node, **signal** node, **block** node, **recycle** node, **storage** node, and two kinds of special blocks: **init** node and **post** node.

If the plant structure is to be saved, it will be saved in the form of a plant file with the suffix of the file name ending with **.pl**. An example MEBL program corresponding to Figure 34 generated by **<Netlist>,<Plant>,<Save>** is given in Section I.

The first part of a plant description begins with the declaration of members of **wire** structure followed by the **netlist** keyword. The **netlist** dec-







The left side plant, part.pl, will not be shown in defining library part process.

2. <Make Lib.>

26,27,28. Type the name of wires.

30. <Save as> and type "lib_part2.fig" at query window

Figure 40: Procedure to define a library part

Block type	Subclass No.
Production	0
Recycle	1
Goods	2
Storage	3
Libray	4

Table 27: Block node subclass number used in MEBL

laration part contains all the network information about how subprocesses are connected together.

A `junction` node is also considered as a special subprocess which abides by special constraints. In the framework of MEBS, the sum of output flow rates are the same as the input flow rate and their unit costs are the same. A `junction` node can take only one input with arbitrary number of output.

A `signal` node is created where either the beginning or the end of the wire does not go to any subprocess. There is nothing to compute in the `signal` node body except the initial condition provided in `<Goal>`.

A `block` node represents a subprocess and contains appropriate statements in its body. Accordingly, there are two `block` nodes corresponding to two subprocesses in this example. The `init` node is a special kind of block node which is done first once before execution of functions associated with each block. The subclass number of the block used in an MEBL statement as in "`block` subclass_number *block_name* { $\cdots$ }" is shown in Table 27.

The order of which node is invoked first depends on the goal set-up in `<Goal>`, solvability of the subprocess, and the network of a graph. First,

the simulation solves for flow rates noted by *capa* for every block by back propagation from the goal. Then it solves for *cost* for each block by forward propagation from the input material costs which were given in the **<Goal>** button.

Finally, after all other solutions of processes are finished, the special **post** block - which might be executed as an option by **<ELU>** - is available to evaluate environmental impact assessment along with environmental index database as in Section E.1.

C.2 Netlist

Again considering the above example, the **netlist** body consists of the list of the wires with its source node to the left and its destination node to the right. For example,

```
W1: S1 -> J1;
```

implies that the wire **W1** goes from the signal node **S1** to junction node **J1**. Note that **->** has a different meaning from the meaning in the C language.

C.3 Node Structure

Considering that every subprocess has an associated building block of rectangular shape in the graph, the **shape** declaration begins first. For the sake of readability, the rest of the declarations indicate which wires are coming in and which ones are going out of a subprocess.

After the declaration part, the statements of a node which determine the behavior or the function of a subprocess begin. It is worth mentioning that any **wire** variables can be accessed in any nodes. However, considering that the messages coming in and going out of a subprocess are highly correlated to the network of subprocesses, it seems to be a good practice to access only the **wire** variables which are in contact with the node. With this limitation of choice in the practice, it helps the program to be modular and structured.

C.4 Wire Variables

A wire variable is a composite object consisting of flow rate *capa*, energy cost *cost*, and a label *name* of wire itself.

The wire variable name followed by period and one of *capa, cost, name* refers to simple data types of either real value or string.

C.5 Read Only Variables

There are two reserved read-only global variables such as **backward** and **forward**. Simulation goes through the phases of backward and forward computation. Those special variables show the direction of traverse during simulation so that each subprocess can define its segment of a program to be computed. Users can read those values, but users are not allowed to set the values of those special variables.

C.6 Constants

To accommodate the frequently used symbol π in trigonometry functions, a specific symbol `PI` is reserved for a constant. And the character constant is a character between single quotation marks while the string constant like **`"this is string"`** is a list of characters within double quotation marks. Both the character and the string constants are accepted as in C language. Some invisible characters are represented by escaping as follows:

`\n`	newline
`\t`	tab
`\f`	form feed
`\\`	back slash

For the numerical representation, both decimal and hexa-representation are accepted for an integer value. For example, the decimal number **10** is equal to the hexa-number **`0xa`**. For the floating number representation, only decimal numbers are allowed as in the following examples:

$$.1234 \quad 1.234 \quad 12.34E5 \quad 123.4e-5$$

There are two reserved variables, **`backward`** and **`forward`**. Those variables show the direction of traverse during simulation so that each subprocess can define a specific process in MEBL, depending on the state of traversal.

D CONTROL FLOW

The control flow is similar to that of the C language except that the expression body between the control flow keywords to be selected or iterated

should be enclosed by { and }, even though the expression body contains only a statement.

D.1 If Else

This control flow keywords select a specific part of the body separated by `if` and `else` keywords for computation. Every expression body between `if` and `else` should be within { and } even when the body has only a line statement.

D.2 For

```
for (expr1; expr2; expr3) {
        statements;
}
```

The above `for` expression has all the control conditions which are optional in one line followed by `for`. *expr1* is the initial condition before any other expression related to the `for` is considered for computation. If *expr2* is true, the main body within { and } is computed followed by computation of *expr3* and *expr2* to make a full cycle again for the next iteration. Otherwise, the main body is skipped from computation.

D.3 While

The control flow `do ... while` in C language is not available yet.

```
while (expr1) {
      statements;
}
```

The statements in the **while** body are computed repeatedly, as long as the *expr1* is true when tested before the execution of the **while** body.

D.4 Loop Control

break terminates the smallest enclosing loop by **for** and **while**. **continue** returns the next computation immediately to the smallest enclosing **while** or **for** control statement. **goto** *"identifier;"* renders the next computation to be the statement after the label *identifier*. An identifier followed by : is considered to be a label or address in a process program.

D.5 Comments

A comment is a list of characters between /* and */ in a line and ignored from computation.

E DATABASE

MEBL has a statement similar to SQL for simple database manipulation. A database table can be managed by a normal text editor because it contains only the plain ASCII text.

E.1 Database Structure

Four keywords which explain database itself used in the database are:

version requires only one argument telling a version number.

delim describes the delimiter between fields.

field requires three arguments. The first argument is the field name and the second argument describes the type of field. There are only two types; *c* implies character type and *f* implies floating number type. The third argument is the maximum field size.

Record delimiter is set to the new line character.

end implies the end of the head information.

An example of a database file is as follows:

```
version   1
delim             ,
field             material    c         20
field             ec          c         4
field             elu         f         10
end

Co,               RM,         76
Cr,               RM,         8.8
Fe,               RM,         0.09
Mn,               RM,         0.97
Mo,               RM,         1.5E3

CO2,              EA,         0.09
CO,               EA,         0.27
SOx,              EA,         0.10
CFC-11,           EA,         300
CH4,              EA,         1.0

Nitrogen,         EW,         0.1
Phosphorus,       EW,         0.3
```

E.2 Database Statement

The syntax for the database statement is of the form:

select *field_name* **from** *database* **where** *query*

The *database* uses the name omitting the suffix `.db` from the database file name.

To specify a field `material` in the database *elu*,

elu#material

is allowed in the *query* statement.

F EXPRESSION

F.1 Matrix

The elements of the matrix are separated by either a comma or a semicolon. A semicolon is for the change to the next row of a matrix while a comma is for delimiting elements column-wise. For example, `a = [ 1, 2, 3; 4, 5, 6] ;` implies 2x3 matrix. Larger matrices can be generated by using variables as shown in the following example:

```
a = [1, 2; 3, 4];
b = [5; 6];
c = [7, 8, 9];
```

A 3x3 matrix d is generated by using a, b, c as follows:

```
d = [a, b; c];
```

will construct d matrix to be three by three matrix resulting in

$$\begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

F.2 Vector

Two or three elements are needed to represent a range of values as follows:

`[expr1: expr2]` or `[expr1: expr2: expr3]`.

The first element **expr1** is the value to start from and the second element **expr2** is the final value of a vector. The third element determines the step size for the next element to generate. If the third element is missing, one is used for the default step size. For example `[0:10:2]` will generate a vector [0, 2, 4, 6, 8, 10].

F.3 Operators

Operators and their precedences are shown in Table 28.

F.4 Output Functions

prval(a) Displays the value of a in the message window.

plot(x, y) Draws x-y plot on a pop-up window.

title(s) Sets the title message of a pop-up drawing window to s.

F.5 Math Functions

det(a) Determinant of the square matrix a.

inv(a) Inverse of the square matrix a.

`+a`	Positive of a
`-a`	Negative of a
`!a`	Negation of a
`a'`	Transpose of a matrix a
`a * b`	Multiplication of a and b
`a / b`	Division of a by b
`a .* b`	Element-wise multiplication
`a ./ b`	Element-wise division
`a + b`	Sum of a and b
`a - b`	Subtraction of a by b
`a > b`	Greater than
`a >= b`	Greater than or equal to
`a < b`	Less than
`a <= b`	Less than or equal to
`a == b`	Equal to
`a != b`	Not equal to
`a && b`	a and b
`a || b`	a or b
`a = b`	Assignment

Table 28: Precedences of operators

diag(a) If a is a matrix, diag(a) is the main diagonal vector. Or if a is a vector, diag(a) creates a square matrix with the diagonal elements the same as a and off-diagonal elements zeros.

size(a) Returns the number of rows and the number of columns.

eye(a) Returns an identity matrix with the same size of a.

zeros(a) Returns a matrix of the same size of a with its elements zeros.

ones(a) Returns a matrix of the same size of a with its elements ones.

exp(a) Returns a matrix of the same size of a with its elements exponential of the elements of a.

ln(a) Returns a matrix of the same size of a with its elements natural logarithms of the elements of a.

log(a) Returns a matrix of the same size of a with its elements base ten logarithms of the elements of a.

cos(a) Returns a matrix of the same size of a with its elements cosine of the elements of a.

sin(a) Returns a matrix of the same size of a with its elements sine of the elements of a.

tan(a) Returns a matrix of the same size of a with its elements tangent of the elements of a.

acos(a) Returns a matrix of the same size of a with its elements inverse cosine of the elements of a.

asin(a) Returns a matrix of the same size of a with its elements inverse sine of the elements of a.

atan(a) Returns a matrix of the same size of a with its elements inverse tangent of the elements of a.

cosh(a) Returns a matrix of the same size of a with its elements hyperbolic cosine of the elements of a.

sinh(a) Returns a matrix of the same size of a with its elements hyperbolic sine of the elements of a.

tanh(a) Returns a matrix of the same size of a with its elements hyberbolic tangent of the elements of a.

G USER'S REFERENCE MANUAL

NAME

meb - Mass-Energy Based simulation tool

SYNOPSIS

meb [*file.fig* | *file.pl*]

DESCRIPTION

There are two kinds of files to model a plant. One of them is the figure file which describes the network of subprocesses in a plant. The figure file is later used to describe the relationship of subprocesses and only contains the geometric information of the subprocesses on the canvas. The other kind of file is the plant file which has all the information to describe a plant except the geometric information to draw on the canvas. The suffix of the file explicitly explains what kinds of files they are i.e. the suffix ".fig" implies a figure file and the suffix ".pl" implies the plant file. The file name with the suffix ".rpt" contains all the simulation results.

If **meb** is invoked without a file name, a new canvas shows up with the default figure file name "untitled.fig" and with the default plant file name "untitled.pl".

For the graphic user interface, the left mouse button is usually interpreted as selection operation and the right button as ESC key. The

double click of the left mouse button is interpreted as RETURN key or as equivalent to a click of the middle button. When the left button is pressed, a rubber band appears on the canvas to show what a consequence would be. If that is a right choice, then the choice can be confirmed by the second press of the left button.

AUTHOR

Youngsun Chun

email: chun@pilot.msu.edu

H MEBL GRAMMAR

program
wire netlist nodes

wire
`wire` { wirebody }

netlist
`netlist` { netbody }

nodes
nodes node
node

wirebody
wirebody identifier ;
identifier ;

netbody
netbody netelem
netelem

netelem
identifier : identifier -> identifier ;

node
classhead { body }

classhead
`init` identifier
`post` identifier
`block` block_type identifier
`junction` identifier
`signal` identifier

body
declarations statements

declarations
declshape declportvar

declshape
`shape` identifier ;
`shape none` ;

declportvar
declportvar portvar
portvar

portvar
in identifier ;
out identifier ;

statements
statements $astat_{opt}$

astat
expression
if (expression) { statements }
if (expression) { statements } else { statements }
while statements
for (forexpr) { statements }
identifier : statements
goto identifier ;
continue
break
return

forexpr
$expression1_{opt}$; $expression2_{opt}$; $expression3_{opt}$

expression
mathfunction (expression)
title (string)
plot (expression, expression)
prval (expression)
term
+ expression
- expression
! expression
expression'
expression binop1 expression
expression binop2 expression
matrix = expression
lval = expression lval = dbstmt

dbstmt
select identifier **from** identifier **where** expression1

mathfunction
det inv exp diag size eye zeros ones
ln log

```
      cos sin tan
      acos asin atan
      cosh sinh tanh
```

term
      (expr)
      matrix
      range
      dbval
      lval
      identifier
      PI
      constant

constant
      integer
      real
      string
      backward
      forward

matrix
      [matrest]
      [rows]

matrest
      matrest rows
      matrest rows ;
      rows ;

rows
      rows comma expr4
      expression binop1 expression

range
      [expr ; expr]
      [expression ; expression ; expression]

dbval
      identifier # identifier

lval
      identifier . identifier
      identifier

binop1

```
.*
./
*
/
```

binop2

```
+
-
>
>=
<
<=
==
!=
&&
||
=
```

I AN MEBL PROGRAM EXAMPLE

```
wire {
        capa;
        cost;
        name;
}

netlist {
        W1: S1 -> U1;
        W2: U1 -> J1;
        W3: J1 -> U3;
        W4: U2 -> J2;
        W5: U3 -> J3;
        W6: J3 -> U4;
        W7: J2 -> U5;
        W8: U4 -> U6;
        W9: U5 -> U7;
        W10: U1 -> S2;
        W11: U3 -> S3;
        W12: U5 -> S4;
        W13: U2 -> S5;
        W14: U4 -> S6;
        W15: J1 -> U2;
        W16: J2 -> U4;
        W17: J3 -> U5;
}

init comment {
        W11.name = "Wasted paint";
        W13.name = "Wasted wood chip";
}

post elu {
        total_elu = 0;
        b = select elu from elu where (elu#material == W10.name);
        total_elu = total_elu + W10.capa*b;
        b = select elu from elu where (elu#material == W11.name);
        total_elu = total_elu + W11.capa*b;
        b = select elu from elu where (elu#material == W12.name);
        total_elu = total_elu + W12.capa*b;
        b = select elu from elu where (elu#material == W13.name);
        total_elu = total_elu + W13.capa*b;
        b = select elu from elu where (elu#material == W14.name);
```

```
        total_elu = total_elu + W14.capa*b;
        title("x: W14 vs. y: total_elu");
        plot(W14.capa, total_elu);
}

junction J1 {
        shape   J1;

        out     W15;
        in      W2;
        out     W3;

        k = [ 1, 1; ];

        if (backward) {
                [ W2.capa; ] = k * [ W15.capa; W3.capa; ];
        } else {
                [ W15.cost; W3.cost; ] = k' * [ W2.cost; ];
        }
}

junction J2 {
        shape   J2;

        out     W16;
        in      W4;
        out     W7;

        k = [ 1, 1; ];

        if (backward) {
                [ W4.capa; ] = k * [ W16.capa; W7.capa; ];
        } else {
                [ W16.cost; W7.cost; ] = k' * [ W4.cost; ];
        }
}

junction J3 {
        shape   J3;

        out     W17;
        in      W5;
        out     W6;

        k = [ 1, 1; ];
```

```
        if (backward) {
                [ W5.capa; ] = k * [ W17.capa; W6.capa; ];
        } else {
                [ W17.cost; W6.cost; ] = k' * [ W5.cost; ];
        }
}

signal S1 {
        shape   none;

        out     W1;
}

signal S2 {
        shape   none;

        in      W10;
}

signal S3 {
        shape   none;

        in      W11;
}

signal S4 {
        shape   none;

        in      W12;
}

signal S5 {
        shape   none;

        in      W13;
}

signal S6 {
        shape   none;

        in        W14;
}

block 0 U1 {
```

```
        shape   U1;

        in      W1;
        out     W10;
        out     W2;

        W10.cost = zeros(size(W2.capa));

        k = [ 1; 1; ];

        if (backward) {
                [ W1.capa; W10.capa; ] = k * [ W2.capa; ];
        } else {
                [ W2.cost; ] = k' * [ W1.cost; W10.cost; ];
        }
}

block 0 U2 {
        shape   U2;

        out     W13;
        in      W15;
        out     W4;

        W13.cost = zeros(size(W4.capa));

        k = [ 1; 1; ];

        if (backward) {
                [ W13.capa; W15.capa; ] = k * [ W4.capa; ];
        } else {
                [ W4.cost; ] = k' * [ W13.cost; W15.cost; ];
        }
}

block 0 U3 {
        shape   U3;

        out     W11;
        in      W3;
        out     W5;

        W11.cost = zeros(size(W5.capa));

        k = [ 1; 1; ];
```

```
        if (backward) {
                [ W11.capa; W3.capa; ] = k * [ W5.capa; ];
        } else {
                [ W5.cost; ] = k' * [ W11.cost; W3.cost; ];
        }
}

block 0 U4 {
        shape   U4;

        out     W14;
        in      W16;
        in      W6;
        out     W8;

        W14.cost = zeros(size(W8.capa));

        k = [ 1; 1; 1; ];

        if (backward) {
                [ W14.capa; W16.capa; W6.capa; ] = k * [ W8.capa; ];
        } else {
                [ W8.cost; ] = k' * [ W14.cost; W16.cost; W6.cost; ];
        }
}

block 0 U5 {
        shape   U5;

        out     W12;
        in      W17;
        in      W7;
        out     W9;

        W12.cost = zeros(size(W9.capa));

        k = [ 1; 1; 1; ];

        if (backward) {
                [ W12.capa; W17.capa; W7.capa; ] = k * [ W9.capa; ];
        } else {
                [ W9.cost; ] = k' * [ W12.cost; W17.cost; W7.cost; ];
        }
}
```

```
block 2 U6 {
        shape   U6;

        in      W8;
}

block 2 U7 {
        shape   U7;

        in      W9;
}
```

J About the CD Rom

Besides the DfE tool, *meb*, the enclosed CD Rom contains the system modelings described in the Section 11 and a hypothetical model to show how to measure the environmental impact of a manufacturing plant.

A system model is composed of four parts as follows:

Graphical description describes plant structure and shows how processes interact together in a plant. File name ends with the suffix *.fig.*

Textual description which is automatically translated from above plant structure describes how to solve each process and integrate those individual solution into various measures to assess a plant in various perspective view. File name ends with the suffix *.pl.*

Environment description describes measures which are provided from outside a system boundary. File name ends with the suffix *.inp.*

Environmental Load Unit(ELU) database stores measures which are agreed upon by environmental scientists. These values define environmental impacts of various kinds of materials in various forms. Default file name of ELU database is *elu.db.*

J.1 System Requirements

At least physical or virtual display size of 800 by 600 pixels is required in X window configuration.

The first generation of this DfE tool *meb* version 1.18 has been tested under following environments:

- Linux Version 2.0.32
- X Window System Version 11

J.2 Installation

The DfE tool, *meb*, can be anywhere as long as the full path name of *meb* is reachable by $PATH shell environment variable. However, since *meb* needs to write on current working directory, system modeling files which are on CD Rom should be copied to hard disk which has writing permission on it.

Assuming that the enclosed CD Rom is mounted under the directory **/cdrom**, following procedures will install the *meb* package under new directory **meb18**.

```
$ /cdrom/setup
```

For those who are not ready for running Linux operating system and X window system, all the MEB modeling files and the simulation results are stored under **/cdrom/meb18** directory ready to be viewed by any operating system, although simulation cannot be done on other operating system than Linux.

Following is an illustration of the *setup* procedures.

```
Thank you for trying Mass Energy Based(MEB) DfE tool.

Please let me know where to find your CD Rom(/cdrom)?
Please let me know where to install this package(/u1/chun)?  /tmp
I am about to install the MEB package under the /tmp/meb18
from the /cdrom.

Are you ready ([y]es, [n]o)? y

Installation completed.

For an example of a MEB modeling try followings.

            $ cd /tmp/meb18
            $ meb farm.pl

For more explanation of included MEB modeling examples,
please read /tmp/meb18/README.pl
```

I hope you to enjoy the DfE tool *meb*.

References

[1] A.A.Jensen, J.Elkington, and five others. Life cycle assessment(lca)/a guide to approaches, experiences, and information sources. Technical report, dk-TEKNIK Energy & Environment, 15 Gladsaxe Mollevej, Soborg, Denmark, 1997.

[2] A.R.Tilma, C.B.Tilma, and E.C.Alocilja. Process network theory(pnt) and analysis of a swine/crop system. *American Society of Agricultural Engineers Meeting Presentation*, 90(7577), 1990.

[3] R.A. Baldwin. *A Discipline Independent Framework for Engineering Design.* PhD thesis, Michigan State University, 1994.

[4] B.Kassahan, M.Saminathan, and J.Sekutowski. Green design tool. *IEEE Symposium on Electronics & Environment*, pages 118–125, 1995.

[5] B.S.Blanchard and W.J.Fabrycky. *Systems Engineering and Analysis.* Prentice-Hall, 1981.

[6] B.Steen and S.Ryding. The epa enviro-accounting method: An application of environmental accounting principle for evaluation and valuation of environmental impact in product design. Technical report, Stockholm: Swedish Environmental Research Institute(IVL), 1992.

[7] C.Hall and J.Day. *Ecosystem Modeling in Theory and Practice.* Wiley Interscience, 1977.

[8] Chen. An approach for material selection that integrates mechanical design and life cycle environmental burdens. *IEEE Symposium on Electronics & Environment*, pages 68–74, 1995.

[9] D.F.Ciambrone. *Environmental Life Cycle Analysis.* Lewis, 1997.

[10] D.J.Richards. *The Industrial Green Game.* National Academy Press, 1997.

[11] D.P.Smolik. *Material Requirements of Manufacturing.* Van Nostrand Reinhold Co., 1983.

[12] E.C.Pielou. *Population and Community Ecology: Principles and Methods.* Gordon and Breach, 1974.

[13] E.Henley and H.Kumamoto. *Designing for Reliability and Safety Control.* Prentice-Hall, 1985.

[14] J. Fiksel. *Design for Environment.* McGraw-Hill, 1996.

[15] M. Goedkoop. *The Eco-indicator95, Final report*, 1995. (URL:http://www.pre.nl/eco-ind.html).

[16] G. Gordon. *System Simulation.* Prentice-Hall, 1978.

[17] T.E. Graedel and B.R. Allenby. *Industrial Ecology.* Prentice-Hall, 1995.

[18] T.E. Graedel and B.R. Allenby. *Design for Environment.* Prentice-Hall, 1996.

[19] Y.Y. Haimes. *Modelin of large scale systems in a hierarchical-multiobjective framework.* North-Holland, 1961.

[20] H.Khalil. *Nonlinear System.* Prentice-Hall, 1995.

[21] H.S.Matthews and L.B.Lave. Price setting for green design., 1998.

[22] J.Berge, O.Levia, and J.Rouillard. *Models in System Design.* Kluwer Academic Publishers, 1997.

[23] J.E.Hopcroft and J.D.Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, 1979.

[24] J.E.M.Klostermann and A.Tukker. *Product Innovation and Eco-efficiency.* Kluwer Academic Publishers, 1998.

[25] W.D. Kelton, R.P.Sadowski, and D.A. Sadowski. *Simulation Language with Arena.* McGraw-Hill, 1998.

[26] B.E. Koenig. *Enterprise Models for the Design and Management of Manufacturing Systems.* PhD thesis, Michigan State University, 1992.

[27] L.Barfield. *The User Interface Concepts and Design.* Addison Wesley, 1993.

[28] L.Weinberg. The development of a streamlined, environmental life cycle analysis matrix for facilities. *IEEE Symposium on Electronics & Environment,* pages 65–70, 1998.

[29] N.M.Darnall, G.I.Nehman, J.W.Priest, and J.Sarkis. A review of environmentally conscious manufacturing theory and practices. *Int'l Journal of Environmentally Conscious Design & Manufacturing*, 3(2):49–57, 1994.

[30] Society of Environmental Toxicology and Chemistry. A technical framework for life-cycle assessments., January 1991. Workshop Report, Washington, D.C.: Society of Environmental Toxicology and Chemistry, and SETAC Foundation for Environmental Education, Inc.,.

[31] J.L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, 1981.

[32] J.R. Poyner and M. Simon. Integration of dfe tools with product development. Technical report, Dept. of Mechanical Eng., Manchester Metropolitan University, 1995. DFE/TR25.

[33] S.B.Billatos and N.A.Basaly. *Green Technology and Design for the Environment.* Taylors&Francis, 1997.

[34] R.A. Sheldon. Organic synthesis-past, present, and future. *Chemistry and Industry*, (23), 1992.

[35] A. Sweatman and M. Simon. Integrating dfe tools into the design process. Technical report, Dept. of Mechanical Eng., Manchester Metropolitan University, 1996. DFE/TR30.

[36] A. Sweatman and M. Simon. Dfe/tr34–products of sustainable future. Technical report, Dept. of Mechanical Eng., Manchester Metropolitan University, 1997. (URL:http://sun1.mpce.stu.mmu.ac.uk/pages/projects/dfe/dfe/.html).

[37] R.L. Tummala and L.J. Connor. Mass-energy based economic models. *IEEE Trans. Sysst., Man, Cybern.*, 3(6):548–555, 1973.

[38] R.L. Tummala and B.E. Koenig. Models for life cycle assessment of manufactured products. *IEEE Int'l Symposium on Electronics and the Environment*, pages 548–555, 1994.

[39] R.L. Tummala, B.E. Koenig, and H.E. Koenig. Process network theory and implementation for technology assessment in manufacturing. *Proceedings of a Joint German/US Conference, Hagen, Germany*, 1992.

[40] S.H. Weissman and J.C.Sekutowski. Environmentally conscious manufacturing: A technology for the nineties. Technical report, AT&T Technical Journal, 1997.