CEPHDA
Chemical Engineering Process Hierarchical Design with Ascend

Kay C Dee, Arthur ¥. Westerberg

EDRC 06-140-92

CEPHDA: CHEMICAL ENGINEERING PROCESS HIERARCHICAL DESIGN WITH "ASCEND"

K. C. DEEt and A. W. WESTERBERG

Department of Chemical Engineering and Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.
t Currently graduate student, Department of Biomedical Engineering, Rensselaer Polytechnic University, Troy, NY 12180, U.S.A.

EDRC Research Report Series #06-140-92

This work has been supported by the National Science Foundation.

CEPHDA: CHEMICAL ENGINEERING PROCESS HIERARCHICAL DESIGN WITH "ASCEND"

K. C. DEEt and A. W. WESTERBERG

Department of Chemical Engineering and Engineering Design Research Center,
Carnegie Mellon University, Pittsburgh, PA 15213, U.SA
t Currently graduate student, Department of Biomedical Engineering,
Rensselaer Polytechnic University, Troy, NY 12180, U.S*JL*

Abstract — CEPHDA is a computer-aided chemical engineering process design tool, which we implemented in the 'ASCEND* (Advanced System for Computations in ENgineering Design) modeling environment. CEPHDA contains models for levels 2 (treating a whole process as a "black box*) and 3 (implementing recycle flows) of the hierarchical design process which Douglas (1988) developed in his text 'Conceptual Design of Chemical Processes.' We have tested CEPHDA by modeling a three step process from toluene to benzene to ethyl benzene to styrene. The CEPHDA library was created so that inexperienced process engineers can readily create, evaluate, and optimize such models.

1. INTRODUCTION

The process of designing a system in any engineering discipline is inherently creative. Typically, engineering problems are under-defined, making it necessary for the engineers involved to supply missing information crucial to the problem in question. When faced with a process design problem, experience in designing similar processes is an important asset, providing knowledge of reliable design heuristics and useful "back of the envelope calculations'[1]. These methods are generally used to sort through process alternatives, helping engineers select optimal system components and conditions, resulting in a well-designed process. Often, when an engineer is in a learning position (either as a student in a process synthesis course or as a new process engineer in industry) constructive use of these design heuristics is hampered by inexperience and unfamiliarity with the engineering design process. A tool to help such 'engineers in training* become familiar with the design process, and to assist experienced engineers in quickly evaluating process alternatives, will be a valuable asset in a design environment.

James M. Douglas is credited with articulating one methodological approach to the design of chemical engineering systems, outlined in his textbook 'Conceptual Design of Chemical Engineering Processes' [1]. The "Douglas Method" is based on hierarchical decision-making, using economic feasibility as a main criterion for process evaluation. A complex problem is gradually solved through completion of a number of arbitrary "stages", or levels of analysis.

The ASCEND (Advanced System of Computations in ENgineering Design) modeling system, currently under development at the Engineering Design Research Center at Carnegie Mellon University, is an interactive model-building environment currently available on the Apollo computer for use in the X window environment. The ASCEND system is heavily structured, declarative, strongly typed and incorporates object-oriented programming [2, 3, 4].

CEPHDA, (Chemical Engineering Process Hierarchical Design with ASCEND) combines the arbitrarily structured design approach of the Douglas method and the inherently structured ASCEND computing environment to create a flexible modeling tool. The user proceeds at will through the

Douglas stages, choosing which design variables to specify and which variables ASCEND will compute or optimize.

CEPHDA has been successfully used to model a typical undergraduate chemical engineering process design problem: a three-step styrene synthesis from toluene to benzene to ethyl benzene to styrene. Compared with manual use of the Douglas method, the CEPHDA-generated design allowed for rapid detection and location of design errors, swift formulation of material balances and reactions, easy generation and evaluation of process alternatives, and quick formulation of primary levels of analysis.

2. THE DOUGLAS METHOD

James M. Douglas divides the conceptual design of chemical processes into five generic steps, or stages of thought. These vary from rapid, simple estimates to detailed calculations. Economic feasibility is used as a main criterion for process evaluation at every stage of the Douglas method. While initial order-of-magnitude process cost estimates have been quoted to possess a probable accuracy of ±40%, a simulation based on costing major items of equipment can allegedly result in a cost estimate accuracy of ±25%, and a preliminary budgeted estimate can allegedly again raise the predicted accuracy to ±12% [9]. It has been claimed that definitive estimate using high levels of analysis in the Douglas method and almost complete process data can result in an accuracy of ±6% (Douglas, 1988). Checking the projected economic potential at early stages of the design process, as is done in the Douglas method, allows for quick elimination of non-feasible design alternatives.

Following are brief discussions of individual levels contained within the Douglas method, detailing the types of decisions and design alternatives to be considered at each stage and some heuristics commonly used as design guidelines for each level of analysis.

2.1 LEVEL ONE

Level One assimilates very basic process information. The user need only know the basic chemistry for the main reactions that will occur in the process: reaction stoichiometry and molar flow rates of either feed components or exit components. The user may assume that the reactions will go to full completion or may either specify or optimize the extent of each reaction. The user may assume that there are no competing or side reactions. This level of analysis assumes that the user has input the correct stoichiometric coefficients and had accounted for all main reactions in the process. A mass balance is performed to determine input or output molar flows, and it is possible to perform a simple economic analysis of the process based solely on market worth of chemicals comprising input and output streams specified by the user.

It is generally assumed that the process in question is a continuous process. A batch process would be designed on consideration of three process characteristics: production factors, market forces, and scale-up factors. If the production factors were such that the production rate was less than 1x10* lbs/yr of product or the process to be designed was a multiproduct plant, the process should be designed as batch. If demand for the product was seasonal or if the product had a short lifetime, market forces would indicate a batch design. Finally, scale-up problems that would indicate a batch process include very long reaction times and handling of slurries at low flow rates or rapidly fouling materials.

2.2 LEVEL TWO

Level Two sets the basic input-output structure of the process flow sheet. The user must decide whether or not to purify feed streams to the process based on product purity requirements. A perfect separation system is assumed in this level of analysis. Thus, the user may recycle reactants or purge streams as desired. A process alternative that should be considered at this level involves reversible by-products: they may be recycled to the reactor and allowed to build to an equilibrium level, but the design engineer

should keep in mind that the equipment will have to be oversized to handle the increased stream flow. This presents additional costs in equipment and possibly utilities as a tradeoff for increased reaction selectivity.

2.3LEVELTHREE

Hard-core process flow sheet decisions are finalized in Level Three. The user may choose how many reactors will be needed, and whether or not some components should be separated between reactors. The user specifies how many recycle streams will be present, and whether or not gas compressors or pumps will be needed for those streams. At this level, reactor operating conditions are considered: is adiabatic operation possible? Will heating and cooling be needed? Reactor, compressor, and pump costing helps provide a more complete economic potential at this level, as does a comparison of reactant conversion versus reactor cost.

2.4LEVELFOUR

Level Four involves the design of the separation system. Depending on the phase states of the process streams involved, a vapor or liquid recovery system must be designed.

There are four basic choices for the location of a vapor recovery system: a purge stream, a gas-recycle stream, or a flash vapor stream. The most common vapor recovery systems include condensation, absorption, adsorption, membrane processes, and reaction systems.

When designing the liquid recovery system, it should be remembered that in general, distillation is the least expensive means of separating mixtures of liquids. The sequencing of columns is an intensive design problem. A number of heuristics can be used to help sequence a series of distillation columns: corrosive components should be removed quickly, as should reactive components or monomers. Products and recycle streams should be removed as distillates, not condensates. More plentiful components should be removed quickly, as should the lightest components. Equimolar splits should be favored, and difficult separations should be saved for last. While distillation is an attractive separation alternative, if the relative volatilities of two components with similar boiling points is less than roughly 1.1, distillation becomes very expensive: a large reflux ratio is required which means large condensers and reboilers, large column diameter, and expensive utilities. Alternatives to continuous distillation with reflux include extraction, extractive distillation, azeotropic distillation, reactive distillation, crystallization, adsorption, and reaction. A detailed economic analysis is possible at this level, taking into account capital equipment costs and projected maintenance and replacement costs.

2.5 LEVEL FIVE

A heat-exchanger network is designed in Level Five. It is necessary to calculate the temperature/enthalpy curves for each process stream to design an effective network: often, the use of software packages such as FLOWTRAN© is useful to help find the minimum heating and cooling requirements of a process, match streams on a cascade diagram, and determine the minimum number of heat exchangers needed.

As of August, 1992, CEPHDA can design up to and including Level Three in the Douglas method.

8. ASCEND

The ASCEND environment has been evolving since the early 1980*8 through the interest and support of a combination of academic and industrial factors, including the Engineering Design Research Center, Carnegie Mellon University, Exxon, DuPont, Kodak, and Mobay. Papers on ASCEND have been published in *Research in Engineering Design, Computers in Chemical Engineering,* the Engineering Design Research Center *Research Report Series,* the Carnegie Mellon University School of Urban and Public Affairs *Working Paper Series,* and presented at the Fourth

International Symposium on Process Systems Engineering and national American Institute of Chemical Engineers meetings. Modification, refinement, and utilization of the ASCEND environment is an ongoing process at the Engineering Design Research Center, involving an interdisciplinary group of faculty, staff, and students.

While many problems in engineering design can be characterized as systems of mathematical relations, this approach is complex; typical equational systems in engineering are large, tightly coupled, nonlinear, and difficult to visualize. The need for computer-aided design is obvious; however, "aided" is the key word in this phrase. The role of human interaction in the design process is one which currently cannot be replaced by automation for tasks such as problem formulation, verification, and debugging. The quality of human intervention with and influence on an equationally based system is directly proportional to the quality of the modeling tools in use. The ASCEND modeling environment allows the user to interact with and influence any level of model analysis, while providing mechanisms for problem examination and debugging. Though ASCEND was primarily motivated by the needs of chemical engineers, its use is not limited to the discipline of chemical engineering.

ASCEND was developed with the goal of producing an equation-oriented modeling environment for use in engineering design with the following attributes: rapid, modular problem formulation, the capability to solve large sets of nonlinear algebraic equations, hierarchical interaction with and modification of all levels of model formulation, and substantial user support for problem formulation, examination, and debugging.

The ASCEND modeling language achieves these goals through a combination of factors. The language is strongly typed, allowing variables to be defined as any meaningful concept. For example, a variable may be defined as a flow rate, a temperature, or a chemical compound. ASCEND requires that all equations within models are dimensionally consistent, significantly reducing debugging time and increasing a user's understanding of and confidence in models. Two other important features of the ASCEND language are incorporated from object-oriented programming: refinement hierarchies and partial/complete merging of models, illustrated below in Figure 1.

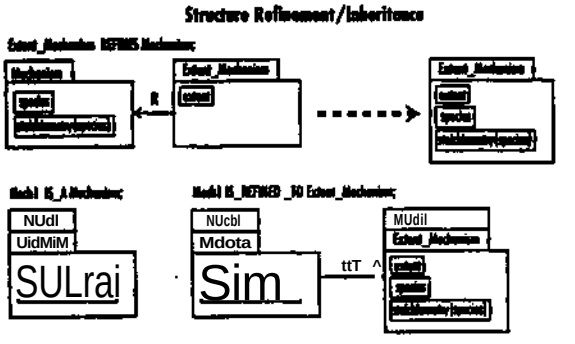

Figure 1.
Refinement ftndlAfcrcfingofIModcls

The user/machine interface consists of a number of windows which correspond to types of model manipulations, including storing libraries of code and compiled simulations, browsing through a simulation, or solving a system. ASCEND provides significant support for debugging. Typical error messages are very specific, including where error is and where and why it is occurring. At the request of the user, ASCEND will search for possible errors in a model which have not yet manifested themselves. ASCEND also displays a wealth of information about the system, including the number and types of equations and variables in models and sub-models, and the solving status of the system.

4. CONCEPTUAL DESIGN OF CEPHDA

4.1 FLEXIBILITY OF ENVIRONMENT

Both ASCEND and the Douglas method of chemical engineering process design are highly structured, yet combine to form a flexible, easily interactive design environment.

The ASCEND environment allows a user to declare groups of equations and variables called models, and then manipulate these models using a set of language-defined operators. Any level of data may be accessed through the ASCEND interface. It is possible to quickly "refine" an existing model using ASCEND operators. Once initial models have been created and linked to form a generic flow sheet, any user may change which design variables have fixed values specified by the user, and which are computed by ASCEND. ASCEND can solve systems of equations with a Newton-step solver, the MINOS package [5], and a successive quadratic programming package under development by Dr. L. T. Biegler, Carnegie Mellon University.

It is commonly noted in the literature on ASCEND that this flexibility and structural integrity is optimal for modeling units in an engineering process: for example, a distillation column. All equations pertinent to a tray of the column may be contained in one block of code: a model of a tray. Another block of code may be used to link as many "trays" as necessary in series. Models can also be constructed of a reboiler and a condenser for the column, and an all-encompassing model of the column itself can be created to contain the trays, reboiler, and condenser.

What is not noted as yet is that this structural integrity also lends itself well to modeling a design process. While there are many different specific articulations of an "interdisciplinary problem-solving design process," there are general underlying principles which are addressed in many different articulations [6,7,8], many of which can be exploited in the ASCEND environment.

4.2 CONCEPTUAL DESIGN STAGES

The first stage of a conceptual design process is generally termed the "Recognition" stage, where an engineer recognizes a problem and decides to tackle it. This stage would be used to decide what type of chemical process needs to be designed: the example used in this paper is that of a styrene production plant. The second stage is often termed "Definition". In order to rationally refine a broad problem statement, one must understand the physical parameters of the problem: the 'boundary conditions', as it were. Any limitations and restrictions on problem solutions should be determined, and the overall problem decomposed into sub-problems if need be.

The third stage, "Preparation", is one of research. Prior and current art should be compiled here, as well as data from the problem and any assumptions that will be made to facilitate a solution. When all applicable background information and data are collected and analyzed, stage four, "Synthesis", can occur. Here, solutions are developed from the information compiled earlier. "Evaluation", stage five, follows synthesis. Any solutions that have been generated should be verified and evaluated for presentation as an effective solution.

As noted previously, the blocking nature of ASCEND lends itself well to these stages. It is possible to construct certain blocks of code meant solely for problem definition, while others may be designated for preparation, synthesis, and evaluation.

5. CEPHDA STRUCTURE AND CODE

The overall goal of CEPHDA is to simplify the design process for engineers in a learning position. Since the ASCEND environment requires a user to interact directly with source code to simulate a process, the code structures need to be designed in such a way that a user with minimal ASCEND programming experience can discern where and how to change any code necessary to modify a wide variety of processes. CEPHDA code was designed to coincide with stages of the design process in order to assist design engineers who will need to maneuver easily through the source code.

Each of the current CEPHDA libraries are discussed below. It should be noted here that CEPHDA was created in the summer of 1992, at which time the only

sophisticated computer programming structures available in ASCEND was the "for loop" and certain operations permissible on sets of data. Throughout this documentation of the libraries, any required set operations within the code are outlined in some detail to illustrate the inner workings of CEPHDA for those who may be using the environment. CEPHDA is available for inspection and use on the Apollo network at the Engineering Design Research Center in Pittsburgh, PA

5.1KECOGN1TION.UB

Because the first stage of the engineering design process is often the recognition that a problem exists and that one has the tools to address it, the recognition library loads the source code necessary for ASCEND to recognize model components.

Two libraries of code loaded into ASCEND through the CEPHDA recognition.lib are 'cephdaatoms.lib' and 'cephdacomps.lib'. 'cephdaatoms.lib' reads in the default library of atoms, while adding two new definitions: cashflow, having dimensions of currency/time, and species_value, having dimensions of currency/mole. These atoms were added to facilitate economic evaluations of processes under consideration, following the Douglas method of process evaluation. The terminology "atoms" is used here in the ASCEND sense, meaning a generic concept which is used often and can have a defined type, dimensionality and default value. To illustrate, the code for the atom "cashjflow" is presented here:

```
ATOM cashjlow REFINES generic_real
        DIMENSION    currency/t
        DEFAULT      0.00 {USdollar/s};
END cashjlow;
```

CEPHDA also loads in 'cephdacomps.lib': a structurally altered version of the standard ASCEND components library. The ASCEND "component_constants" model has been expanded to include a listing of elements which make up each compound and a tally of how many atoms of each element are present in each compound. While it is possible to load in only selected components to an ASCEND model, all possible components are loaded for use with CEPHDA to provide flexibility for users and allow modeling of as wide a variety of processes as possible. The standard components library has been expanded as a result of CEPHDA's development, and now includes the following molecules.

Note that within CEPHDA, it is very important to follow the naming system presented here. For example, while it may seem structurally intuitive to name diphenyl as ^{M}(C6H5)2" instead of as "C12Hl(r, "(C6H5)y will not be recognized as a compound within ASCEND.

COMPONENT	NAMING
Hydrogen	H2
Carbon Dioxide	CO2
Water	H2O
Chloroform	CHC13
Methane	CH4
Methanol	CH4O
Ethane	C2H6
Ethanol	C2H6O
Ethylene	C2H4
Propylene	C3H6
Acetone	C3H6O
Propane	C3H8
n-Propanol	nC3H8O
i-Propanol	iC3H8O
n-Butane	nC4H10
i-Butane	1C4H10
n-Butanol	nC4H10O
i-Butanol	iC4H10O
n-Pentane	nC5H12
i-Pentane	iC5H12
Benzene	C6H6
n-Hexane	nC6H14
Toluene	C7H8
n-Heptane	nC7H16
Styrene	C6H5C2H3
Ethyl benzene	C6H5C2H5
Diethyl benzene	C8H9C2H5
Diphenyl	C12H10

As a further illustration, the code for the component "Hydrogen" is presented below. Note that the model "Hydrogen" is a 'universal model': in other words, it can be used at any level of any simulation once it has been loaded into memory. Note also that "Hydrogen" is a refinement of the model "componentjconstants". Physical data for all components used in CEPHDA

was taken from Reed, Prausnitz and Sherwood (1991).

```
MODEL componentjconstants;
 name              IS_A  set OF string;
 groups            KLA  set OF string;
 subgroups         IS_A  set OF string;
 nufsubgroups]     IS_A constant;
 elements          IS^A  set OF string;
 number[elements] IS_A constant;
 mw, Tb, Tc, Pc, Vc, Zc, omega, cpvapa, cpvapb,
 cpvapc, cpvapd, Hf, Gf, vpa, vpb, vpc_t vpd,Hv,
 Tliq, Vliq, TO, PO, HO, SO     ISJV  constant;
 INITIALIZE
   PROCEDURE reference;
        TO := 298.15 {K};
        PO := 1.0 {atm};
        HO := Hf;
        SO:=(Hf.Qf)/TO;
   END reference;
END componentjconstants;


UNIVERSAL MODEL Hydrogen REFINES
        componentjconstants;
 name := [H2']ƒ
 groups := Q;
 subgroups := Q;
 elements := fH[1]];
 INITIALIZE
   PROCEDURE values;
        mw := 2.016 {g/gm_mol};
        Tb := 20.37 {K};
        Tc:=33.3{K};
        Pc := 12.8 {atm};
        Vc := 66.0 {cm^A3/gm_mol};
        Zc := 0.305;
        omega := -0.22;
        cpvapa := 6.483 {cal/gm.mol/K};
        cpvapb := 2.215e-3 {cal/gm_mol/K};
        cpvapc := -3.298e-6 {cal/gm.mol/K};
        cpvapd := 1.826e-9 {cal/gm_mol/K};
        Hf:=0.00{J/gm_mol};
        Gf := 0.00 (J/gmjnolJ;
        vpa := -5.57929;
        vpb := 2.60012;
        vpc := -0.85506;
        vpd:=  1.70503;
        Hv := 904 {J/gm_mol};
        Tliq := 20.0 {K};
        Vliq := 28.3944 {cm^A3/gm_mol};
        numberfH[1]] := 2;
        RUN reference;
   END values;
END Hydrogen;
```

5.2DEFIN1TION.UB

This library consists of building blocks for a chemical process: blocks of code which define concepts CEPHDA needs in order to understand a process design problem. In other types of process flow simulations, the user would connect these building blocks together in a systematic way, imitating the physical flow of chemicals through a plant. While ASCEND allows a user access to all levels of code libraries, and thus the option of designing processes in that linear manner, it is anticipated that at lower levels of process design, users may not perform such operations. To accommodate use by these engineers, there are models stored in this library which correspond to the pre-designed Douglas levels of analysis, in addition to typical process flow blocks.

Following is documentation of the models within the Definition library. The order of documentation here follows the order of the models within the library. This documentation is primarily intended for those who are implementing CEPHDA in their own research, and have access to the complete code. Others may wish to proceed to Section 6, Styrene Synthesis Problem.

STREAM A stream is defined here as consisting of a set of species, the mole fractions of each species, the total molar flow rate, and the parameter 'datafspecies][1], which is a device to relate the species names to the corresponding component properties in the cephdacomps.lib. The equations describing a stream are that the total molar flow rate equals the sum of the individual molar flow rates, and that the individual molar flow rates equal the individual mole fraction times the total molar flow rate.

To illustrate, the CEPHDA code for the model 'stream* is presented here.

```
MODEL stream;
  species          IS_A  set OF string;
  x[species]       IS_A  fraction;
  F, M[species]    IS_A  molar_rate;
  datafspecies]  IS^A  component .constants;
          F = sum(m[species]);
          FOR i IN species CREATE
                  m[i] = x[i]*F;
          END;
END stream;
```

STREAMS Throughout CEPHDA, once a model is defined, a plural model is usually defined as well. Thus, 'streams[1] allows an array of type 'stream[1] to be formed.

To illustrate, the code for 'streams' is presented here.

```
MODEL streams;
  njstreams        IS^A  integer;
  s[l..n_streams] IS_A  stream;
END streams;
```

REACTION A generic reaction contains the set of chemical species which are used within that reaction and the stoichiometric coefficient of each of those species. There is a set of string termed 'flag' which is automatically assigned a value elsewhere when a reaction is refined. The variable 'flag' is used to determine which type of reactor or reactor mechanism is needed for that reaction. It may have values indicating a reaction governed by selectivity, extent, or conversion.

REACTIONS Reactions creates an array of type 'reaction[1].

EXTJIEACTION (refines reaction) The naming convention for reaction and reactor types is as follows: the prefix 'ext' denotes an extent-governed mechanism, the prefix 'sel* denotes a selectivity-governed mechanism, and the prefix 'con' denotes a conversion-governed mechanism. This extent-governed reaction contains the parameter 'extent', which is the extent of reaction. The variable "flag[11], contained in the original data type 'reaction', is set to •extent' by CEPHDA

EXTJREACTIONS Creates an array of type extjreaction.

SEL.REACTION (refines reaction) The parameter 'selectivity* is defined here as the ratio of a desired product formed to the ratio of an undesired product formed. The species which are desired and undesired are specified by the user, as is the selectivity value. The variable "flag", contained in the original data type 'reaction', is set to 'select' by CEPHDA.

SELJEtEACTIONS Creates an array of type seLreaction.

CON_REACTION (refines reaction) The user must specify the reactants in the reaction as a set of species as well as which of those species will be the limiting reagent. The conversion of the limiting reagent is also specified. The variable "flag", contained in the original data type 'reaction', is set to 'conversion' by CEPHDA.

CON.REACTIONS Creates an array of type con_reaction.

UPDATE.DATA This is more of a compiler instruction than an actual model. Basically, when a stream is passed into 'update_data', the individual components within that stream are refined from the generic type 'component.constants' to the component corresponding to the species name. This is currently done through one of a number of possible set interactions due to the lack of an "if* condition in ASCEND. No plural for this model currently exists.

ADD.STREAM This is a mixer, with the number of inputs specified by the user. An array of feed streams and an output stream is created. An array called 'streamjfor* is created to contain integers. The species in the output stream are declared to be the union of the species within all the input streams. The array 'streamjbr* for any component contains the numbers of all streams which contain that component, and the output molar flow rate of each component is then set to the sum of the molar flow rates of that component in each feed stream, where the number of each feed

stream needs to be contained in the array 'streamjbr[1] for that component.

ADDJ9TREAMS Creates an array of type addjstream.

REACTOR A generic reactor contains an input and an output stream and a number of reactions.

REACTORS Creates an array of type reactor.

EXT_REACTOR (refines reactor) An extent reactor contains an array of extent reactions and array of integers termed 'needed* for each species in the output stream. The species in the output stream are set to the union of all species involved in all reactions and the species in the input stream. The array 'needed[1] is filled in the same way that the array 'streamjbr* is filled in the model [add_stream\ where the integers correspond to the reaction array number where that species is needed. For all species in the output stream, the output molar flow rate minus the input molar flow rate (if such a flow rate exists: a set manipulation involving sums is used here. If there is no input molar flow rate, the set is empty and the sum skips that parameter) is set equal to the sum of the stoichiometric coefficient of each component multiplied by the extent of reaction, provided that the reaction is in the array 'needed' for that component.

Since the other reactor types are similar in structure to the extent reactor, code for the extent reactor is presented here for illustrative purposes.

```
MODEL extjreactor REFINES reactor;
  rxn[l..n__rxns]              IS_A  ext_reaction;
  needed[output.species]  IS_A  set OF integer;
  update                       IS_A  update_data;
output.species:=UNIO N(rxn [1. .n_rxns}.involved,
          input.species);
  FORj IN output.species CREATE
          needed[j] := [k IN [l.n.rxns) I j IN
          rxn(k].involved];
  END;
  FOR a IN output.species CREATE
          (output.m[a] - SUM(input.m[j] I j IN
          INTERSECTION (a,input.species))) =
          SUM(rxn[b].stoich[a]*rxn[b].extent | b
          INneeded[a]);
  END;
output, update.strm ARE.THEJSAME;
END extjreactor;
```

EXT_REACTORS Creates an array of type extjreactor.

SEL_REACTOR (refines reactor) A selectivity-governed reactor. Creates an array of selectivity reactions and sets which function purely as storage space, used in finding die differences between sets of reacting species. There is also an array of streams called "temp", used as intermediates between reaction blocks: reactions are run sequentially through this reactor in an order determined by the user. The stream 'temp[l]' is set to be equivalent to the input stream. Throughout the reaction mechanism, streams termed temp[(i+l)] may be thought of as an output stream for each separate reaction and the streams temp[i] may be thought of as input streams for each separate reaction. The basic equations operating here are: 1) The moles of desired product formed divided by the moles of undesired product formed equals the selectivity, and 2) The ratio of the stoichiometric coefficient of the desired product over that of another species involved in a reaction is equal to the ratio of the moles of the desired product formed over the moles of the other species which is either formed or consumed.

SEL__REACTORS Creates an array of type seLreactor.

CON_REACTOR (refines reactor) This is a conversion-governed reactor. An array of conversion reactions is formed. Sets are created for storage when finding the differences between sets of reacting species. Streams termed "temp[1]* are used as the intermediates between reaction blocks: reactions are run sequentially through this reactor in an order determined by the user. The stream 'temp[l][v] is set to be equivalent to the input stream. Throughout the reaction mechanism, the streams termed temp[(i+1)] may be thought of as output

streams for each separate reaction and the streams temp[i] may be thought of as input streams for each separate reaction. The basic equations operating here are: 1) The molar flow rate of the limiting reagent times the conversion of that reagent is equal to the molar flow of that reagent which is consumed in the reaction and 2) The ratio of the stoichiometric coefficient of the desired product over that of another species involved in a reaction is equal to the ratio of the moles of the desired product formed over the moles of the other species which is either formed or consumed

CON_REACTORS Creates an array of type con_jreactor.

PFR The term 'plug flow reactor' is used to denote a multi-function reactor here. A pfr contains an input and an output stream, an array of reactions of a size determined by the user, and an array of reactors of unspecified type (simply the generic reactors) which each correspond to a different reaction. There are again streams termed "temp" which are used as the intermediates between reaction blocks: reactions are also run sequentially through this reactor in an order determined by the user. The stream 'temp[l]^{f} is set to be equivalent to the input stream. Throughout the reaction mechanism, the streams termed temp[(i+l)] may be thought of as output streams for each separate reaction and the streams temp[i] may be thought of as an input stream for each separate reaction. For each reaction 'n' fed into the pfr, the reaction variable 'flag' is checked. The reaction 'n^{f} is then refined to the appropriate type of reaction (extent, selectivity, or conversion), according to the value of the variable "flag", the reactor 'n* is refined to the appropriate type of reactor (extent, selectivity, or conversion), the temp[i] stream is defined as the input stream for reactor 'n' and the stream temp[(i+l)] is defined as the output stream for the reactor 'n\ Only one type of reaction and reactor may be utilized for each reaction.

PFRS Creates an array of type pfr.

RECYCLE The recycle block has an input, an output, and a recycle stream. 'Recycle* calls in the model 'sep.stream' to create a perfect separation. The user specifies which species are to be recycled. The output stream can continue through a flow sheet.

RECYCLES Creates an array of type recycle.

PURGE The purge block consists of three streams: an input, an output, and a 'release' stream, where the 'release' stream is the purge stream. There also exists a parameter 'tract[1], which is the fraction of the input stream which will go to the purge stream.

PURGES Creates an array of type purge.

BASIC JJNIT This is the Douglas Level One pre-defined process unit. It consists solely of an addstream and a pfr in series. The number of addstreams and pfrs are specified here as a CEPHDA default, as is the input/output structure.

EECYC_UNIT This recycle unit consists of a basic unit with a recycle block added after the reactor. The recycle is added in to the addstream.

RECPURGJJNIT This is a Douglas Level Two pre-defined process unit. It consists of an addstream, a pfr, a recycle block and a purge block. The recycle stream occurs after the reactor. A fraction of the recycle stream is purged.

5.3 EVALUATION.LIB

This library consists of models intended to help a user evaluate system performance. Currently, it contains two evaluation models.

The economics model is filled with market prices for chemicals from August 1992. The 'evaluation' model has a number of cost streams and a number of product streams defined by the user. The user may declare any process stream to be either a cost or a product stream. The economic potential model is in terms of money coming in from

the process and total amount of money being spent for the process. The economic potential evaluated at Level One of the Douglas Method is the economic potential of the process in terms of stream values.

The atom balancer is a check on stoichiometry. For every kind of element [i] in every species fi] involved in a reaction, it sums the number of atoms of [i] in each species involved in the reaction multiplied by the stoichiometric coefficient of the species containing that type of atom. If the stoichiometry for the reactions has been input incorrectly by the user, an error message appears.

5.4 PREPARATION.LIB

The user need only interact with the code presented in the preparation libraries. All of the code in the other libraries can be arranged in any configuration to simulate any process through modification of the models in preparation.lib. All models requiring user interaction are named with the prefix "prep*[1], and are outlined below.

PREP.STREAMS The user must specify the number of process streams they wish to designate information about. The species contained in each stream must be designated here also. Each stream is initialized in a separate procedure, which gives the user the option to create a number of streams and include or exclude them at will.

PREP_RXNS Prep_rxns prepares the reaction information. The user declares each reaction as a specific type: con.reaction, seljreaction, or ext.reaction for reactions governed by conversion, selectivity, or extent respectively. Each of the three types of reactions must have all involved species designated here. All conversion reactions must have the reactants and limiting reagent explicitly specified as well. All selectivity reactions must have the desired and undesired components specified.

PREPJEVALUATION The number of streams which are to be evaluated economically as process costs (containing chemical species which must be purchased) and product streams (containing chemical species which are to be sold) are specified here.

PREPJJNIT Prepjunit is intended to prepare a reactor block: to assign the corresponding reactions to the reactor in the sequential order they are to be carried out The number of reactions within the reactor must also be designated. If a recycle or purge stream is to be used, the chemical species to be recycled or purged must be specified here as welL

BLACKJ3OX The "black box" here, in keeping with Douglas's terminology, contains all the inner workings of the process. The number of input and output streams to the entire process must be specified by the user. Any streams which had been prepared in "prepjstreams" and are to be used in the simulation must be assigned to the corresponding system parts. The streams which will be included in the economic evaluation should also be designated here.

6. STYRENE SYNTHESIS PROBLEM

The problem used to test CEPHDA's performance was that of styrene synthesis from the dehydrogenation of ethyl benzene. This process is considered here as a three part system followed by a styrene purification sequence. First, toluene is hydrodealkylated into benzene, separated, and sent to the second stage. The benzene is then catalytically converted to ethyl benzene. The ethyl benzene is converted in a catalytic reactor into styrene. An inhibitor (for example, dinitrophenol inhibitor or DNP) may be added to increase product purity.

Following is the chemical reaction stoichiometry. Brackets indicate reactions grouped into one of the stages specified above.

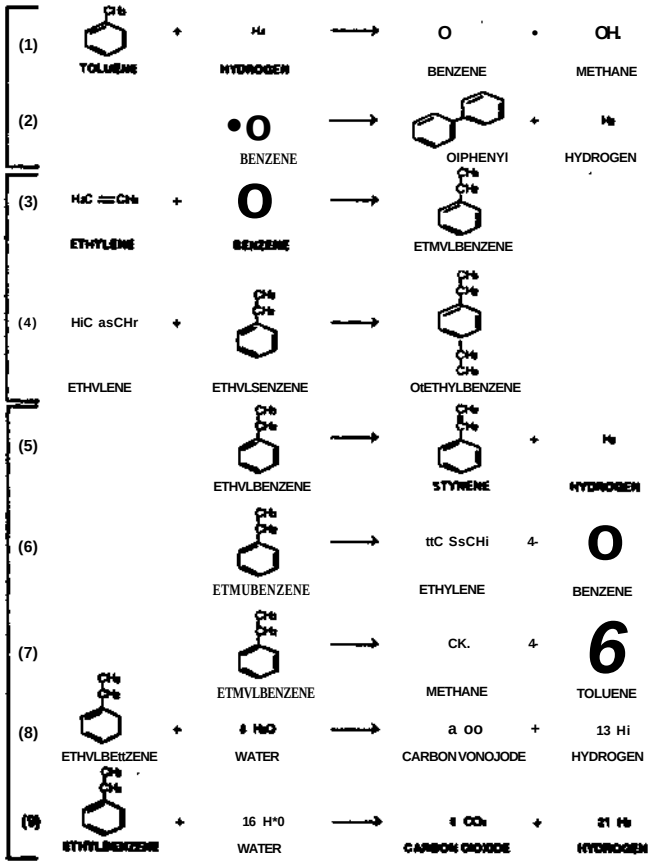

Figure 2.
Styrene Synthesis Reactions

Specific product and feed specifications were set as follows. Three feed streams were assumed to be readily available from existing processes: an ethylene feed stream of 98 mol% ethylene and 2 mol% methane at 100 psia and ambient temperature, a toluene feed stream at 99 mol% toluene and 1 mol% benzene at 1 atm and ambient temperature, and a hydrogen feed stream at 96 mol% hydrogen and 4 mol% methane at 300 psia and ambient temperature. Utilities are assumed to be available at standard market prices.

This problem is commonly used as a design task for engineering students. Plant designs composed for course 06-302, Process Engineering and Synthesis, (Fall 1991, Department of Chemical Engineering, Carnegie Mellon University) using the above design parameters and the Douglas method manually [10] were compared with CEPHDA-generated plant designs.

6.1 LEVEL ONE

The same assumptions were made using CEPHDA and in the exercise of the Douglas method manually: namely, that the main reactions would dominate in the reactor (thus, side and competing reactions could be ignored), that the process was continuous, and that the reactions would go to near completion. Only reactions # 1, 3, and 5 from Figure 2 were used in this Level One analysis.

6.2 LEVEL TWO

Assumptions made for Level Two included that the reactor was operating adiabatically and that a perfect separation system is implemented. All nine reactions from Figure 2 were used as synthesis reactions. Hydrogen was recycled, and a thirty percent purge was specified.

7. CONCLUSIONS

At the Level One stage of analysis, given the product specifications and the species involved in the reactions, CEPHDA successfully solved for other process parameters, creating a process with a positive economic potential. The time required to formulate this simulation was orders of magnitude less than that required to formulate such a simulation manually. The plant designs produced for course 06-302 by students had errors in material balances and reaction formulation while the CEPDHA plant design did not.

While CEPHDA again successfully produced an economically feasible plant design at Level Two, the design was operating under non-optimum conditions, resulting in a low yield of styrene product. Interestingly, the plant design created by CEPHDA at this level is remarkably similar to a plant which was designed for 06-302, implying that the same conceptual errors had been made in both plants. However, it took three student chemical engineers two months of work to reach the design level that was reached in less than half an hour by one student chemical engineer using CEPHDA. Using CEPHDA, it was possible

to produce alternate designs in a time period of less than half an hour, instead of in a time period of days.

Compared with manual use of the Douglas method, the CEPHDA-generated designs allowed for rapid detection and location of design errors, swift formulation of material balances and reactions, easy generation and evaluation of process alternatives, and quick formulation of primary levels of analysis.

8. CONTINUING DEVELOPMENT

CEPHDA should be expanded to include the higher levels of the Douglas method. The components library may be expanded as use of CEPHDA warrants, and the existing code may be refined as the ASCEND programming language becomes more sophisticated. An extension of the Douglas design method may be added: notably, the addition of ASCEND/CEPHDA models which would allow for the hierarchical design of a process control system [11]. A hierarchical design of a process control system would be advantageous in that the convoluted system of control loops on a complete flow sheet can be reduced to a foundation of those which are strategically important and then built upward from there. Designing a process and the corresponding control system at the same time would allow for unusually complimentary designs, with the control system design exerting some influence on the process design. [11] As of December, 1992, refinement of CEPHDA code continues as the ASCEND environment becomes more sophisticated.

9. ACKNOWLEDGEMENTS

This work has been supported by the Engineering Design Research Center, a National Science Foundation Engineering Research Center, as a part of the REU (Research Experience for Undergraduates) program.

The author would like to acknowledge the invaluable assistance and patience of her advisor, Dr. Arthur Westerberg, as well as that of Peter Piela, Joe Zaher, and Bob Huss.

10. REFERENCES

1) Douglas, J. M., **Conceptual Design of Chemical Engineering Processes,** McGraw-Hill Chemical Engineering Series, McGraw-Hill Book Company, New York (1988).

2) Piela, P., McKelvey, R., **and** Westerberg, A., *An Introduction to ASCEND: Its Language and Interactive Environment,* Engineering Design Research Center, Carnegie Mellon University (1991).

3) Piela, P.C., Epperly, T. G., Westerberg, K. M., and Westerberg, A. W., "ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis: The Modeling Language," *Computers Chem. Engng,* Vol. 15, pp. 53-72 (1991).

4) Piela, P., Katzenberg, B., and McKelvey, R., *Integrating the User into Research on Engineering Design Systems,* EDRC Report, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, 15213 (1991).

5) Murtagh, B. A., and Saunders, M. A., *MINOS User's Guide,* Technical Report SOL 83-20, Systems Optimization Laboratory, Dept. of Operations Research. Stanford University (1985).

6) Sturges, R., Professor of Mechanical Engineering, Mechanical Engineering Department, Carnegie Mellon University, "The Creative Design Process", seminar, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, 15213 (1992).

7) Future Problem Solving Association, the University of Michigan, Ann Arbor, MI (1985).

8) Dee, K. C, "Engineering Design of an Angioplasty Catheter," Carnegie Mellon University, Department of Biomedical Engineering, Pittsburgh, PA, 15213 (1992).

9) Pikulik, A., and Diaz, H. E., "Cost Estimating Major Process Equipment," *Chem. Eng.*, 84 (21): 106 (1977).

10) Bauer, M., Dee, K.C., and Lehrer, R., "Design of a Styrene Production Plant", Process Engineering and Synthesis, (course #06-302), Carnegie Mellon University, Department of Chemical Engineering, Pittsburgh, PA 15213 (1991).

11) Ponton, J., Professor of Chemical Engineering, Chemical Engineering Department, the University of Edinburgh, "The Creative Design Process", seminar, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, 15213 (1992).