

**A Framework for the Language and Logic of Computer-Aided
Phenomena-Based Process Modeling**

by

Jerry Bieszczad

B.S. Chemical Engineering
University of Connecticut, Storrs (1994)

Submitted to the
Department of Chemical Engineering
in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY IN CHEMICAL ENGINEERING

at the
Massachusetts Institute of Technology

February 2000

© 2000 Massachusetts Institute of Technology
All rights reserved.

Signature of Author: _____
Department of Chemical Engineering
October 18, 1999

Certified by: _____
George Stephanopoulos
Arthur D. Little Professor of Chemical Engineering
Thesis Supervisor

Accepted by: _____
Robert E. Cohen
St. Laurent Professor of Chemical Engineering
Chairman, Committee for Graduate Students

A Framework for the Language and Logic of Computer-Aided Phenomena-Based Process Modeling

by

Jerry Bieszczad

Submitted to the Department of Chemical Engineering
on October 18, 1999 in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Chemical Engineering

ABSTRACT

Chemical process engineering activities such as design, optimization, analysis, control, scheduling, diagnosis, and training all rely on mathematical models for solution of some engineering problem. Likewise, most of the undergraduate chemical engineering curricula are model-based. However, the lack of formalization and systematization associated with model development leads most students and engineers to view modeling as an art, not as a science. Consequently, model development in practice is usually left to specialized modeling experts.

This work seeks to address this issue through development of a framework that raises the level of model development from procedural computations and mathematical equations to the fundamental concepts of chemical engineering science. This framework, suitable for implementation in a computer-aided environment, encompasses a phenomena-based modeling language and logical operators. The modeling language, which represents chemical processes in terms of interacting physicochemical phenomena, provides a high-level vocabulary for describing the topological and hierarchical structure of lumped or spatially distributed systems, mechanistic characterization of relevant phenomena (e.g., reactions, equilibria, heat and mass transport), and thermodynamic and physical characterization of process materials. The logical operators systematize the modeling process by explicitly capturing procedural and declarative aspects of the modeling activity. This enables a computer to provide assistance for analyzing and constructing phenomena-based models, detect model inconsistencies and incompleteness, and automatically derive and explain the resulting model equations from chemical engineering first principles.

In order to provide an experimental apparatus suitable for evaluating this framework, the phenomena-based language and logical operators have been implemented in a computer-aided modeling environment, named *MODEL.LA*. *MODEL.LA* enables phenomena-based modeling of dynamic systems of arbitrary structure and spatial distribution, hierarchical levels of detail, and multicontext depictions. Additional components allow incorporation of thermodynamic and physical property data, integration of control structures, operational task scheduling, and external models, and assistance for specification and solution of the resulting mathematical model. Application of this environment to several modeling examples, as well as its classroom and industrial deployment, demonstrate the potential benefits of rapid, reliable, and documented chemical process modeling that may be realized from this high-level phenomena-based approach.

Thesis Supervisor: George Stephanopoulos

Title: Arthur D. Little Professor of Chemical Engineering

Acknowledgements

I would like to thank my advisor, Professor George Stephanopoulos, for the inspired mentorship and scholarly example he has given me. His encouragement to critically evaluate and consider the ideas of my work beyond the scope of this thesis has been a constant source of motivation over the past several years. Professor Stephanopoulos has provided indispensable guidance regarding both my research and professional development, while allowing me the requisite freedom to pursue ideas and approaches of greatest interest to me.

I am also deeply indebted to the members of my thesis committee, Prof. Paul Barton, Prof. Jack Howard, Dr. Michael Mohr, and Prof. Gregory Rutledge. Through various discussions they have provided insightful direction and brought a broad perspective to the evaluation of my work. In addition, Professor Alan Foss at the University of California at Berkeley has provided sage advice regarding the role of *MODEL.LA* in an educational setting.

A project with the scope of *MODEL.LA* could not be accomplished alone, and I gratefully acknowledge the invaluable contributions of my colleagues. Dr. Alexandros Koulouris designed and implemented the *Properties Manager*, the incorporation of control structures, and the integration of external *gPROMS* models into *MODEL.LA*. Dr. Kevin Geurts, at the University of California at Berkeley, implemented the foundational design of the *Numerical Engine*. Finally, Dr. Manuel Rodriguez incorporated the declaration of operational schedules into *MODEL.LA*. With these extraordinary gentlemen, I have experienced many triumphs, shared a few defeats, and developed close friendships.

I give kind regards to members of the LISPE research group, both past and present, who have provided me with an endless source of ideas, advice, frivolous diversions, and many, many laughs. Particularly, the companionship of Matthew Dyer and Orhan Karsligi on our quest toward a Ph.D. has been most appreciated.

Finally, I must thank my family for the support that has truly made this thesis possible. *Dziękuję wszystkim!* My mom has sacrificed a great deal to give her children the best. I hope my small accomplishment gives her a source of pride. Most of all, I thank my wife, Chris, whose love, support, encouragement, and friendship make it all worth it.

Table of Contents

Acknowledgements	5
Table of Contents	7
List of Figures	13
List of Tables	17
Chapter 1 Introduction	19
1.1 Chemical Process Modeling Needs in Engineering Practice.....	20
1.2 Chemical Process Modeling Needs in Undergraduate Education.....	21
1.3 Potential Role of the Computer in Process Modeling	24
1.4 Existing Computer-Aided Process Modeling Tools.....	25
1.4.1 Sequential Modular Flowsheet Simulators	25
1.4.2 Programming Languages	26
1.4.3 Spreadsheets	27
1.4.4 Equation-Based Process Modeling Tools.....	28
1.4.5 Summary of Existing Computer-Aided Process Modeling Tools	31
1.5 Physicochemical Phenomena-Based Process Modeling	32
1.5.1 Proposed Phenomena-Based Modeling Approaches	33
1.5.2 Summary of Proposed Phenomena-Based Modeling Approaches	35
1.6 Research Objectives	36
1.6.1 Development of a Formalized Phenomena-Based Modeling Language	36
1.6.2 Systematization of Modeling Activity through Modeling Logic	37
1.6.3 Implementation and Evaluation of Modeling Language and Logic through a Computer-Aided Modeling Environment	37
1.7 Thesis Outline	38

Chapter 2 Requirements for High-Level Process Modeling.....	39
2.1 Requirements for the Representation of Process Models.....	39
2.1.1 Declarative Model Representation	40
2.1.2 Chemical Engineering Science Basis of Models.....	41
2.1.3 Explicit Documentation of Assumptions	41
2.1.4 Hierarchical Nature of Models.....	42
2.1.5 Contextual Nature of Models.....	42
2.2 Requirements for Systematization of the Process Modeling Activity	43
2.2.1 Procedural Nature of Modeling Activity	43
2.2.2 Contextual Nature of Modeling Activity	45
2.2.3 Science and Art of Modeling	46
2.2.4 Documentation of Modeling Activity.....	47
2.3 Implementation of High-Level Computer-Aided Modeling Support	48
2.3.1 Phenomena-Based Modeling Language.....	48
2.3.2 Modeling Logic	49
2.3.3 Computer-Aided Modeling Environment	50
Chapter 3 Modeling Language Framework	53
3.1 Formal Modeling Language Representation.....	54
3.2 Hierarchy of Model Equations.....	56
3.3 Phenomena-Based Model Characterization.....	58
3.3.1 Structural Characterization	59
3.3.2 Chemical Characterization	60
3.3.3 Derivation Context	61
3.4 Characterization of Modeling Elements	61
3.4.1 Modeled-Unit Characterization.....	62
3.4.2 Flux Characterization	69
3.4.3 Material-Content Characterization.....	72
3.4.4 Phase Characterization	76
3.4.5 Chemical Species Characterization.....	79
3.4.6 Chemical Reaction Characterization.....	79
3.5 Semantic Relationships.....	82
3.6 Model Digraph.....	83
3.7 Model Derivation Tree.....	84
3.8 Complete Context-Free Grammar Description.....	84

Chapter 4 Modeling Logic Framework.....	87
4.1 Computational Logic.....	88
4.2 Formal Description of Modeling Logic Operators.....	89
4.2.1 Modeling Logic Operators.....	90
4.2.2 Elementary Graph Operators	91
4.3 Model Analysis Operators.....	92
4.3.1 Modeling Element Identification.....	94
4.3.2 Hierarchical Structure.....	97
4.3.3 Topological Structure.....	98
4.3.4 Material Characterization	100
4.3.5 Chemical Content	103
4.3.6 Mechanistic Characterization.....	104
4.4 Model Construction Operators.....	106
4.4.1 Modeling Elements.....	106
4.4.2 Topological Characterization.....	108
4.4.3 Chemical Content	112
4.4.4 Hierarchical Characterization.....	114
4.4.5 Material Characterization	118
4.4.6 Mechanistic Characterization.....	120
4.4.7 Behavioral Characterization.....	121
4.5 Model Consistency Operators	122
4.5.1 Hierarchical Consistency.....	123
4.5.2 Material Characterization Consistency	127
4.5.3 Species Topology Rules	129
4.6 Model Completeness Operators.....	131
4.7 Model Derivation Operators.....	133
4.7.1 Chemical Species Conservation Equation Derivation	133
4.7.2 Energy Conservation Equation Derivation	137
4.7.3 Chemical Reaction Rate Equation Derivation.....	140
4.7.4 Material-Content Characterization Equation Derivation.....	141
4.7.5 Phase Characterization Equation Derivation	145
4.7.6 Mechanistic Characterization Equation Derivation.....	148
4.7.7 Thermodynamic and Physical Properties of Phases Equation Derivation.....	149
4.7.8 Thermodynamic and Physical Properties of Fluxes Equation Derivation.....	151
4.8 Model Explanation.....	151
4.9 Extensions to Modeling Logic Operators	153
4.10 Supervisory Logic Operators.....	153

Chapter 5 The MODEL.LA Modeling Environment.....	159
5.1 Software Structure.....	160
5.2 Model Generator.....	160
5.2.1 Topological Structure.....	162
5.2.2 Hierarchical Structure.....	165
5.2.3 Chemical Characterization.....	170
5.2.4 Phenomena-Based Mechanistic Characterization.....	177
5.2.5 Phenomena-Based Model Summary.....	182
5.2.6 Mathematical Model Derivation.....	183
5.3 Properties Manager.....	185
5.3.1 Pure Species Properties.....	186
5.3.2 Binary Interaction Parameters.....	186
5.3.3 Material Behavior Analysis.....	186
5.4 Operations Manager.....	191
5.4.1 User Equations.....	192
5.4.2 Process Controllers.....	193
5.4.3 External Models.....	195
5.4.4 Operational Schedules.....	196
5.5 Numerical Engine.....	200
5.5.1 Display of Model Equations.....	201
5.5.2 Design Variable Specification.....	201
5.5.3 Index Analysis.....	203
5.5.4 Initial Conditions.....	203
5.5.5 Initial Guess Specification.....	204
5.5.6 Solution of Model Equations.....	204
5.5.7 DAE Systems Numerical Solution Methods.....	206
5.5.8 IPDAE Systems Numerical Solution Methods.....	208
5.5.9 Display of Numerical Results.....	210
5.6 Summary of MODEL.LA Modeling Environment.....	211
Chapter 6 Software Design of the MODEL.LA Modeling Environment.....	213
6.1 The Object Modeling Technique.....	213
6.2 MODEL.LA Modeling Element Object Models.....	214
6.3 MODEL.LA Modeling Environment Object Models.....	222
6.4 Functional Model of the MODEL.LA Modeling Environment.....	229
6.5 Summary of MODEL.LA Modeling Environment Software Design.....	233

Chapter 7 Phenomena-Based Modeling Examples.....	235
7.1.1 HDA Plant	235
7.1.2 Acetic Anhydride Plant.....	242
7.1.3 Dynamic Distillation Column Example.....	251
7.1.4 1-D Spatially Distributed Reaction and Separation Processes.....	254
7.1.5 2-D Tubular Reactor	257
7.2 Summary of Model Examples.....	260
Chapter 8 Conclusions and Recommendations	261
8.1 Research Contributions	261
8.1.1 Phenomena-Based Modeling Language.....	261
8.1.2 Formalized Modeling Logic.....	262
8.1.3 Computer-Aided Modeling Environment	262
8.2 Potential Impact on Modeling in Engineering Practice	263
8.3 Potential Impact on Undergraduate Chemical Engineering Education	264
8.3.1 Structuring of Modeling Activities.....	265
8.3.2 Classroom Deployment of MODEL.LA.....	265
8.3.3 Pedagogical Use of MODEL.LA	266
8.3.4 Unique Impact on Undergraduate Education	267
8.4 Directions for Future Research.....	269
8.4.1 Phenomena-Based Modeling Language Extensions.....	269
8.4.2 Integration with Molecular Modeling Tools	269
8.4.3 Implementation of Supervisory Logic	270
8.4.4 Standardization and Integration with External Modeling Tools	270
8.5 Conclusions	271
Bibliography	273
Appendix A MODEL.LA Context-Free Grammar.....	279
Appendix B Properties Manager	285
Appendix C Operational Schedules.....	299
Appendix D Jacketed-CSTR Model Equations.....	309
Appendix E 2-D Spatially Distributed Tubular Reactor Model Equations	315

List of Figures

Figure 3-1: Evolution of Process Model Representations	54
Figure 3-2: Example Production Tree	56
Figure 3-3: Phenomena-Based Model Production Tree	58
Figure 3-4: Expanded Phenomena-Based Model Production Tree	61
Figure 3-5: Modeled-Unit Production Tree	62
Figure 3-6: Hierarchical Structure Production Tree	63
Figure 3-7: Spatial Distribution Production Tree	64
Figure 3-8: Example Spatial Dimension Production Tree	65
Figure 3-9: Topological Structure Production Tree	67
Figure 3-10: Chemical Content Production Tree	68
Figure 3-11: Flux Production Tree	69
Figure 3-12: Flux Type Production Tree	69
Figure 3-13: Convective Flux Production Tree	70
Figure 3-14: Energy Flux Production Tree	71
Figure 3-15: Species Flux Production Tree	72
Figure 3-16: Flux Connectivity Production Tree	72
Figure 3-17: Material-Content Production Tree	73
Figure 3-18: Phase Instance Production Tree	74
Figure 3-19: Vessel Geometry Production Tree	75
Figure 3-20: Flux Allocations Production Tree	76
Figure 3-21: Phase Production Tree	77
Figure 3-22: Phase Identification Production Tree	77
Figure 3-23: Thermodynamic Phase Characterization Production Tree	78
Figure 3-24: Species Production Tree	79
Figure 3-25: Reaction Production Tree	80
Figure 3-26: Reaction Participants Production Tree	81
Figure 3-27: Reaction Kinetics Production Tree	81
Figure 3-28: Example Model Digraph	84
Figure 3-29: Example Model Derivation Tree	85
Figure 4-1: Modeled-Unit Digraph Representation	94
Figure 4-2: New Modeled-Unit Declaration	107
Figure 4-3: Declaration of Topological Structure	108
Figure 4-4: Declaration of Internal Flux	109
Figure 4-5: Declaration of Multi-Level Flux	110
Figure 4-6: Hierarchical Structure Characterization Example	116
Figure 4-7: Hierarchical Abstraction Example	117
Figure 4-8: Mathematical Model Derivation	152
Figure 4-9: Supervisory Logic Operators	154
Figure 5-1: MODEL.LA Modeling Environment Software Structure	160
Figure 5-2: MODEL.LA Graphical User Interface	161

Figure 5-3: Declaration of a Modeled-Unit	162
Figure 5-4: Declaration of a Convective Flux.....	163
Figure 5-5: Decomposition Flowsheet.....	164
Figure 5-6: Jacketed_CSTR Decomposition	165
Figure 5-7: Modeled-Unit Aggregation Dialog.....	166
Figure 5-8: Example Staged Modeled-Unit	167
Figure 5-9: Spatial Distribution Dialog.....	168
Figure 5-10: 1-D Distributed Heated Fin Example	170
Figure 5-11: Project Species Selection Dialog.....	171
Figure 5-12: Project Reaction Dialog	172
Figure 5-13: Modeled-Unit Chemical Content Characterization Dialog	173
Figure 5-14: Material-Content Declaration Dialog	174
Figure 5-15: Material-Content Geometry Declaration Dialog	174
Figure 5-16: Material-Content Flux Allocation Dialog	175
Figure 5-17: Example Vessel Geometry and Flux Allocation.....	177
Figure 5-18: Modeling Assistant: Edit Fluxes Tab.....	177
Figure 5-19: Convective Flux Characterization Dialog	178
Figure 5-20: Energy Flux Characterization Dialog	178
Figure 5-21: Species Flux Characterization Dialog.....	179
Figure 5-22: Project Reaction Rate Law Dialog.....	180
Figure 5-23: Project Data Summary Dialog.....	182
Figure 5-24: Modeled-Unit Template Selection Dialog	183
Figure 5-25: Model Inconsistency Dialog.....	184
Figure 5-26: Simulation Options Dialog.....	185
Figure 5-27: Species Database: Identification Properties	187
Figure 5-28: Species Database: Constant Properties.....	187
Figure 5-29: Species Database: Temperature Dependant Properties	188
Figure 5-30: Species Database: UNIFAC Groups Properties.....	188
Figure 5-31: Species Database: Binary Interaction Parameters for Equations of State	189
Figure 5-32: Species Database: Binary Interaction Parameters for Activity Coefficient Models	189
Figure 5-33: Phase Equilibrium Calculations Dialog.....	190
Figure 5-34: Phase Properties Calculations Dialog	190
Figure 5-35: Phase Diagram Dialog	191
Figure 5-36: User-Entered Equation Dialog.....	192
Figure 5-37: Declaration of Control Structures	193
Figure 5-38: Transmission Variable Selection Dialog	194
Figure 5-39: Control Law Specification Dialog.....	195
Figure 5-40: gPROMS External Model Definition Dialog	196
Figure 5-41: Specification of Operational Schedule.....	197
Figure 5-42: Discrete/Continuous Behavior of Scheduled Process Model.....	199
Figure 5-43: Numerical Engine Toolbar	200
Figure 5-44: Model Equations Dialog	200
Figure 5-45: Design Variable Specification Dialog	201
Figure 5-46: High Index Diagnosis Dialog	202

Figure 5-47: Initial Condition Specification Dialog.....	203
Figure 5-48: Initial Guesses Specification Dialog	204
Figure 5-49: Numerical Solver Specification Dialog.....	205
Figure 5-50: MODEL.LA Block Solver Dialog.....	206
Figure 5-51: Numerical Results Display Dialog.....	211
Figure 6-1: Object Modeling Notation for Classes.....	214
Figure 6-2: Modeling Element Class Object Model.....	215
Figure 6-3: Species Container Class Object Model.....	216
Figure 6-4: Reaction Container Class Object Model.....	216
Figure 6-5: Modeled-Unit Class Object Model.....	217
Figure 6-6: Flux Class Object Model.....	218
Figure 6-7: Material-Content Class and Phase Class Object Models	219
Figure 6-8: Reaction Class Object Model.....	220
Figure 6-9: Species Class Object Model.....	220
Figure 6-10: Modeling Elements Integrated Object Model	221
Figure 6-11: MODEL.LA Modeling Environment Object Model.....	222
Figure 6-12: Phenomena-Based Model Object Model	222
Figure 6-13: Mathematical Model Object Model.....	223
Figure 6-14: Model Generator Object Model	224
Figure 6-15: Properties Manager Object Model	225
Figure 6-16: Properties Database Object Model.....	226
Figure 6-17: Operations Manager Object Model	227
Figure 6-18: Numerical Engine Object Model	228
Figure 6-19: Overall Model Derivation and Solution Functional Model.....	229
Figure 6-20: Model Derivation Functional Model	230
Figure 6-21: Property Correlation Generation Functional Model.....	231
Figure 6-22: Mathematical Model Specification and Solution Functional Model.....	232
Figure 6-23: Model Solution Functional Model.....	233
Figure 7-1: Input-Output Level Design for HDA Plant.....	237
Figure 7-2: Reaction and Separation Section Design for HDA Plant.....	238
Figure 7-3: Separation Section Design for HDA Plant.....	239
Figure 7-4: Reaction Section with Energy Integration for HDA Plant.....	240
Figure 7-5: Distillation Column Design for HDA Plant.....	241
Figure 7-6: Simulation Results for HDA Plant Base Case Design	242
Figure 7-7: Input-Output Level Design for Acetic Anhydride Plant	244
Figure 7-8: Chemical Species and Reactions for Acetic Anhydride Plant Design.....	245
Figure 7-9: Simulation Results for Input-Output Level Design of Acetic Anhydride Plant	246
Figure 7-10: Reaction and Separation Section Design for Acetic Anhydride Plant	247
Figure 7-11: Reactor Design for Acetic Anhydride Plant.....	248
Figure 7-12: Separations Subsystem Design for Acetic Anhydride Plant.....	249
Figure 7-13: Economic Potential for Base Case Design of Acetic Anhydride Plant	250
Figure 7-14: BTX Dynamic Distillation.....	251
Figure 7-15: PI Control of Dynamic Distillation Column.....	252
Figure 7-16: Closed Loop Dynamic Response of BTX Distillation Column.....	253
Figure 7-17: 1-D Spatially Distributed Reaction and Separation Process	254

Figure 7-18: Structure of 1-D Spatially Distributed Tubular Reactor and Gas Absorption Column	255
Figure 7-19: 1-D Spatially Distributed Reactor and Absorption Column Results	257
Figure 7-20: 2-D Spatially Distributed Tubular Reactor Example.....	258
Figure 7-21: 2-D Spatially Distributed Tubular Reactor Simulation Results.....	259
Figure C-1: Generic Structure of Hybrid System.....	300

List of Tables

Table 4-1: Intrinsic Declarative Graph Operators	91
Table 4-2: Intrinsic Graph Assignment Operators.....	92
Table 4-3: Elementary Procedural Graph Operators	93
Table 4-4: Modeling Element Identification Operators	95
Table 4-5: Specialized Modeling Element Identification Operators.....	96
Table 4-6: Hierarchical Structure Analysis Operators.....	97
Table 4-7: Topological Structure Analysis Operators	99
Table 4-8: Material Characterization Analysis Operators.....	102
Table 4-9: Chemical Content Analysis Operators	103
Table 4-10: Mechanistic Characterization Analysis Operators	105
Table 4-11: Modeling Element Declaration Operators.....	107
Table 4-12: Topological Structure Declaration Operators	111
Table 4-13: Chemical Content Declaration Operators	113
Table 4-14: Hierarchical Structure Characterization Operators.....	115
Table 4-15: Material Characterization Operators.....	119
Table 4-16: Mechanistic Characterization Operators	121
Table 4-17: Behavioral Characterization Operators	122
Table 4-18: Hierarchical Consistency Operators.....	126
Table 4-19: Material Characterization Consistency Operators	129
Table 4-20: Species Topology Consistency Operators.....	131
Table 4-21: Model Completeness Operators	132
Table 4-22: Species Conservation Derivation Operators.....	135
Table 4-23: Species Aggregation Derivation Operators.....	136
Table 4-24: Convective Species Transport Derivation Operator	137
Table 4-25: Energy Conservation Derivation Operators	138
Table 4-26: Internal Energy Aggregation Derivation Operators.....	139
Table 4-27: Energy Transport Derivation Operator	140
Table 4-28: Chemical Reaction Rate Derivation Operators.....	141
Table 4-29: Material-Content Aggregation Operators	142
Table 4-30: Phase Equilibrium Derivation Operators.....	144
Table 4-31: Phase Species Aggregation Operators	145
Table 4-32: Species Fraction Summation Operators	146
Table 4-33: Species Holdup Derivation Operators	146
Table 4-34: Species Concentration Derivation Operators	147
Table 4-35: Phase Density Derivation Operators.....	148

Table 4-36: Phase Internal Energy Derivation Operators.....	148
Table 4-37: Mechanistic Characterization Operators.....	149
Table 4-38: Physical and Thermodynamic Phase Property Operators.....	150
Table 4-39: Physical and Thermodynamic Flux Property Operators.....	151
Table 4-40: Level-1 Supervisory Operator.....	156
Table 4-41: Level-2 Supervisory Operator.....	157
Table 5-1: Heated Fin Model Equations.....	169
Table 5-2: Heated Fin Model Equations with Mechanistic Characterizations.....	169
Table 5-3: Geometric Vessel Configurations for a Material-Content.....	176
Table 5-4: MODEL.LA Reaction Rate Law Templates.....	181
Table 5-5: Summary of gPROMS IPDAE Solution Methods.....	208
Table 7-1: Design Objectives for HDA Plant.....	236
Table 7-2: Design Objectives for Acetic Anhydride Plant.....	243
Table 7-3: PI Controllers of BTX Dynamic Distillation.....	253
Table 7-4: Reaction Data for 1-D Spatially Distributed Reaction and Separation Process.....	254
Table 7-5: Solution Methods for 1-D Spatially Distributed Reactor and Absorption Column.....	256
Table 7-6: Reaction Data for 2-D Spatially Distributed Tubular Reactor Example.....	258
Table 7-7: Solution Methods for 2-D Spatially Distributed Tubular Reactor Example.....	259
Table 7-8: Summary of Phenomena-Based Modeling Examples.....	260
Table B-1: Pure Species Identification Properties.....	286
Table B-2: Pure Species Constant Value Properties.....	287
Table B-3: Pure Species Temperature-Dependent Properties.....	288
Table B-4: Pure Species Temperature-Dependent Property Correlations.....	289
Table B-5: Properties Manager Equations of State.....	291
Table B-6: Properties Manager Activity Coefficient Models.....	292
Table B-7: Thermodynamic and Physical Property Correlations for Phases.....	293
Table C-1: Operational Schedule Elements.....	302
Table C-2: Example gPROMS Schedule Translation.....	307

Chapter 1

Introduction

Modeling is the quintessential activity that characterizes modern chemical process systems engineering. Physical experiments on real processes are expensive, time-consuming, and often potentially hazardous. Fortunately, models provide alternative means to answer questions about the behavior of such processes through computational experiments. While the potential benefits of process modeling are commonly acknowledged, its use is not nearly as widespread. Rather, the development of chemical process models is usually left in the hands of a select group of modeling experts. This is because the lack of formalization and systematization associated with model development leads most students and practicing engineers to view modeling as an art, rather than as a science.

Several computer-aided modeling tools have been proposed to assist the process of model development. While these tools provide varying degrees of assistance to some aspects of the modeling activity, they are inadequate to alleviate the modeling bottleneck that restricts the use of models in engineering practice and in undergraduate education. This is because none of these computer-aided tools possess an explicit understanding of the chemical engineering concepts behind model development. Consequently, existing tools must either rely on a library of predefined models or must provide a computational language for specifying numerical calculations to be performed or sets of equations to be solved. Use of the former is limited by its inherent inflexibility, while use of the latter is limited by its inherent complexity.

To provide high-level modeling support to all chemical engineers, not just modeling experts, a computer-aided modeling environment is needed that lifts the level of model development from procedural calculations and mathematical equations to the fundamental

concepts that characterize chemical engineering science. With this ability, the computer will allow engineers to develop models at the level of chemical engineering knowledge—through explicit assumptions about structure, physicochemical phenomena, and mechanistic characterizations. However, before such a tool may be developed, a high-level language is needed that provides a vocabulary for description of such models. Once this language is in place, it will provide the basis for formalized modeling logic that enables the computer to construct mathematical models as a human engineer would from a phenomena-based description, to provide feedback on model completeness and consistency, and to provide guidance during the modeling activity.

1.1 Chemical Process Modeling Needs in Engineering Practice

Models are utilized to solve a variety of engineering problems related to a chemical process throughout its lifetime. Process engineering activities such as design, optimization, analysis, control, scheduling, diagnosis and training all rely on mathematical models that provide a basis for computational experiments used to solve some engineering problem. In this manner, all aspects of a process, its operation, economics, robustness, and safety, can be verified through development of a valid model.

Properly documented, such models should become valuable assets for a company. The motivation, ideas, assumptions, and decisions behind their development are an investment in engineering knowledge. A company that can readily access, accumulate, analyze, and reapply this knowledge gains a major advantage over its competitors.

Unfortunately, experience has shown that these benefits are not realized because the investment in model development is lost. Since the knowledge behind a model is not retained nor explicitly linked to model development, most modeling efforts start from scratch. Not only is there little retention of modeling knowledge from process to process, but even the same process is modeled “over and over” for different engineering applications (e.g., process design and process control). Obviously, under these circumstances the accumulation of modeling assets is minimal.

The complex process engineering problems of today demand computer-aided modeling tools that extend beyond the paradigm of numerical computations. Since high-powered personal computers and efficient numerical algorithms are now commonly available, the substantial effort required for model formulation, verification, documentation, modification, and reuse is now

viewed as the key obstacle preventing a more prevalent use of process modeling.

A properly designed modeling environment is needed which will alleviate the weaknesses associated with traditional computer-aided process modeling. These weaknesses include:

1. The time and cost associated with model development are high,
2. Model development is an ad hoc activity carried out by a select group of experts,
3. Engineers and scientists in various backgrounds cannot readily contribute their expertise in a collaborative modeling effort,
4. Models are not defined at the common level of chemical engineering assumptions, but rather in the narrow terms of a language specific to a given modeling tool,
5. The resulting models are difficult to document and maintain,
6. The reuse of models is minimal, as they tend to be task-specific and linked to solution procedures, and
7. Computer assistance is primarily limited to numerical computations and the solution of equations.

Until a new computer-aided modeling paradigm is developed that addresses these issues, the potential benefits of chemical process modeling cannot be fully realized.

1.2 Chemical Process Modeling Needs in Undergraduate Education

The core material of chemical engineering education has been developed over many decades into a focused and well-organized curriculum centered around such courses as heat and mass transport, fluid mechanics, thermodynamics, kinetics, separation processes, and reaction engineering. Instruction in these core courses focuses primarily on analysis—the deduction of some behavior of a given system. For example, problems are phrased as *plot the temperature versus time*, *find the phases present and species concentrations*, *determine the velocity profile*, etc. Rarely is such behavior determined experimentally. Rather, it is determined by solution of models derived from conservation principles and mechanistic *laws* (e.g., ideal gas law, Raoult's law, Fick's law) describing physical and chemical phenomena.

For simplicity, the fundamental concepts and phenomena introduced in these courses are presented in the context of idealized situations and may be characterized as *set pieces* of the foundation material. Since students have little prior knowledge in describing these situations, the

derivations of models of these phenomena are only passively presented to the students. These models typically take the form of one or more mathematical equations prominently boxed off in the textbook or on the blackboard. In the interest of time, the context of a model (i.e., its motivation, objectives, assumptions, limitations, and derivation) are treated as only of secondary importance. Typically, emphasis then quickly shifts to mathematical techniques required to solve the resulting equations. Subsequent homework problems, tagged as “applications”, are actually similar set pieces that require few decisions of the student other than selection of which predefined model equations (that best accommodate given data) and/or solution methods are to be used.

These set pieces of the foundation material are used in order to make the fundamental concepts and solution methods introduced as clear as possible to the student and are a necessary part of the educational process. However, it is evident that the focus of the chemical engineering syllabus needs to be expanded. The true measure of engineering ability is not the understanding of idealized textbook examples but rather the application of fundamental concepts in developing concise, appropriate models within the context of a particular engineering problem. Formulating and deriving an adequate mathematical model is a greater challenge and yields far more rewards to the student than just understanding and appreciating models derived in a textbook or by an instructor. When students are required to apply fundamental concepts to derive models in unfamiliar contexts, the necessary associations are not made and they are often confused as to the appropriate decisions/assumptions that are required.

Comments from instructors of all undergraduate chemical engineering courses at both MIT (Mohr, 1996) and the University of California at Berkeley (Foss, 1996) have illustrated that such views are common:

- Students understand basic concepts but have a poor idea of when and how to apply them to a given engineering problem.
- The foundations of disciplined and effective modeling must be laid down at the beginning, in the introductory sophomore course in chemical engineering.
- The current teaching of modeling is ad hoc and taught primarily by a sequence of examples. Thus, any systematization and discipline brought to the modeling activity will be extremely beneficial.

- The effectiveness of the modeling activity is largely left to the intellectual capabilities of the individual student. Consequently, the creativity of engineering students in general suffers significantly. Assistance is needed to enhance the modeling creativity of students overall.

Unfortunately, students are never presented an explicit modeling methodology. Rather, they can only infer modeling techniques from exposure to a long sequence of examples. The ad hoc nature of modeling leads to frustration for both the student, who struggles to adapt poorly understood existing models to other problems, and the instructor, who must puzzle over what the student is doing wrong and what information is not properly understood.

One approach for addressing the need to present a modeling methodology would be to develop pre-wired tutorial software which would lead the student through examples of model development for non-trivial engineering problems. However, in the context of teaching modeling, the benefits of producing such software would be quickly exhausted. The prespecified paths and alternatives through which students could proceed would inhibit them from directing and experimenting with model development and would risk trivializing the complex task of modeling. On the other hand, a well-designed computer-aided modeling environment should enable students to express their notions of what a proper model should be. This would allow students to proceed in an structured process of model development. First, the important structure and physicochemical phenomena are articulated. Next, these phenomena are characterized mechanistically. This description provides the basis for derivation of the model equations. The required data to solve the problem are then identified. Finally, the behavior of the process model is observed, providing immediate feedback on the applicability of the crafted model. Obviously, chemical engineering students must possess the skills for deriving and solving equation-based models independently. In various problem-solving contexts, however, by taking primary responsibility for equation formulation and solution, the computer would give students the opportunity to concentrate on the chemical engineering assumptions and decisions needed to synthesize a model. When students make the transition from passive onlookers to active participants during model development, they gain the experience, knowledge, and confidence that is necessary to solve the real engineering problems they will face throughout their education and their careers.

1.3 Potential Role of the Computer in Process Modeling

Chemical engineers have long since embraced the computer as an algorithmic tool. The speed and efficiency with which computers can perform numerical procedures is well-appreciated. Yet advances in computer-aided process modeling over the past few decades have essentially been limited to the realm of numerical computations. Modelers should expect more assistance from computers than solely the ability to solve larger and larger numerical problems with ever increasing speed. Instead of just providing passive data structures for organizing computations, the computer should possess “modeling knowledge” that would allow it to communicate with and assist the engineer at the level of chemical engineering understanding.

While all student and practicing chemical engineers must possess sound modeling fundamentals, they should not be expected to be computer programmers or highly specialized software experts in order to develop models. Process modeling should not be a career goal—all chemical engineers need to create and use models to solve a wide variety of problems. Furthermore, the computer should provide a collaborative environment for modeling during process engineering activities, allowing experts in varying backgrounds (e.g., process design, physical chemistry, reaction kinetics, process control, etc.) to readily contribute to model development in parallel, without each having to understand the fine details of every part of the model. Models should be viewed as repositories of engineering knowledge, not as collections of subroutines or equations.

The computer has the potential to unleash the benefits of process modeling, but this is not possible without a formal representation of the fundamental principles of chemical process modeling. This representation would allow systematization of the modeling activity, where the computer provides varying degrees of support to the engineer. Certain modeling tasks can be completely automated by the computer. Other modeling tasks can be viewed as a structured interaction where the computer guides the activity but the engineer makes the key modeling decisions based on the context of the particular engineering problem. Other modeling tasks that require human understanding and creativity cannot be automated, but the computer can still provide a framework for explicitly documenting the motivation for these tasks, the rationale for key decisions made, alternatives considered, etc. The computer should also have internal logic to generate models under different contexts or levels of detail, check models for inconsistencies or

incompleteness, and review model simulation results to check the validity of assumptions based on chemical engineering principles.

1.4 Existing Computer-Aided Process Modeling Tools

Many computer-aided tools designed to facilitate certain aspects of process modeling in industry and education have been developed. However, all have failed to alleviate the perceived *process modeling bottleneck*. This is because none has satisfactorily addressed all the key industrial and educational chemical engineering modeling needs. Several examples of these approaches, their key features, and their shortcomings are given below.

1.4.1 Sequential Modular Flowsheet Simulators

The concept of *unit operations*, established by Arthur Little at MIT in 1915, helped to define the profession of chemical engineering. Decades later, unit operations are still the paradigm on which the most popular type of commercial chemical process modeling software, *sequential modular flowsheet simulators*, is based. Examples of these tools include ASPENPLUS by AspenTech, HYSIS by Hyprotech, and PRO/II by SimSci. These simulators allow the generation of process model flowsheets through a structured integration of predefined unit operation models (e.g., heat exchangers, distillation columns, CSTRs, etc.) from a library. Chemical species and, where applicable, chemical reactions are then added to the flowsheet model. Each proprietary unit operation model from the simulator library encompasses a computational procedure which performs predetermined calculations on a fixed set of input variables to yield values of prespecified outputs. Calculations are performed in sequential modular fashion, where the output values of each unit become the input values of the subsequent unit, as dictated by the topology of specified process streams. During these calculations, the thermodynamic and physical properties of materials are determined by pure species properties from a database and the selected thermodynamic and physical property models of the materials.

The *language* of unit operations provided by sequential modular flowsheet simulators helps modelers deal with the complexity of chemical process modeling by abstracting the details of the embedded solution procedures. Although these tools are relatively easy to use and have gained wide acceptance in industry, their use is restricted due to several inherent limitations. In a flowsheet, each process unit is essentially selected as a *blackbox* from a finite library of available

models. Although some parameters may be user-specified, the model of each module is fixed. The applicability of these inflexible models may be questionable because assumptions used in their derivation may not be explicitly stated or readily ascertained. Furthermore, the degree of detail these models require may be more or less than that which is needed for a particular engineering application.

The fundamental limitation of sequential modular flowsheet simulators is that the level of modeling granularity provided by their inflexible unit operation models is too coarse. These libraries cannot anticipate the requirements of non-standard unique or novel processes, where the need for models is often most critical. Furthermore, the modeling of distributed systems and the development of hierarchical multi-level process models are not adequately supported. While most sequential modular simulators provide a facility for adding “new” unit operations to the existing model library, minimal assistance is provided for developing these model definitions, which must be procedural to maintain the sequential modular calculation paradigm.

1.4.2 Programming Languages

When sequential modular flowsheet simulator models are inadequate or unavailable altogether, a process model must be characterized in terms of elementary physical and chemical phenomena rather than unit operations. From this description, a set of mathematical equations meant to represent the behavior of the system may be developed from chemical engineering first principles. The burden imposed on an engineer who must develop such a model for a complex process can be overwhelming. The tasks involved in deriving such a model from first principles may be decomposed into the following four aspects:

1. *Declaration of Assumptions:* The system to be modeled is identified, and within the context of the modeling objectives, the variables of interest are specified and assumptions are made regarding structure, relevant physicochemical phenomena, and characterization of materials.
2. *Derivation of Equations:* Conservation principles, constitutive relationships, design correlations, and thermodynamic and physical property values are used to derive a consistent set of mathematical equations based on assumptions from the first aspect.

3. *Solution of Equations:* Appropriate numerical algorithms are identified and the equations from the second aspect are encoded in a form suitable for solution using a procedural programming language (e.g., Fortran, C, Pascal, etc.).
4. *Validate Model:* After the programming code is debugged and numerical convergence is obtained, the modeled behavior of the process is observed. If necessary, the first, second, and third aspects are reviewed and repeated.

Obviously, to accomplish this task unassisted the modeler must possess a broad range of capabilities. The first aspect requires an understanding of the modeling context and insight into which phenomena are of practical relevance and significance to the problem application. The second aspect requires comprehension of all the chemical engineering science involved in logically deriving the equations, along with the ability to determine if they are mathematically well-defined. The third aspect requires not only an extensive knowledge of numerical methods but also familiarity with a programming language needed to implement them. Finally, the fourth aspect requires extreme patience. Due to the current lack of structured logic and formalized procedures to guide these tasks, process modelers must rely largely upon not only the science of chemical engineering knowledge and mathematical ability, but also the art of intuition, insight, and experience.

Moreover, since the resulting procedural models are intrinsically linked to the methods used for solution, they are inherently difficult to reuse or modify to model a similar system or even the same system within the context of a different process engineering problem (e.g., diagnosis rather than process optimization). This is because the chemical engineering modeling knowledge is embedded and obscured in the solution techniques of the corresponding numerical algorithms. As a result, the investment in developing the process model must usually be repeated for each implementation.

1.4.3 Spreadsheets

The relatively low cost and easy learning curve of spreadsheeting software (e.g., Microsoft Excel) have led to their wide use in industry and in undergraduate education for many modeling applications. While many engineers find them easy to use, spreadsheets are also inherently designed for procedural computations and suffer from the same limitations as programming

languages. The distinction is blurred further by macro languages (e.g., Visual Basic in Microsoft Excel) that are provided which essentially turn a spreadsheet into an interface to underlying procedural programs.

1.4.4 Equation-Based Process Modeling Tools

In order to alleviate the problems associated with procedural model formulations, several *equation-based modeling languages* (e.g., ASCEND, OMOLA, and gPROMS) have been developed. These tools allow the modeler to declare large systems of equations in symbolic form. The computer then takes primary responsibility for selecting and implementing the appropriate numerical methods to determine the results. These tools extend beyond general equation-solvers by providing logical operators to allow conditional model definitions and by implementing certain object-oriented programming concepts to organize the mathematical description of models.

In object-oriented programming, *classes* are structured data types used to describe a set of similar *objects*. Each class definition encompasses a characterizing description, defined by a set of values (*attributes*), and a functionality, defined by a set of procedures (*methods*) which operate on the attributes. An object is created by instantiating a class (i.e., assigning values to its attributes). Each instantiation of a class produces a new object which is coexistent but independent of previous objects produced from that particular class. *Inheritance* is used to establish a multi-level class hierarchy, where a child class inherits the attributes and methods of its parent class. The child class may then be further refined by adding additional attributes or methods, or by modifying methods inherited from the parent class.

Generally, in application to equation-based modeling tools, classes provide a means of abstraction by grouping variables and equations pertaining to a certain system into a single object. Inheritance reduces redundant modeling by grouping similar classes under a single parent class and also promotes model reuse by allowing new models to be defined through modification and extension of an existing model class. For example, in ASCEND (Piela, 1991, Piela et al, 1991), *atoms* are an object class used to represent variables. *Models* are classes defined by the user to encompass a set of variables and mathematical equations. Inheritance allows atoms to be partially specified from previously defined atoms and models from instances of other models and atoms. OMOLA (Nilsson, 1993, 1995) is designed for the description of dynamic models for simulation

purposes. In OMOLA, the *model* class is the root class of all user-defined models. Models are grouped into two types, *primitive* and *structured*. Primitive models are defined by attributes describing *parameters* (constant values), *variables* (time-varying values), *realizations* (equations and constraints describing behavior of the variables) and *terminals*. Terminals provide a means of communication of variables between models. Structured models allow hierarchical equation models defined by attributes which identify the corresponding submodels and their connected terminals. Neither Ascend nor OMOLA is limited to chemical process modeling but are both designed to also support equation-based modeling in any other technical discipline. gPROMS (Barton, 1992, Barton and Pantelides, 1993) is designed for the dynamic simulation of combined discrete and continuous processes. *Processes* are formed by the application of *tasks* to instances of *models*. The user-defined models encompass continuous mathematical equations meant to describe the behavior of a modeled system. Models are defined by attributes describing *parameters* (constant values), *variables* (time-varying values), *equations* (algebraic and ordinary differential equations describing the behavior of the variables) and *streams* (corresponding to terminals in OMOLA). The user may also specify tasks which represent discrete procedures, such as control actions or disturbances, imposed on the system. Both models and tasks may be defined hierarchically through inheritance from other models or tasks, respectively. gPROMS has been extended to solve partial differential and integral equations in addition to ordinary differential and algebraic systems of equations (Oh, 1995, Oh and Pantelides, 1996). gPROMS is now available as a commercial product by Process Systems Enterprises Limited. Other examples of equation-based modeling tools include SPEEDUP (Perkins 1982) which has been commercialized by AspenTech, and ABACUSS which is the only equation-based modeling tool capable of integrating systems of numerically high-index differential and algebraic equations (Feehery and Barton, 1996).

By automatically determining the numerical solutions to the user-specified equations, these mathematical tools are a significant aid to equation-oriented modeling. Since the equation-based model definition is not intrinsically linked to a particular solution method, the equations may be readily used to solve for different sets of unknown variables, for optimization, or for regression purposes. Also, since only the equations and not the procedural solution method must be specified, the amount of coding needed to develop a new model or modify an existing one is

significantly reduced. While object-oriented programming concepts such as object classes and inheritance facilitate the writing and organization of the model equations, it is clear the focus of these mathematical tools is to expedite the third aspect of modeling, the solution of equations using numerical algorithms.

The mathematical modeling tools described above are capable of solving systems of thousands of equations. However, these solution capabilities cannot be fully exploited unless the correct, consistent generation and maintenance of these complex sets of equations can be ensured. The mathematical modeling tools cannot provide this assistance because they focus on equation-solving methods. However, to achieve widespread use of modeling, assistance for the first and second aspects of modeling, the declaration of assumptions and the derivation of equations, may be the most critical. It has been observed (Denn, 1986) that “the truly challenging aspect of modeling is in the use of physical principles to arrive at a proper mathematical formulations.” Similarly, in (Aris, 1979), “it is comparatively easy to teach the method of solution of standard mathematical equations, but much harder to communicate the ability to formulate the equations adequately and economically.” Since the equation-based description of models in the mathematical modeling tools is essentially context-free and not explicitly linked to chemical engineering concepts, the computer is unable to offer assistance beyond numerical advice such as a degrees-of-freedom or an index analysis.

Furthermore, although the symbolic (rather than procedural) form of these equations facilitates reuse, their applicability may be uncertain because assumptions used in deriving these equations are not explicitly maintained. Thus, it is the responsibility of the modeler to provide comments explaining assumptions used as the basis of model equations, to analyze the consistency and logic of these assumptions, and to correctly derive the equations. These tasks are neither assisted nor enforced by any of the equation-based modeling tools. This can especially lead to difficulty in model editing and analysis. For example, to avoid redundant modeling it would be ideal if one continuously evolving model could be used over the lifetime of a project. Over time, such a model may grow to hundreds or thousands of equations while modifications are made by several different modelers. However, whenever an assumption is changed or added, the modeler must analyze the set of existing model equations (which may or may not be consistently commented), determine the modifications necessary throughout the system of equations, then

implement and document these changes. As a result, this task may quickly become unwieldy as a model grows in complexity. Obviously, further assistance to the modeler is also required for the first and second aspects of the modeling process, the declaration of assumptions and derivation of the corresponding mathematical equations.

1.4.5 Summary of Existing Computer-Aided Process Modeling Tools

Each of the existing computer-aided modeling tools do facilitate certain aspects of process modeling. However, they fall far short of providing the high-level of modeling assistance that is envisioned computers can provide. This is because these computer-aided tools are limited by the *language* that they use to communicate with the human modeler.

Modular flowsheet simulators provide the traditional *language of unit operations*. While this language is intuitive to chemical engineers, the inflexible and blackbox nature of these models restrict their use greatly. The coarse modeling granularity provided by these tools limits their application to the modeling of essentially well-understood, rather than unique and novel, processes.

Programming languages provide the *language of computational procedures*, which maximizes flexibility for developing process models. However this medium is tailored for the description of solution procedures for a mathematical model, not expression of the model itself. Use of these languages requires proficient programming ability, wide knowledge of numerical techniques, and a significant investment of time and effort. Moreover, the procedural models resulting from these efforts are inherently difficult to reuse or modify to model a similar system or even the same system within the context of a different process engineering problem. Spreadsheets software is more user-friendly than programming languages, but is also designed essentially for procedural computations and suffers from the same limitations as programming languages.

Equation-based tools overcome these procedural restrictions by providing the *language of mathematical equations*. These computer environments possess advanced equation-editing capabilities designed to facilitate the compilation of mathematical relationships. However, computer assistance to the modeler is primarily limited to the numerical solution of equations. The modeler is still left solely responsible for making the necessary modeling assumptions and

simplifications, analyzing the logic and consistency of these assumptions, and deriving the corresponding equations. The experience and skill required to construct these equation-based models efficiently and reliably still restricts their development to the realm of modeling experts. Consequently, most engineers are reluctant to pursue this time-consuming, error-prone, and difficult to document approach, leading them to abandon the modeling effort or to limit the number of alternatives they will consider during model development. Furthermore, since the resulting model consists of mathematical equations, the assumptions used in the model definition become implicit, inhibiting reuse of the model.

1.5 Physicochemical Phenomena-Based Process Modeling

By defining appropriate primitives and means of combination and abstraction, engineers in several disciplines have created specialized problem-oriented modeling languages designed for various applications. These languages provide facilities that lift the task of model formulation above the level of mathematical equations and calculation procedures. Examples include the computer-aided languages of *electrical engineering networks* (Sussman and Steele, 1980) for modeling circuits in terms of primitives that form discrete electrical elements (e.g., resistors, capacitors, inductors, etc.) and *civil engineering networks* (Maher, 1988) for modeling structures from discrete physical elements (e.g., girders, rods, beams, etc.). For chemical process engineering, the language of *unit operations networks* is no longer adequate to model the complexities of modern processing systems. Rather, a more elementary characterization of chemical processes is required which would allow processes to be represented as networks of interacting *physicochemical phenomena*. Thus, elementary physical and chemical concepts such as system, flow, reaction, and diffusion can be integrated to form phenomena-based models of not only traditional unit operation systems but also unique and novel processes. These phenomena-based model descriptions can then be used to automatically derive the requisite mathematical model equations from chemical engineering first principles. In this manner, a high-level process modeling environment based on the *language of elementary physical and chemical phenomena* can be developed that combines the high-level approach of unit operation-based simulators with the power and flexibility of equation-based modeling tools.

1.5.1 Proposed Phenomena-Based Modeling Approaches

As reviewed by Marquardt (1996), the representation of process models through the description of elementary physical and chemical phenomena has been approached by several researchers. This direction of research was initially established by the development of the process modeling language MODEL.LA (Stephanopoulos et al, 1990a, 1990b), on which preliminary ideas for this work are based. The key language elements of MODEL.LA are divided into two categories, structural and functional. *Generic-units*, *ports*, and *streams*, are used to describe the structural characteristics of a modeled process, while the *modeling-scope*, *constraints*, and *generic-variables* describe the functional characteristics. Similar to object-oriented programming concepts, each modeling element is associated with an object class, which is described by a set of attributes. Generic-units represent a system delimited by its boundaries, and are defined by attributes specifying structural components, modeling assumptions, and constraints. Ports, which may be of type convective, material, energy, or information, represent boundaries through which generic-units interact. Streams link ports of the same type between separate generic-units. The modeling-scope encapsulates and explicitly documents all modeling assumptions and constraints describing the generic-units. Constraints represent mathematical relationships derived from the modeling assumptions, and are composed of generic-variables and mathematical operators. Generic-variables encapsulate characteristics of a particular process quantity. Semantic relationships are provided to allow specialization (through inheritance and class membership), specification (through composition, communication, and description), abstraction and disaggregation (through hierarchical structuring), and definition (through characterization) of the modeling elements. The MODEL.LA language is designed for derivation of the model equations through mass, species, and energy balances for each generic-unit. The flux terms of each balance equation may be determined through summation or subtraction of the transferred quantities over the corresponding ports, while the generation or consumption terms may be determined through summation or subtraction over the corresponding sources or sinks specified for the balanced quantity. The form of each term is determined by translation of the assumed mechanism of a declared phenomenon into a constitutive equation. The implementation of MODEL.LA was limited to the modeling of static, lumped systems.

Vazquez-Roman (1992) and Perkins et al (1994) describe a prototype environment for

modeling lumped, dynamic systems based on a purely physical description. In that work, a *process* is a set of *vessels* whose *ports* are linked by *connections*. A vessel is characterized by its geometry (describing shape, size and port locations). Vessels contain *phases* which exchange material and/or energy with other phases. These interactions occur through connections according to a set of *transfer laws*, which are specified by user-defined assumptions regarding the relevant physicochemical phenomena. The thermodynamic state of each phase is characterized by the masses of each chemical component present, the internal energy, and the pressure. Controllers are modeled as *blackboxes*, which relate a state variable of a phase to a controller output. Species and energy balance equations are made for each phase. The terms of the balance equation include an accumulation term (which determines the dynamic behavior of the system), a generation term (due to specified chemical reactions), and a term for each defined transfer (as described by the transfer laws). The balance equations need to be coupled with thermodynamic relations which determine the phases present in a vessel, methods for physical and thermodynamic property calculation, pressure and volume relations for each vessel, and controller laws in order to compile the set of equations needed to carry out dynamic simulations.

In the context of the frame-based data model VEDA, Marquardt has proposed using a general systems theory approach to formalize the modeling knowledge for chemical engineering processes (Marquardt, 1996). In that work, modeling objects are divided into two categories, substantial and phenomenological. Substantial modeling objects (including *devices* and *connections*) represent structure while phenomenological modeling objects (including *variables* and *equations*) represent the behavior of substantial objects. These elementary modeling objects also may be combined to form composite devices, connections, variables, and systems of equations. Devices represent any delimitable part of a process. The role of a device is to determine its state properties from known fluxes from the surroundings. Subclasses of devices include generalized phases and signal transformers. Connections represent entities situated between devices. The role of a connection is to transform a driving force (as determined by the known states of two adjacent devices) into a flux. Subclasses of connections include signal and phase connections. From the phenomenological modeling objects (which include state variables, balance equations, constitutive equations, and constraints) associated with the substantial modeling objects, the equations of the model may be derived.

The prototype systems described above and other similar approaches (e.g., Preisig, 1995; Woods, 1993; Cho, 1998) have dealt primarily with the formulation of generic object-oriented classes for representation of hierarchical systems whose behavior is described in terms of physicochemical phenomena.

1.5.2 Summary of Proposed Phenomena-Based Modeling Approaches

Each of the phenomena-based modeling approaches described above proposes different *terms* (e.g., generic-unit, vessel, or device) meant to describe similar concepts. The influence of object-oriented programming techniques, especially with respect to classes and inheritance, is evident. However, in their current form these works are best characterized as *vocabularies*, rather than high-level computer-aided modeling tools. Evaluation of these vocabularies, and even meaningful delineation between them, is difficult. Furthermore, while the presentation of class taxonomies does seem appealing and often intuitive, it cannot be viewed as the culminating endpoint of this area of research. Rather, they can only be evaluated in light of a much larger and more ambitious goal: *To enable all chemical engineers to readily build and use models by supporting the modeling activity at the level of chemical engineering knowledge.*

Obviously, these research efforts are still in their infancy and before the benefits of any phenomena-based modeling approach can be appraised, several issues must be resolved:

1. The syntax and semantics of a phenomena-based modeling language must be formalized. Existing approaches have provided only common-sense, by-example, unsystematized explanations of these aspects. Most importantly, the impact of assumptions regarding a phenomena-based model on the resulting mathematical model has not been explicitly explained. As a result, one must rely on intuition to interpret these models.
2. The proposed vocabularies provide means for the phenomena-based representation of process models. However, the logic necessary for the selection, instantiation, and combination of these object classes in the context of chemical engineering modeling is unclear. This is because the procedural aspects of model development, necessary for the computer to provide high-level modeling assistance, have not been characterized. While Jarke and Marquardt (1995) describe generic

representations for tasks performed during the modeling process, it is impossible to meaningfully implement or evaluate such ideas without a formalized modeling language as a basis.

3. The integration of these ideas into computer-aided modeling tools has not passed the conceptual prototype stage. Pantelides and Britt (1995) stress that the implementation and practical application of these ideas is essential for assessing their value.

The phenomena-based modeling approach does promise to enable the computer to provide high-level modeling support. However, for this to be possible, these three issues must be addressed. This final issue of implementation is perhaps the most critical, because the primary contribution of this research area will be to facilitate the model development process for all chemical engineers, not just modeling experts. Conceptual designs on paper alone do not achieve this. Until these ideas are implemented in a computer-aided environment, the human modeler cannot directly participate in evaluation, the methodology cannot be meaningfully compared to existing approaches, and no real benefits are realized.

1.6 Research Objectives

In light of these issues, the goal of this research is to present a formal framework that enables the computer to support the chemical process modeling activity at the level of chemical engineering knowledge. Such a framework will enable all chemical engineers to readily construct and use process models. With respect to this goal, three objectives are identified:

1. The development of a formalized phenomena-based modeling language,
2. The systematization of the modeling activity through modeling logic, and
3. The implementation and evaluation of the modeling language and logic through a computer-aided modeling environment.

Each of these objectives will now be discussed.

1.6.1 Development of a Formalized Phenomena-Based Modeling Language

A high-level, declarative modeling language, capable of describing chemical processes in terms of structured networks of interacting physical and chemical phenomena, must be developed and described formally. This language will allow modelers to develop models by naturally articulating

assumptions about a process, instead of writing equations. The phenomena-based representation will explicitly maintain the underlying assumptions about a process model, allowing accumulation of modeling assets, and resulting in models which are easier to create, edit, document, reuse, analyze, and understand.

The starting point for this language is the MODEL.LA modeling language described in (Stephanopoulos et al, 1990a, 1990b). However this language must be significantly modified and extended to encompass the description of dynamic processes, the representation of spatially distributed processes (whose behavior is described by partial differential equations), and the thermodynamic and physical characterization of materials in a process.

1.6.2 Systematization of Modeling Activity through Modeling Logic

The modeling language will allow chemical processes to be represented at the level of elementary physical and chemical phenomena. For the computer to comprehend and interpret these models, the underlying logic of model development must be systematized by expressing the concepts of chemical engineering science in a computational formalism. In other words, this will allow us to *teach chemical process modeling principles to the computer*. The uncovering of such modeling logic will systematize the modeling activity by formalizing modeling tasks that are currently carried out by modelers in an implicit and informal manner.

With this logic, the procedural modeling knowledge behind the development of a process model can be captured, thus allowing the computer to interactively guide model development, automatically derive mathematical model equations, explain the resulting equations in terms of the phenomena-based description, and detect model inconsistencies and incompleteness. This will enable the computer to take responsibility for much of the burden of model development, while providing interactive guidance and feedback to the modeler. As a result, the engineer can concentrate on the creative aspects of modeling, easily exploring alternatives, tracking decisions, and utilizing models in multiple contexts.

1.6.3 Implementation and Evaluation of Modeling Language and Logic through a Computer-Aided Modeling Environment

In order to provide an experimental framework to test the ideas of phenomena-based modeling language and logic, these concepts must be integrated in a computer-aided modeling environment.

This environment should address the chemical engineering modeling needs of both practicing engineering and students. For the evaluation to be meaningful, the environment should be capable of modeling nontrivial examples, including dynamic, discontinuous, spatially distributed, and hierarchical processes under a variety of contexts. Modeling assistance should be extended to all aspects of the process modeling activity, from the phenomena-based model declaration to evaluation of behavior determined from numerical solution of the derived equations. For valid comparison with other computer-aided modeling tools, the phenomena-based modeling environment should integrate state-of-the-art computer-aided modeling features, including an interactive graphical interface, incorporation of thermodynamic and physical property database information, description of process control and operational schedules, assistance for consistent specification of degrees of freedom and initial conditions for mathematical models, solution of the model equations using an equation-based modeling tool, and graphical display of results.

1.7 Thesis Outline

This thesis organized as follows. Chapter 2 identifies the declarative characteristics of chemical process models and procedural characteristics of process model development that a high-level computer-aided modeling tool should address. Chapter 3 describes the MODEL.LA phenomena-based modeling language, which provides the basis for development of the systematized modeling logic described in Chapter 4. Chapters 5 and 6 describe the functionality, structure, and design of the MODEL.LA computer-aided modeling environment, which integrates the formalized modeling language and logic. Chapter 7 illustrates several modeling examples that utilize MODEL.LA, including models for hierarchical process design, dynamic processes, and spatially distributed processes. Finally, Chapter 8 summarizes the contributions of this work, describes the potential impact it may have on chemical process modeling in engineering practice and undergraduate education, and identifies areas for future research.

Chapter 2

Requirements for High-Level Process

Modeling

This chapter elaborates on the concept of high-level computer-aided process modeling support. The objective is to identify characteristics of process modeling that a computer-aided modeling environment, which is designed to communicate with the modeler at the level of chemical engineering knowledge, should address. First, requirements for the high-level representation of a model are identified. The modeling activity that produces such a model is then discussed, identifying several requirements for systematization of the process modeling activity. Finally, how these requirements for model representation and modeling activity systematization can be addressed by a high-level computer-aided modeling environment that incorporates a phenomena-based modeling language and logic is presented.

2.1 Requirements for the Representation of Process Models

Models are abstractions designed to predict desired aspects of the behavior of real systems. Models come in many forms. Mental models, which typically capture qualitative cause-effect relationships, are usually based on intuition and experience. Physical models, which capture physical relationships between structures, are often constructed as reduced-scale versions of real systems. Mathematical models, which quantitatively express mathematical relationships between variables of interest that are meant to represent the behavior of real systems, are the most common basis for chemical process engineering modeling activities. However, chemical engineers do not perceive of processes in terms of the equations of these mathematical models. Rather, it is

natural for engineers to consider processes in terms of physical and chemical concepts, such as structure, materials, and relevant physicochemical phenomena. As discussed in the previous chapter, a high-level modeling approach is needed that lifts the representation of models from the level of mathematical equations to the level of physical and chemical phenomenological concepts.

Equation-based model representations can only capture and represent mathematical information about a process. A high-level process model representation should extend far beyond this, capturing and representing chemical engineering knowledge about a process. Such a representation would facilitate all aspects of the modeling activity, including model development, documentation, analysis, editing and reuse. Several requirements for the design of such a representation are now posed:

1. The high-level chemical process model representation should be fully declarative,
2. The model representation should be rooted in the principles of chemical engineering science,
3. The engineering assumptions behind a model should be explicitly captured and linked to the equations and terms of the resulting mathematical model,
4. Hierarchical structuring should enable the construction and analysis of a model at multiple levels of detail, and
5. The model representation should support mathematical model derivation under multiple contexts.

In the remainder of this section, each of these requirements will be discussed.

2.1.1 Declarative Model Representation

The representation of chemical process models should be fully declarative. It should not be a procedural language for dictating instructions to a computer. Rather, it should allow a modeler to naturally articulate assumptions about a chemical process. Declarative knowledge about a model must be kept distinct from procedural knowledge, such as how a model is used in simulation experiments or how model equations are derived or solved. This decoupling allows the model representation to be developed independently of the intended process engineering application. It also prevents the engineering assumptions behind a model from being obscured by details of its implementation. While procedural modeling knowledge is an important part of the modeling

activity, it should be treated independently from the model representation so that it may be applied generically in different modeling contexts.

2.1.2 Chemical Engineering Science Basis of Models

Chemical engineering science seeks to characterize the behavior of complex processes in terms of more readily understood elementary physical and chemical phenomena. These physical and chemical phenomena are quantified mechanistically, allowing the generation of mathematical models that relate variables of interest to other known or predictable quantities. Thus, a high-level representation of process models should be firmly rooted in the principles of chemical engineering science. It should be capable of capturing all types of physicochemical phenomena and mechanistic characterizations, including conservation relationships, phase and reaction equilibria, transport mechanisms, reaction kinetics, and thermodynamic and physical property models. The representation must also be readily extendible to capture new characterizations of phenomena and process modeling concepts. By grounding the model representation in terms of the concepts of chemical engineering science, it will be possible to automatically generate mathematical models from the high-level representation based on chemical engineering first principles.

2.1.3 Explicit Documentation of Assumptions

A representation of a process model is by definition a simplified description of a real system. As such, it cannot serve as a valid model of the system under all conditions. Rather its applicability is fundamentally limited by the range of validity of its underlying assumptions and simplifications. The appropriate level of detail of a model, its relevant phenomena and mechanistic characterizations, and its thermodynamic and physical property models are all dependent on the original context under which the model was constructed. Extrapolation of a model beyond its valid range renders the behavior it predicts meaningless. A mathematical model cannot be viewed independently of its assumptions. Therefore, the underlying assumptions and simplifications must be explicitly maintained by a model representation.

Furthermore, once a mathematical model is derived from a high-level representation, the relationship between the assumptions behind a model and the equations and terms of the resulting mathematical model should be retained. Otherwise, the linkage between the observed process

behavior and the underlying assumptions would be lost. This would inhibit the evaluation and critique of modeling assumptions, which are necessary when validating a model.

A representation that defines models in terms of explicit assumptions is much easier to understand than equation-based models. It allows modelers of varying areas of expertise to easily examine a model, analyze its behavior, and consider its assumptions without having to infer this information from equations. Models that are easily understood can be reused much more readily. The applicability of a model for use in a different context can easily and reliably be ascertained by examining its assumptions. Furthermore, if modification is needed, this may be accomplished by manipulating the model at the same high level at which it was initially constructed.

2.1.4 Hierarchical Nature of Models

A model may be examined and utilized at varying levels of detail. A model representation should explicitly reflect this hierarchical nature, allowing multiple coexisting levels of abstraction for a single model. The hierarchical structure of a process model is defined by declaring how abstract systems are conceptually decomposed into more refined subsystems. These subsystems in turn may be recursively broken down into more refined subsystems. Hierarchical modeling allows the modeler to concentrate on certain aspects of a model while abstracting others; to increase modularity and control the complexity of a model by aggregating related units into more abstract units; to generate models at multiple resolutions depending on the level of detail required for a particular application; and in the case of process design, to incrementally develop a process model where the behavior of a process model at a given level dictates the refinements at a subsequent, more detailed, level (Douglas, 1985, 1988).

2.1.5 Contextual Nature of Models

A high-level model representation should allow a model to be reused in different contexts with minimal modification required by the modeler. It should support the generation of different types of mathematical models: dynamic or steady-state conditions, intensive or extensive state characterizations, lumped or spatially distributed properties, detailed or abstract levels of detail. Therefore, the context under which model equations are derived should be considered and defined independently of the high-level representation.

2.2 Requirements for Systematization of the Process Modeling Activity

Various modeling “methodologies” have been proposed as guides to the modeling activity. Typically, these methodologies are presented in textbooks as flowcharts that provide generic templates of major tasks that a modeler tackles during model development. However, these methodologies are restricted in the guidance they can offer because they are not based on a formal representation of process models. Therefore, no explicit systematization of the modeling activity can be presented. Rather, modeling techniques can only be inferred from a sequence of examples.

A formalized high-level model representation is necessary to systematize the modeling activity. This will allow tasks that are currently carried out by expert modelers in an implicit and informal manner to be characterized explicitly. Several requirements for the characterization of these tasks, which provide the basis for systematization of the modeling activity, are now posed:

1. Systematization of the modeling activity must explicitly capture the procedural knowledge of model development,
2. The contextual nature of modeling, which is driven by the objectives and requirements of an engineering problem, should be captured,
3. Systematization should reflect that modeling is both a science and an art, and
4. The modeling framework should record decisions made during the modeling activity.

In the remainder of this section, each of these requirements will be discussed.

2.2.1 Procedural Nature of Modeling Activity

The high-level model representation, discussed in the previous section, captures the *declarative* “what-is” knowledge regarding the description of a process model. Systematization of the modeling activity, however, must capture the *procedural* “how-to” knowledge about the process of model development. The modeling activity may be decomposed into a sequence of hierarchical tasks. The objectives of these tasks and the steps taken to complete them can be expressed in terms of operators. These operators may then be used to construct the high-level model descriptions, to verify the completeness and consistency of this description, and to derive the mathematical model from the high-level model description. Each of these three aspects is now discussed:

- Model Construction: During construction of a high-level model representation, decisions must be made regarding the structure, the characterization of materials, and the mechanistic characterizations of physicochemical phenomena assumed to occur in a process. Each of these decisions may be reflected by an operator that changes the state of the model in an evolutionary manner by adding detail as additional assumptions are specified. Such operators may be characterized explicitly by their purpose, preconditions (describing conditions which must be met before an operator is initiated), and suboperations (which are the steps taken in completing a task). Systematization of model construction through these operators allows the modeler to consider computer-aided process model development as an interactive sequence of engineering decisions, rather than as the unassisted composition of a textual input file to a language compiler.
- Model Consistency and Completeness: As a model grows in complexity, even straightforward logic checks for consistency or completeness become tedious and readily overlooked. Therefore, a systematized methodology to detect these circumstances is needed. The high-level model representation allows the computer-aided analysis of models to offer more than solely mathematical model analysis techniques. Through explicit knowledge of the assumptions behind a model, operators can be formulated to detect logical errors in a model, such as hierarchical and topological structural inconsistencies and the misallocation of chemical species and reactions in a process. Model incompleteness can also be discovered by defining operators that detect missing assumptions. Finally chemical engineering guidelines and heuristics (e.g., Felder and Rousseau (1986) state that the ideal gas law should yield an error of 1% or less if the molar volume is greater than 5 l/mol for diatomic gases and 20 l/mol for other gases) can be incorporated to detect possible errors in mechanistic characterizations. These operators can be formulated as rules, characterized by a set of preconditions (describing model conditions that activate an operator) and a set of postconditions (describing model conditions that must be true upon completion of an operator). If the postconditions for an activated operator are not valid, a model inconsistency or incompleteness is detected.
- Mathematical Model Derivation: Mathematical model derivation can be viewed as a set of operators applied to the high-level model description to generate the requisite model equations. These operators are fully based on the principles of chemical engineering science.

They contain knowledge of how to construct relationships that express mass and energy conservation, thermodynamic and reaction equilibria, physical and thermodynamic property models, transport mechanism and reaction kinetics rate laws, and other constitutive relationships, based on the assumptions by the modeler. These operators can be also be considered hierarchically. For example, an operator that constructs a chemical species conservation relationship will consist of suboperations that construct the individual terms of the balance equation, which includes an accumulation term, boundary input and output flux terms, and internal consumption and generation terms. The operators that construct the mathematical relationships from the high-level model descriptions allow equation-based modeling from first principles to become a systematized process based on sound engineering principles.

2.2.2 Contextual Nature of Modeling Activity

Modeling is the essential activity that characterizes modern process engineering. However, it is important to recognize that modeling is always a contextual activity. The extent and detail of a required model are closely related to the scope and objectives of the particular process engineering problem being addressed. Context-free modeling not only risks the development of oversimplified models which may be inadequate to fill the needs of a particular application, but also risks wasting engineering effort and resources to generate overdetailed models which may be difficult to solve or require information which is irrelevant to the problem at hand. This context-dependent nature of modeling makes it impossible to provide a generic systematic modeling template for all engineering problems. However, many specific engineering problems (e.g., the hierarchical design of continuous processes) are associated with a wealth of generic guidelines and heuristics for model development. A systematic framework for model development should be capable of incorporating such knowledge, through extension of the model construction operators described above. Such contextual operators can guide the modeling activity based on the modeling objectives by presenting available alternatives and tracking decisions made. In this way, process model development can be linked to the context and objectives of the problem being addressed.

2.2.3 Science and Art of Modeling

Due to the absence of structured logic and formalized procedures to guide the arduous task of modeling from first principles, process modelers must rely largely upon not only chemical engineering knowledge, but also intuition, insight, and experience. As a result, modeling is typically regarded by both student and practicing engineers as an art, not a science. The entire modeling process cannot be formalized because the route of formulation for any non-trivial model is always an ambiguous path, determined by the goals, preferences, and perspective of the individual modeler. Modeling creativity would be greatly restricted if some rigid structure were to be artificially imposed over the entire modeling process. However, chemical engineering provides many scientific principles that allow several procedural aspects of the modeling activity to be systematized to some degree. Various degrees of systematization relevant to the modeling activity can be distinguished, each of which provides different levels of assistance and guidance to the modeler. These degrees of systematization include automation, sequencing, and organization.

Certain modeling tasks can be completely automated. In such a case, due to a set of predefined conditions or inputs by the modeler, some aspect of model development may proceed unambiguously with any further input. The most typical cases of automation will pertain to the symbolic generation, manipulation, and numerical solution of the mathematical modeling equations. Another important task for automation is in model abstraction, where a set of modeled systems are aggregated into one. Since this process is a many-to-one mapping, the structure and relationships which define the more abstract object can automatically be generated.

Other modeling tasks can be sequenced according to their respective preconditions and postconditions. For example, it does not make sense to define the physicochemical interactions (e.g., diffusion is occurring) or materials (e.g. water is present) of a model until the respective system boundaries have been identified. In another scenario, a decision made by the modeler may unambiguously determine that other decisions are required. For example, if the modeler indicates that a reaction is occurring in a particular system, an interaction may then initiated to determine the reaction medium, the species present, the relevant reactions and their stoichiometry, the kinetics, etc.

While many modeling tasks can be neither automated nor sequenced, most can be still be structured to some degree by organizing them into orthogonal sets. In this case, sequencing does

not apply because a task in one orthogonal set can be pursued independently of whether or not tasks in other orthogonal sets have been completed. This type of organization allows the user to focus on one modeling aspect at a time. For example, after the modeler defines some of the systems composing a model, (s)he may proceed with the refinement of topological structure (by adding additional systems or by indicating convective, diffusive, and energy interactions), concentrate on the individual systems (by characterizing materials present, reactions occurring, internal structure, etc.), or work on these tasks in parallel. Such organization will allow the modeler (or modelers) to easily and quickly review, add to, and/or edit the results of related modeling tasks.

By incorporating these three aspects of systematization, the process of modeling will be greatly facilitated. Automation will benefit all modelers by taking care of mundane, straightforward, and repetitive tasks during the modeling process. Sequencing will be most beneficial to inexperienced modelers (e.g., students) who may have many facts or assumptions in mind about a particular process they wish to model, but have no idea where to begin the model or where more information is required. Finally, organization will benefit all modelers not only by focusing the tasks of model definition, but also by directing the tasks of model editing and analysis by grouping similar aspects of a model.

2.2.4 Documentation of Modeling Activity

During the modeling activity, the decisions made by the modeler can be captured by the sequence of model operators activated during model development. These operators, documented along with the objectives of the engineering problem and assumptions made, provide an explicit record of the modeling activity. The purpose of this is threefold. First, this allows the modeler to revisit intermediate steps of the modeling process and define alternative (but possibly coexisting) contexts. Second, this record provides a method to trace, repeat, and debug the modeling process. For example, in an educational setting both the student and the instructor can analyze and critique how a model was created by studying the assumptions and decisions made during model formulation. Third, given a record of several previous modeling attempts, a foundation is established for automating a critique of modeling activities, distinguishing patterns between similar modeling activities, and identifying analogies from past modeling efforts for application to

new ones.

2.3 Implementation of High-Level Computer-Aided Modeling Support

In this preceding sections of this chapter, the requirements for a high-level representation of chemical process models and the systematization of the modeling activity that produces these models was discussed. In this section, an overview of the modeling language and logical framework for addressing these requirements in a computer-aided modeling environment is presented.

2.3.1 Phenomena-Based Modeling Language

Given the requirements for a high-level process model representation, there are many possible designs for its implementation in a computer-aided modeling environment. However, the most important consideration is that this implementation is designed primarily with the needs of the human modeler in mind. A computer can capture the high-level representation with any type of internal data structure. However, from the perspective of the human, the most intuitive means of communication and documentation of a model is natural language. This would offer the lowest learning curve, allowing the human to naturally express assumptions about a given process model with maximum flexibility. However, for practical purposes of human-computer intercommunication, natural language is ambiguous and redundant, offering limitless ways of expressing the same concepts. Therefore the high-level model representation will be a well-defined subset of natural language, where each element of the language has a direct and explicit impact on the resulting model.

In addition to the requirements for a high-level model representation already discussed, the modeling language will be designed to meet the following requirements:

1. The modeling language will allow assumptions about a model to be articulated and documented through linguistic (or “English-like”) declarations regarding the structure of the modeled systems, the characterization of materials present in these systems, and the relevant physicochemical phenomena occurring within and among the modeled systems.
2. The vocabulary of the language will encompass a library of concepts from chemical engineering science, including conservation principles, equilibria, reaction kinetics,

transport mechanisms, and thermodynamic and physical property models, etc. The language will also be readily extendible to incorporate additional modeling concepts.

3. The modeling language will be composed of a set of modeling elements and semantic relationships. Model elements represent chemical engineering concepts (e.g., systems, fluxes, reactions, species, materials, etc.). Semantic relationships unambiguously describe how these modeling elements are interrelated in forming an instance of a particular model. For example, in a model fragment such as:

REACTOR_Y *has-convective-output* REACTOR_EFFLUENT

REACTOR_Y and REACTOR_EFFLUENT are modeling elements representing a system and a flux, respectively, and *has-convective-output* is a semantic relationship identifying that REACTOR_EFFLUENT is a convective flux that transports material from the system REACTOR_Y.

4. The syntax of the modeling language will be defined formally using computational language representations from computer science.
5. The definition of the language will be independent of the computer-aided environment in which it is implemented.

The high-level representation provided by the modeling language will be complemented by the systematization of the modeling activity made possible by modeling logic.

2.3.2 Modeling Logic

Given the requirements for systematization for the modeling activity, it is important to clearly identify the procedural modeling knowledge necessary for model development. The details of how this knowledge is used and implemented in a computer-aided modeling environment should be treated separately.

The use of logic in computer science has its roots in the field of artificial intelligence, specifically for the purpose of automated theorem-proving using propositional assertions. With the advent of logic programming, its use expanded to capture both declarative and procedural knowledge. It has been proposed (Kowalski, 1979) that

LOGIC + CONTROL = ALGORITHM

where

1. LOGIC identifies the knowledge required to solve a problem,
2. CONTROL specifies the way the knowledge is used to solve the problem, and
3. ALGORITHM results from the combination of LOGIC and CONTROL, yielding an algorithm, or computer program, suitable for practical use.

Thus, modeling logic is proposed as the appropriate framework for providing a concise, modular, extendible framework to capture chemical engineering modeling knowledge. This logical framework will be defined independently of the details of its implementation. In addition to the requirements for systematization of the modeling activity, the modeling logic will be designed to meet the following requirements:

1. The modeling logic will consist of a set of logical operators, which may be interpreted as “if-then” statements of knowledge, where “if” certain conditions are true, “then” certain actions are taken or certain conclusions are made.
2. The modeling logic operators will be defined in terms of the modeling elements and semantic relationships of the modeling language.
3. The modeling logic operators will be designed to cover the major tasks encountered during model development, including high-level model construction, derivation of a mathematical model from the high-level model description, explanation of the terms and equations of the resulting mathematical model in terms of the underlying language-based assumptions, and detection of model inconsistencies and incompleteness.
4. The modeling logic will be readily extendible to encompass context-dependent modeling knowledge, applicable to specific types of process engineering problems, which can serve as a guide to the modeling activity.

By combining the aspects of a high-level modeling language and systematized modeling logic, it will be possible to support the modeling process at the level of chemical engineering knowledge. These ideas will be evaluated by implementing them in a computer-aided modeling environment.

2.3.3 Computer-Aided Modeling Environment

In order to provide an experimental apparatus for testing the ambitious goals of phenomena-based

modeling, the modeling language and logical framework will be integrated in a computer-aided modeling environment. Computer-aided process modeling may be viewed as an interactive dialogue between a human engineer and a computer. The phenomena-based language will provide the vocabulary of discourse that makes this possible. The language provides explicit means for communication, analysis, and documentation of the modeling assumptions.

Although the elements of the language are rooted in the principles of chemical engineering, it is the modeling logic that enables the computer to understand chemical engineering modeling concepts and, through a graphical user interface, to become a true modeling assistant by:

1. Interpreting the meaning of the high-level phenomena-based description, automatically deriving the requisite model equations based on the context specified by the modeler, and explaining the resulting equations in terms of the model assumptions,
2. Assisting hierarchical modeling by carrying over assumptions from one model level to another and maintaining consistency among all levels,
3. Increasing modeling efficiency by assuming responsibility for the straightforward, repetitive, or mundane aspects of modeling,
4. Formalizing and structuring many modeling tasks which are presently carried out by modelers in an informal and implicit manner,
5. Facilitating model editing and reuse by explicitly documenting modeling assumptions and decisions,
6. Providing interactive guidance and feedback by detecting modeling inconsistencies and incompleteness, and
7. Using context-dependent engineering knowledge to guide the modeling activity based on the objectives of a given engineering problem.

Thus, by providing a set of universal modeling language elements capable of describing practically any process at any level of detail, and logical methods for selecting, instantiating and combining these language elements through declarative phenomena-based assumptions, modeling from first principles can be supported with an ease of use that far exceeds that of existing computer-aided modeling approaches.

While the modeling language and logic will be the foundation of the modeling environment, its scope will be expanded through interfaces with external software components so that a valid comparison may be made with existing computer-aided modeling tools through application to a wide range of real engineering problems. The environment will integrate many state of the art computer-aided modeling features. Relational databases will be accessed for the physical and thermodynamic property data of chemical species. Software for the calculation of properties of mixtures will be incorporated. Specification of arbitrary control structures and operational schedules will be included. Structural algorithms will be used for interactive specification of consistent sets of design variables and initial conditions. Equation-based modeling tools will provide numerical routines to solve the model equations, and plotting software will display results in tabular and graphical form. Capabilities for the access of other external software, such as programs for equipment sizing and costing, will also be provided.

In summary, by enabling the computer to act as a knowledgeable assistant to the modeler, phenomena-based modeling can transform much of equation-based process modeling from an abstract art-form into a well-defined science. High-level modeling activities allow the rapid creation and investigation of models of unique and novel processes. The computerized generation and solution of model equations increases efficiency and reduces the risk of error. Since all assumptions can be explicitly documented and the task of model formulation entirely decoupled from computation procedures, the model may be readily applied for use in a variety of contexts. Through explicit knowledge of all assumptions pertaining to a particular model, the computer will be able to offer guidance and assistance during model development and modification, while continuously checking for logical consistency. Finally, through an integrated modeling environment, computer-aided assistance can be extended to all aspects of modeling, from the declaration and documentation of the modeling objectives and assumptions, formulation of the mathematical model, specification and solution of the model, and interpretation of results.

Chapter 3

Modeling Language Framework

A model is an artifact that provides an experimental framework for inferring some aspect of the behavior of a particular system of interest. For this purpose, engineers have traditionally employed mathematical models as the basis of process modeling. These mathematical models are expressed using the “language” of mathematical equations. Subsequent numerical solution of the resulting set of algebraic, differential, partial differential, and/or integral equations requires translation of the mathematical model into a computational model expressed using some procedural programming language.

As illustrated in Figure 3-1, a model must evolve through a series of representations in order to close the gap that exists between a real process and a valid computational model of that process. At the start of the modeling activity, an engineer uses natural language to express his or her interpretation of the physical and chemical phenomena that characterize a process. While natural language provides a high-level intuitive means for the engineer to express his or her interpretation of a physical process, it is not a computationally formalized language (in terms of syntax and semantics). This lack of formalization makes the meaning of the model ambiguous, as only the original modeler can know the exact intended meaning behind its description. Without a formal model description, translation into the next modeling language level, mathematical equations, cannot be automated nor even rigorously documented. Lack of such formalization attributes to chemical process modeling being viewed as an art, rather than as a science.

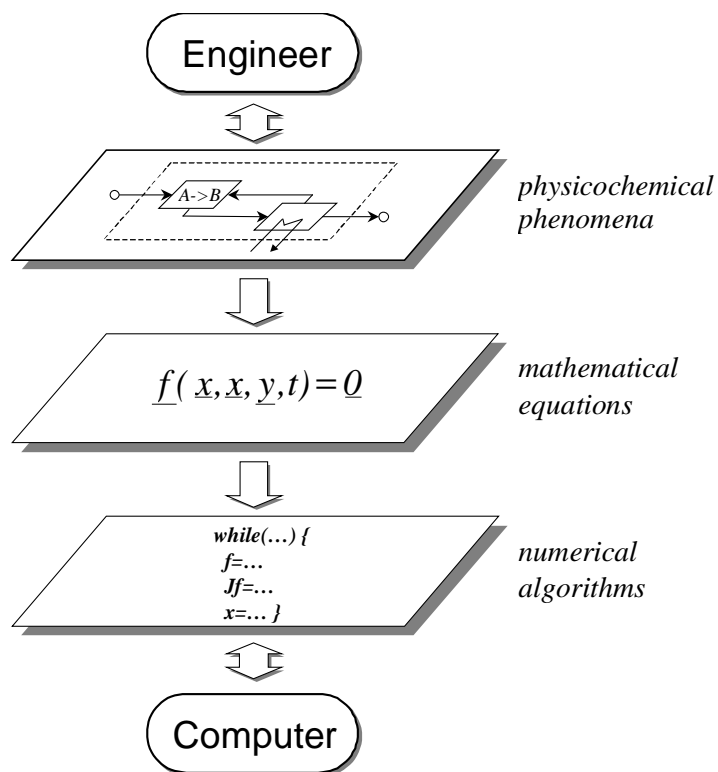


Figure 3-1: Evolution of Process Model Representations

In this chapter, a natural characterization of chemical process models leads to the identification of a set of fundamental modeling elements. Each modeling element may be interpreted as structured piece of modeling knowledge that captures a set of related assumptions. These elements provide the building blocks of a high-level phenomena-based process modeling language, named MODEL.LA, for describing a physicochemical interpretation of a chemical process. This high-level language provides means for rigorous documentation of chemical process modeling assumptions and automated mathematical model generation. In addition to modeling elements, semantic relationships are introduced in MODEL.LA to unambiguously describe how the modeling elements are interrelated in forming a particular instance of a process model. This allows the phenomena-based model to be represented as a semantic network. This representation organizes the knowledge behind the model as a directed graph and provides a structured and modular means of analyzing it.

3.1 Formal Modeling Language Representation

The definition of any language must include a set of specifications that describe its syntax and its

semantics. In order to formally define the syntax of the MODEL.LA modeling language, a context-free grammar (Sipser, 1998) is used. A context-free grammar is a 4-tuple (V, Σ, R, S) , where

1. V is a finite set called the *variables*,
2. Σ is a finite set, disjoint from V , called the *terminals*,
3. R is a finite set of *production rules*, with each rule being a variable and a string of variables and terminals, and
4. S is the start symbol.

Context-free grammars are commonly used for representing the syntax of programming languages and also several examples of subsets of natural languages. Furthermore, in this work, the production rules of the context-free grammar are used to structure the modeling assumptions hierarchically and capture the necessary and alternative decisions made during model development.

In the definition of the context-free grammar of MODEL.LA, variables appear as bracketed strings. For example, the string $\langle abc \rangle$ is a variable. Terminals appear as Ariel font text strings. For example, xyz is a terminal. Production rules appear as a variable and a string of variables and terminals, separated by the arrow symbol \rightarrow . For example,

$$\langle abc \rangle \rightarrow \langle def \rangle xyz$$

is a rule that produces $\langle def \rangle xyz$ by substitution with $\langle abc \rangle$. The complete set of all strings that may be produced by such substitutions, beginning with the start symbol, comprise the *language* of the grammar. Alternative substitutions for the same variable are written as a single rule, with each substitution separated by the pipe symbol $|$. For example,

$$\langle def \rangle \rightarrow xyz | \langle def \rangle uvw |$$

is a rule where variable $\langle def \rangle$ may be substituted for one of three strings: xyz , $\langle def \rangle uvw$, or the empty string. Note that the second substitution is recursive. This property allows the production of an infinite number of strings (i.e., the language of the grammar is infinite).

Production rules may be depicted hierarchically using a graphical tree-like structure, which

will be referred to as a *production tree*. For example, Figure 3-2 illustrates the production tree for the two rules described above.

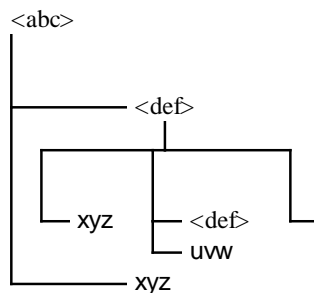


Figure 3-2: Example Production Tree

In a production tree, the variable on the left-hand side of the first production rule of interest appears at the top root node of the tree (e.g., $\langle abc \rangle$ in Figure 3-2). Recursively, the variables and terminals introduced on the right-hand side of each production rule appear as horizontal branches from the variable on the left-hand side of the rule (e.g., branches $\langle def \rangle$ and xyz from $\langle abc \rangle$ in Figure 3-2). For production rules with alternative substitutions, each set of possible substituted variables and terminals on the right hand side of the rule appear on separate vertical branches from the variable on the left-hand side of the rule (e.g., the three vertical branches from the first occurrence of $\langle def \rangle$, introducing xyz , $\langle def \rangle uvw$, or the empty string, respectively, in Figure 3-2).

The sequence of substitutions used to obtain a string in the language is called a *derivation*. In this work, the production rules are designed to correspond to structured sets of modeling assumptions (e.g., *structural characterization*) introduced during model development. In this way, derivations in the modeling language provide a formal record of the modeling assumptions that produce a model. The production rule representation introduces a hierarchical nature to these assumptions (e.g., *structural characterization* requires declaration of *system boundaries* and *boundary fluxes*). Furthermore, by capturing the rationale for each substitution, additional knowledge regarding the purpose, applicability, and context of the model can be retained explicitly in the modeling framework.

3.2 Hierarchy of Model Equations

The fundamental bases of chemical process models are the conservation principles. Conservation relationships for mass, energy, and momentum for each system of interest may be expressed

mathematically using balance equations. The general balance equation for any of the conserved quantities, represented as B , in a system may be expressed generically as:

$$(\textit{accumulation})_B = (\textit{flux in})_B - (\textit{flux out})_B + (\textit{source})_B$$

where:

1. $(\textit{accumulation})_B$ is the rate of accumulation of quantity B within the system boundaries,
2. $(\textit{flux in})_B$ is the total flux of B entering through the system boundaries by all modes of transport,
3. $(\textit{flux out})_B$ is the total flux of B leaving through the system boundaries by all modes of transport, and
4. $(\textit{source})_B$ is the net rate of B generated or consumed by all modes of phenomena within the system boundaries.

The terms of a balance equation are dependant on the system boundaries, or control volume, selected and the physicochemical phenomena assumed to occur within and across the system boundaries. Selection of an appropriate control volume is determined by level of detail required by the context of a particular engineering problem. Models may also be examined at multiple levels of detail, where an abstract control volume subsumes other more refined control volumes. However, a model viewed at any level of detail must be consistent with models at more abstract or more refined levels. For example, the net rate of accumulation of any conserved quantity in an abstract system must equal the aggregate sum of the accumulation of the conserved quantity over each of the system's subunits.

The balance equations are supplemented by constitutive equations that result from the mechanistic characterization of assumed physicochemical phenomena. These constitutive equations include reaction kinetics and transport rate expressions, thermodynamic and physical property relationships, empirical correlations, etc. The modeler may further supplement the mathematical model equations by declaring additional relationships expressing design constraints, controller relationships, and other external influences affecting the process.

3.3 Phenomena-Based Model Characterization

The informal description of chemical process modeling equations in the previous section identifies several dimensions of process characterization necessary for model development. To formalize these concepts, the first production rule of the context-free grammar of the MODEL.LA modeling language is introduced:

$$\langle \text{phenomena-based model} \rangle \rightarrow \langle \text{structural characterization} \rangle \langle \text{chemical characterization} \rangle \langle \text{derivation context} \rangle$$

This production rule may also be represented by the production tree depicted in Figure 3-3.

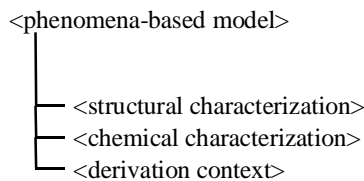


Figure 3-3: Phenomena-Based Model Production Tree

The variable $\langle \text{phenomena-based model} \rangle$ introduces the start symbol of the grammar, and is the root of all derivations in the language. The form of this production is motivated by the conservation principles, which depend on a phenomena-based mechanistic characterization of the process. The assumptions behind a phenomena-based model are organized into three aspects: characterization of process structure (represented by the variable $\langle \text{structural characterization} \rangle$), characterization of the chemical content of the process (represented by the variable $\langle \text{chemical characterization} \rangle$), and declaration of a modeling context (represented by the variable $\langle \text{derivation context} \rangle$) under which the mathematical model is to be derived. The three substitution variables identify these three primary aspects of a phenomena-based process model and can be regarded independently. The $\langle \text{structural characterization} \rangle$ provides a topological and hierarchical structural template for the $\langle \text{phenomena-based model} \rangle$. It identifies control volumes, for which balance equations will be written, and how these control volumes interact, providing generic $(flux\ in)_B$ and $(flux\ out)_B$ terms in the balance equations. As an example, a simple interpretation of a $\langle \text{structural characterization} \rangle$ may be a process flowsheet. The $\langle \text{chemical characterization} \rangle$ provides additional details by specifying the chemical species, reactions, and materials that are present in the process. Species information identifies the number of chemical species balance equations required for each control volume, and reaction

information identifies the $(source)_B$ terms in these equations. Information on materials in the process adds specifications on their physical and thermodynamic properties. Finally, the $\langle derivation\ context \rangle$ specifies the context under which the mathematical model is derived. For example, an assumption of steady-state or dynamic conditions determines the form of the $(accumulation)_B$ terms in the balance equations.

The independence of the three facets of the $\langle phenomena\ based\ model \rangle$ provides powerful means for model reuse. For a given $\langle structural\ characterization \rangle$ and $\langle chemical\ characterization \rangle$, different mathematical models may be derived for each independent $\langle derivation\ context \rangle$. Furthermore, the $\langle structural\ characterization \rangle$ provides a model template which is independent of the $\langle chemical\ characterization \rangle$. As an example, this allows the $\langle structural\ characterization \rangle$ of a distillation column to be developed and used for multiple $\langle chemical\ characterization \rangle$'s (i.e., distillation of any mixture of chemical species).

3.3.1 Structural Characterization

The characterization of the structure of a phenomena-based model, represented by the variable $\langle structural\ characterization \rangle$, encompasses both topological structure and hierarchical structure. It is captured by the declaration of instances of modeling elements *modeled-units* and *fluxes*. This declaration is expressed formally by the production:

$$\langle structural\ characterization \rangle \rightarrow \langle modeled-units \rangle \langle fluxes \rangle$$

Variables $\langle modeled-units \rangle$ and $\langle fluxes \rangle$ represent lists of element type $\langle modeled-unit \rangle$ and $\langle flux \rangle$, respectively:

$$\begin{aligned} \langle modeled-units \rangle &\rightarrow \langle modeled-units \rangle \langle modeled-unit \rangle | \langle modeled-unit \rangle \\ \langle fluxes \rangle &\rightarrow \langle fluxes \rangle \langle flux \rangle | \end{aligned}$$

The topological structure of a process model is defined by declaring the systems of interest and how they interact through the transfer of mass and energy. Each instance of a modeled-unit modeling element represents a control volume—a modeled system delimited from its environment by its boundaries. Each modeled-unit is represented in the mathematical model by a balance equation for each conserved quantity of interest (mass, energy, and chemical species).

Each flux represents the transport of material (through convective flow), energy, or

selected chemical species across the boundaries of two separate interacting modeled-units, or between a modeled-unit and the unmodeled surrounding environment. Each flux is represented in the mathematical model as a term in the balance equations of the corresponding modeled-units.

Note from the form of the above recursive productions that the list of <modeled-units> must contain at least one element, while the list of fluxes may be empty. This is because a model must contain at least one control volume for conservation relationships to be expressed, while boundary fluxes into these systems are not mandatory.

The modeled-unit also captures the hierarchical decomposition of a model as any modeled-unit may be refined into any number of more refined modeled-units. In a mathematical model, a composite modeled-unit may be viewed abstractly, where balance equations are derived as for an elementary process unit, or as an aggregate, where the extensive quantities characterizing the unit are determined by summing over the corresponding extensive quantities of its subunits.

3.3.2 Chemical Characterization

The structural characterization of the phenomena-based process model is complemented by the chemical characterization. The chemical characterization identifies instances of modeling elements representing the chemical species, reactions, material-contents, and phases assumed to be present in a process. Declaration of these elements in a phenomena-based model is represented by the production:

$$\langle \text{chemical characterization} \rangle \rightarrow \langle \text{chemical species list} \rangle \langle \text{chemical reactions} \rangle \\ \langle \text{material-contents} \rangle \langle \text{phases} \rangle$$

Variables <chemical species list>, <chemical reactions> <material-contents>, and <phases> represent lists of element type <chemical species>, <chemical reaction>, <material-content>, and <phase> respectively:

$$\langle \text{chemical species list} \rangle \rightarrow \langle \text{chemical species list} \rangle \langle \text{chemical species} \rangle | \\ \langle \text{chemical reactions} \rangle \rightarrow \langle \text{chemical reactions} \rangle \langle \text{chemical reaction} \rangle | \\ \langle \text{material-contents} \rangle \rightarrow \langle \text{material-contents} \rangle \langle \text{material-content} \rangle | \\ \langle \text{phases} \rangle \rightarrow \langle \text{phases} \rangle \langle \text{phase} \rangle |$$

The details of these elements are discussed in the following section.

3.3.3 Derivation Context

The derivation context does not introduce additional information to the physicochemical nature of the phenomena-based model. Rather, it introduces assumptions under which the mathematical model is to be derived. For example, it encompasses whether the model is assumed to be dynamic or steady state, whether the equations are derived on a per mole or per mass basis, the level of resolution desired for a hierarchical model, etc. Declaration of these assumptions is represented by the production rule for the <derivation context>:

<derivation context> → <dynamic assumption> <mole or mass basis> <level of resolution>
<intensive or extensive characterization> <energy balance inclusion>...

Note that the specification of the <derivation context> may be regarded independently of the definition of the <structural characterization> and <chemical characterization> of a phenomena-based model.

3.4 Characterization of Modeling Elements

The production rules introduced thus far are depicted in the production tree shown in Figure 3-4.

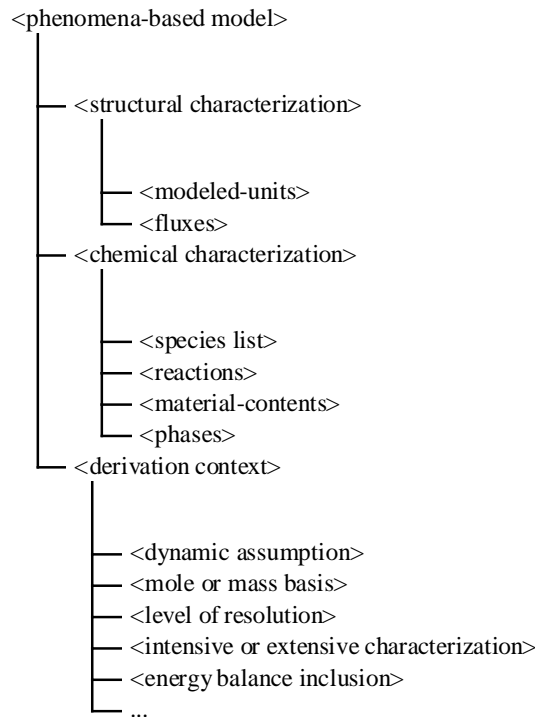


Figure 3-4: Expanded Phenomena-Based Model Production Tree

The structural and chemical characterization of the phenomena-based process model has introduced six fundamental modeling elements, represented by variables <modeled-unit>, <flux>, <chemical species>, <chemical reaction>, <material-content>, and <phase>. Instances of these elements can capture the physicochemical phenomena-based description of a limitless number of chemical processes. The information required to fully specify instances of these element types will now be discussed.

3.4.1 Modeled-Unit Characterization

The modeled-unit represents an instance of a control volume in a phenomena-based process model. Assumptions necessary to fully define an instance of a modeled-unit are represented by the production tree depicted in Figure 3-5.

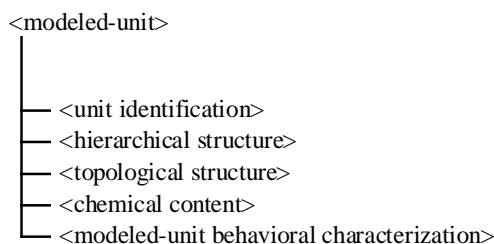


Figure 3-5: Modeled-Unit Production Tree

- **Unit Identification:** The variable <unit identification> identifies the modeling element as a modeled-unit and the unique textual name used to refer to it. This is indicated by use of the semantic relationship *is-a* introduced in the production rule for <unit identification>:

<unit identification> → [modeled-unit id] *is-a* modeled-unit

where [modeled-unit id] is a string representing the name of the modeled-unit. This results in a declaration in the phenomena-based model such as:

REACTOR_VESSEL *is-a* modeled-unit

where REACTOR_VESSEL is the name given to a particular instance of a modeled-unit.

- **Hierarchical Structure:** The variable <hierarchical characterization> represents declaration of the hierarchical structure of a modeled-unit. It identifies parent unit of the modeled-unit and its <internal characterization>. The production tree for <hierarchical characterization> is shown in

Figure 3-6.

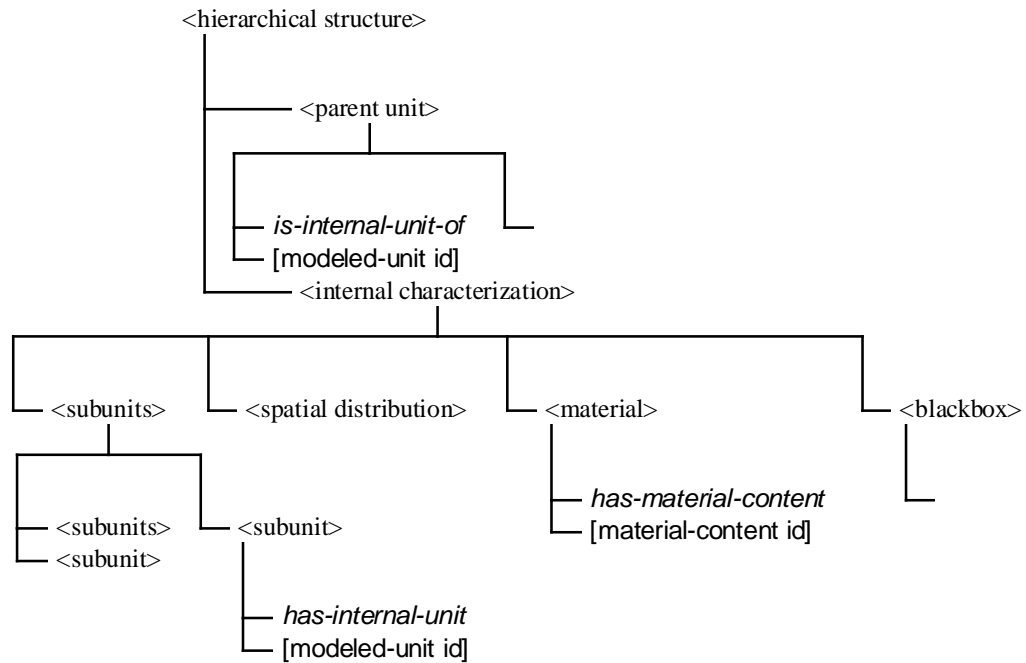


Figure 3-6: Hierarchical Structure Production Tree

The parent unit of a modeled-unit, if any, is identified by the semantic relationship *is-internal-unit-of*. The complementary association, identifying the subunit of a composite parent modeled-unit, is captured by the semantic relationship *has-internal-unit*. These associations are illustrated by the following model declarations:

```

JACKETED_CSTR    is-a modeled-unit
                 has-internal-unit JACKET
                 has-internal-unit VESSEL

JACKET           is-a modeled-unit
                 is-internal-unit-of JACKETED_CSTR

VESSEL           is-a modeled-unit
                 is-internal-unit-of JACKETED_CSTR
  
```

The modeler must decide to model the internal structure of a modeled-unit in one of several different ways. The first substitution for <internal characterization>, illustrated above by the semantic relationship *has-internal-unit*, indicates an abstract modeled-unit which is decomposed into a set of subunits. The third substitution indicates a modeled-unit which is assumed to

have a material-content. This is declared using the semantic relationship *has-material-content*, as illustrated below:

```

VESSEL      is-a modeled-unit
            has-material-content VESSEL_MATL
    
```

where VESSEL_MATL is an instance of a material-content modeling element. The fourth substitution for <internal characterization> indicates a blackbox modeled-unit, where the unit is modeled as an arbitrary point of mixing, separation, and/or reaction. Since this substitution introduces the empty string, no semantic relationship is necessary. The second substitution for <internal characterization> indicates a spatially distributed modeled-unit. A spatially distributed modeled-unit represents a process unit that is characterized internally by spatially distributed properties. It is modeled using a differential element subunit, along with boundary element subunits for each of the distributed dimensions. The balance equations for such a unit are in the form of partial differential equations (PDEs). The production tree for <spatial distribution> is illustrated in Figure 3-7.

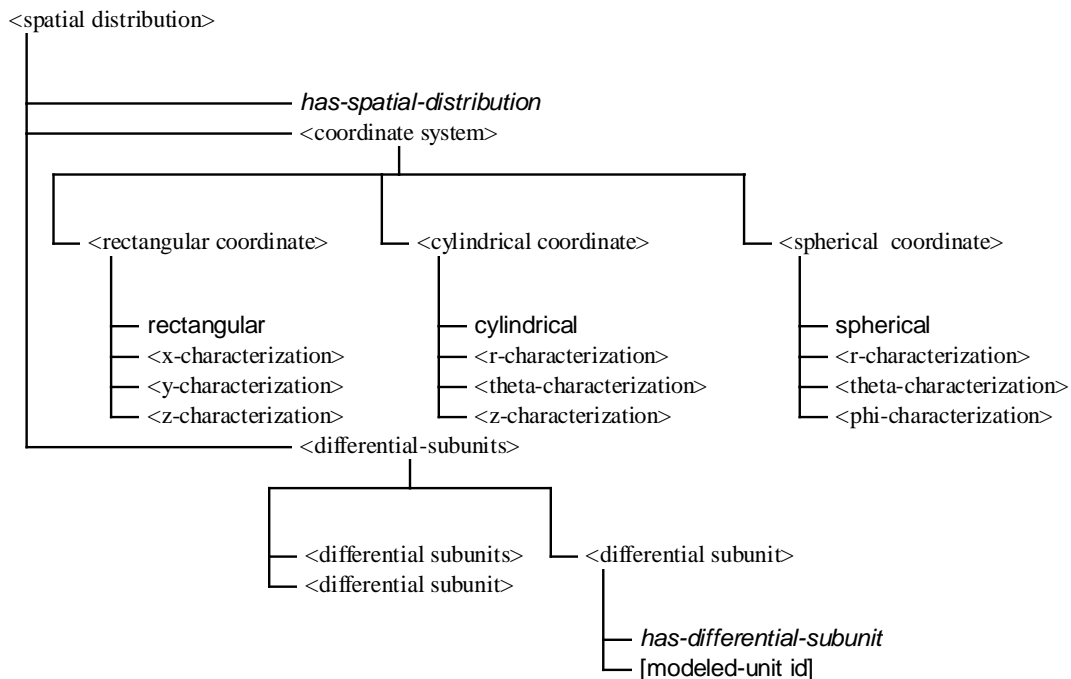


Figure 3-7: Spatial Distribution Production Tree

Declaration of a spatially distributed modeled-unit requires selection of a coordinate system,

expressed using the semantic relationship *has-spatial-distribution*, characterization of each spatial dimension of the selected coordinate system, and declaration of the differential element subunits, expressed by the semantic relationship *has-differential-subunit*. Characterization of the spatial dimensions is illustrated by the production tree for a representative variable *<x-characterization>* shown in Figure 3-8.

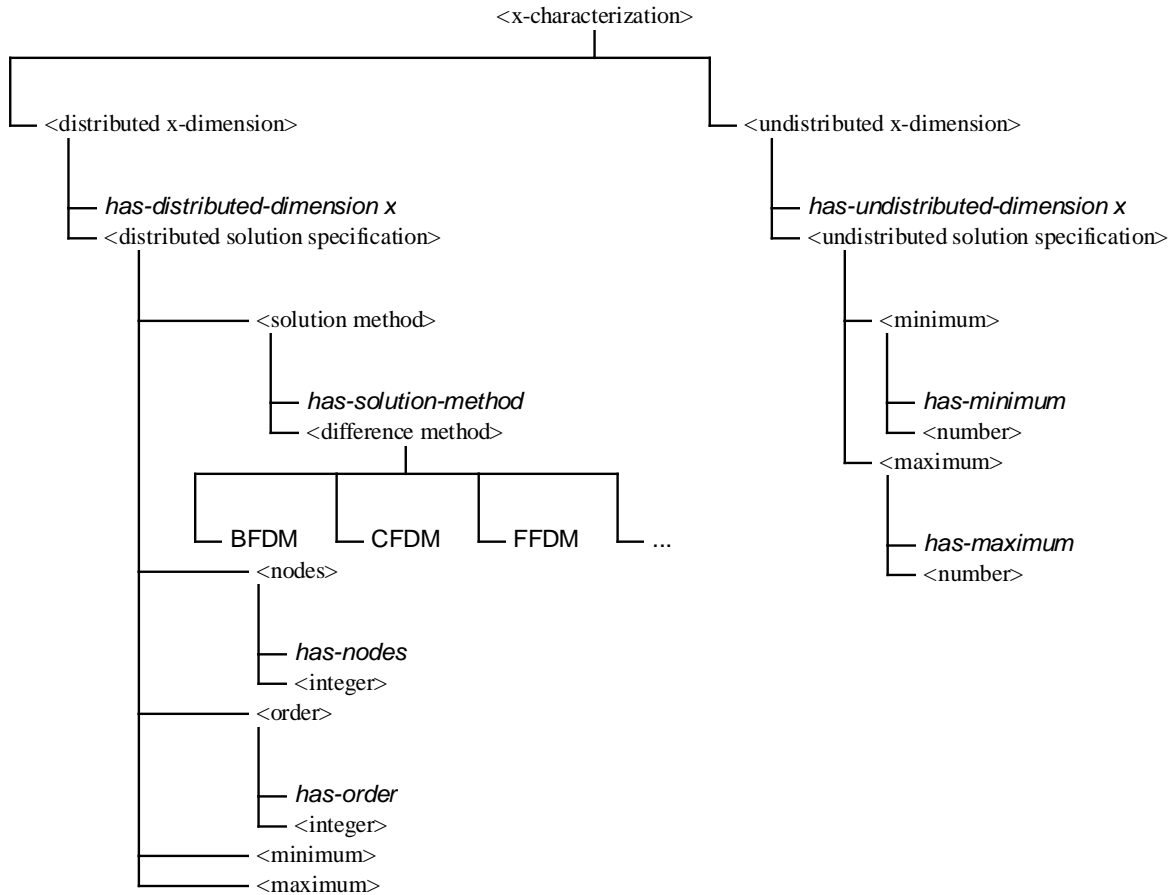


Figure 3-8: Example Spatial Dimension Production Tree

Each dimension is either assumed to be distributed, declared by semantic relationship *has-distributed-dimension*, or non-distributed, declared by semantic relationship *has-undistributed-dimension*. The declaration of a cylindrical tubular reactor with radial and axial distribution is illustrated below:

```

Tube_reactor  is-a modeled-unit
               has-spatial-distribution cylindrical
               has-distributed-dimension r
               has-undistributed-dimension theta
    
```

```

has-distributed-dimension z
has-differential-subunit Tube_reactor_rz
has-differential-subunit Tube_reactor_r1
has-differential-subunit Tube_reactor_r2
has-differential-subunit Tube_reactor_z1
has-differential-subunit Tube_reactor_z2

```

The modeled-unit `Tube_reactor_rz` represents the differential element subunit for which the partial differential equations characterizing the spatially distributed modeled-unit will be derived. The modeled-units `Tube_reactor_r1` and `Tube_reactor_r2` represent the differential element boundary subunits which will determine the radial boundary conditions, while modeled-units `Tube_reactor_z1` and `Tube_reactor_z2` will determine the axial boundary conditions.

Additional specifications regarding the particular numerical solution method used to solve the partial differential equations characterizing the spatially distributed modeled-unit may be declared as illustrated in Figure 3-8 using semantic relationships *has-solution-method*, *has-nodes*, *has-order*, *has-minimum*, and *has-maximum*. Of course, these specifications depend on the particular numerical solution method selected and do not affect the phenomena-based model description.

- **Topological Structure:** The topological structure of a modeled-unit requires declaration of all boundary inputs and outputs to other modeled-units or the surroundings. The production tree for <topological structure> is illustrated in Figure 3-9. The input and output boundary fluxes are characterized as convective, energy, or species transport, as declared by semantic relationships *has-convective-input* and *has-convective-output*, *has-energy-input* and *has-energy-output*, and *has-species-input* and *has-species-output*, respectively. For example, the declarations:

```

REACTOR_VESSEL  is-a modeled-unit
                 has-convective-input REACTOR_FEED
                 has-convective-output REACTOR_EFFLUENT
                 has-energy-output REACTOR_Q

```

illustrate a modeled-unit, `REACTOR_VESSEL`, with a convective input stream, `REACTOR_FEED`, a convective output stream, `REACTOR_EFFLUENT`, and an energy output

flow, REACTOR_Q.

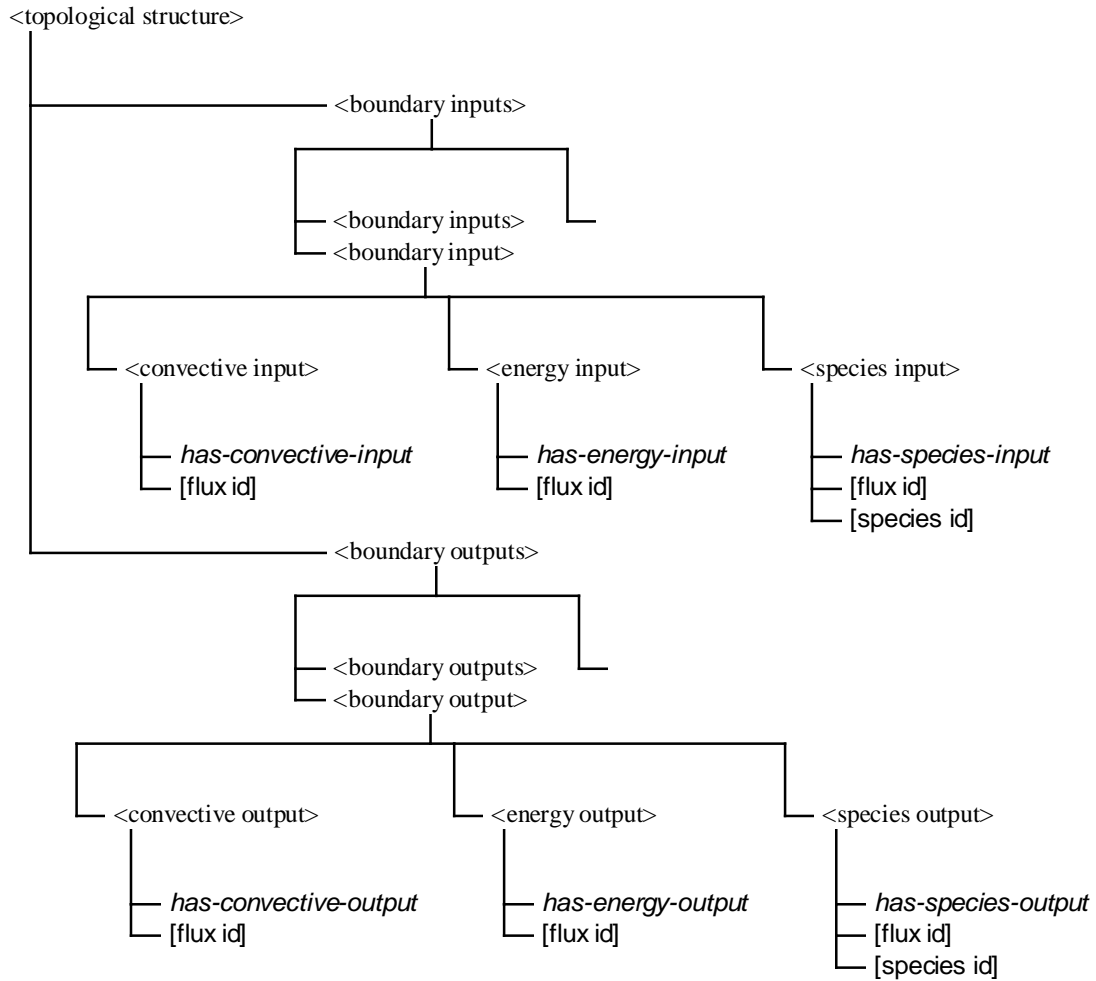


Figure 3-9: Topological Structure Production Tree

- Chemical Content: The chemical content characterization of the modeled-unit involves selection of all chemical species and reactions assumed to occur internally. The production tree for <chemical content> is illustrated in Figure 3-10.

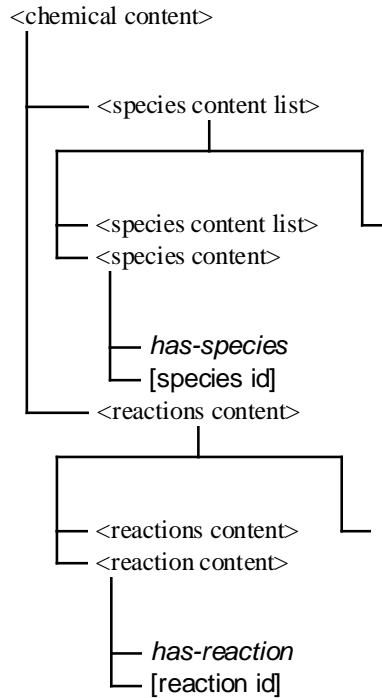


Figure 3-10: Chemical Content Production Tree

Chemical species and reactions assumed for a modeled-unit are declared using the semantic relationships *has-species* and *has-reaction*, respectively. For example, the declarations:

```

REACTOR_VESSEL is-a modeled-unit
has-species WATER
has-species o_XYLENE
has-species PHTHALIC_ANHYDRIDE
has-species OXYGEN
has-reaction RXN_101
  
```

illustrate a modeled-unit, REACTOR_VESSEL, with a four chemical species, WATER, o_XYLENE, PHTHALIC_ANHYDRIDE, and OXYGEN, and a chemical reaction, RXN_101. The species and reactions assumed in the <chemical content> of the modeled-unit will be a subset of those in the production of <chemical characterization> for the overall <phenomena-based model>.

- **Behavioral Characterization:** The behavioral characterization of the modeled-unit involves specialized assumptions regarding its operation. It can be used to express assumptions such as those in the <derivation context> of the overall <phenomena-based model> localized to a particular modeled-unit using the semantic relationship *is-modeled-as*. For example, the

declarations:

```
MIXING_POINT      is-a modeled-unit
                  is-modeled-as no-holdup
```

indicates a modeled-unit, MIXING_POINT, assumed to have no-holdup (i.e., the accumulation terms in the balance equation for the modeled-unit will be zero).

3.4.2 Flux Characterization

Each flux represents transport across the boundaries of two separate interacting modeled-units, or between a modeled-unit and the unmodeled surrounding environment. Assumptions necessary to fully define an instance of a flux are represented by the production tree depicted in Figure 3-11.

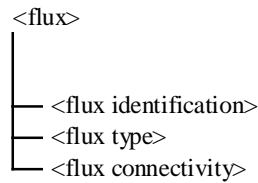


Figure 3-11: Flux Production Tree

- **Flux Identification:** The variable <flux identification> identifies the modeling element as a flux and the unique textual name used to refer to it. It is analogous to the variable <unit identification> of a modeled-unit. Declaration of a flux in the phenomena-based model is represented using the semantic relationship *is-a*, as illustrated by the declaration:

```
LIQUID_RECYCLE is-a flux
```

where LIQUID_RECYCLE is the name given to a particular instance of a flux.

- **Flux Type:** The type of a flux may be assumed to be either convective material transport, energy transport, or transport of a selected chemical species. This assumption is characterized by the production tree for <flux type> illustrated in Figure 3-12.

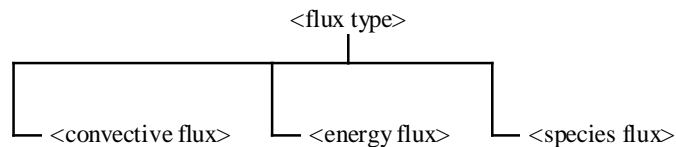


Figure 3-12: Flux Type Production Tree

Declaration of a convective flux requires additional decisions regarding the physical state of the transported material, a thermodynamic equation of state for the transported material, and a convective transport mechanism assumed to drive the flow, as illustrated by the production tree for <convective flux> in Figure 3-13.

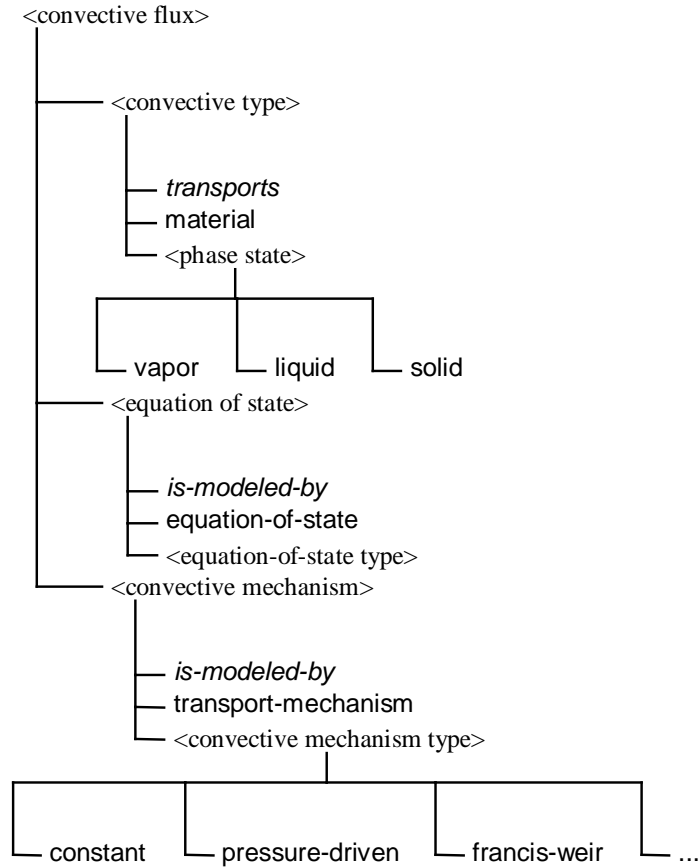


Figure 3-13: Convective Flux Production Tree

For example, the declarations:

```

GAS_PURGE is-a flux
           transports material vapor
           is-modeled-by equation-of-state ideal-gas
           is-modeled-by transport-mechanism pressure-driven
  
```

indicates pressure-driven gaseous flow of material whose thermodynamic behavior is modeled as an ideal gas.

Declaration of an energy flux requires a decision regarding the transport mechanism assumed to drive the flow, as illustrated by the production tree for <energy flux> in Figure 3-14.

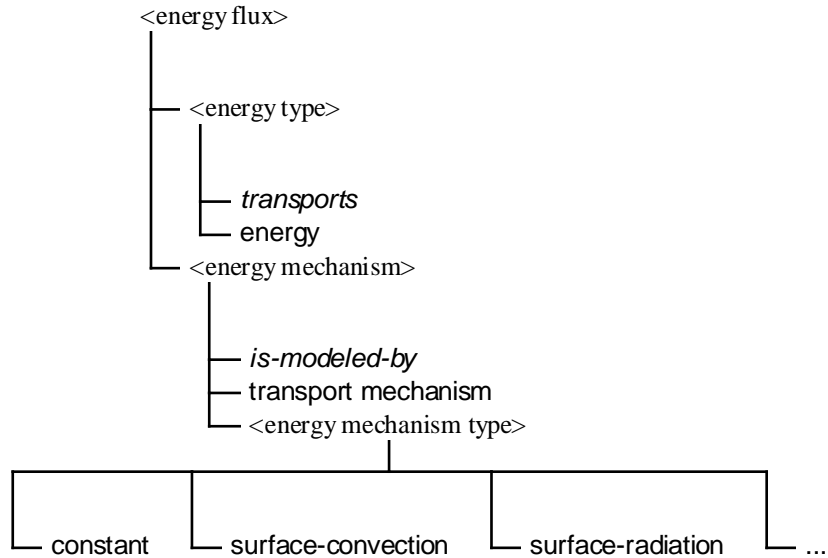


Figure 3-14: Energy Flux Production Tree

For example, the declarations:

```

REACTOR_Q  is-a flux
            transports energy
            is-modeled-by transport-mechanism surface-convection
  
```

indicates an energy flow driven by a surface convection transport mechanism.

Declaration of a species flux requires decisions regarding which species is transported and the transport mechanism assumed to drive the flow, as illustrated by the production tree for <species flux> in Figure 3-15. For example, the declarations:

```

REACTOR_Q  is-a flux
            transports species OXYGEN
            is-modeled-by transport-mechanism fickian-diffusion
  
```

indicates transport of species OXYGEN due to Fickian diffusion.

The definition of all three flux types have two semantic relationships in common. The semantic relationships *transports* indicates the type of transport, while the semantic relationship *is-modeled-by* is used to characterize a flux mechanistically.

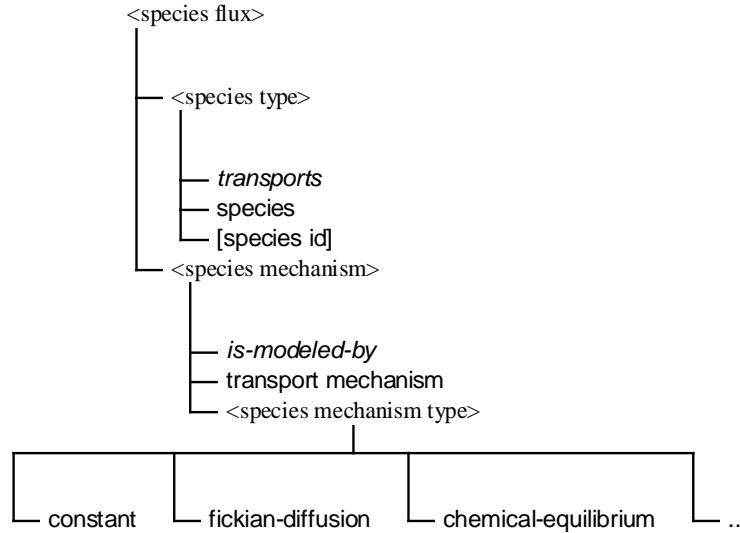


Figure 3-15: Species Flux Production Tree

- Flux Connectivity: The connectivity of a flux identifies the modeled-units that it associates. The production tree for <flux connectivity> is illustrated in Figure 3-16.

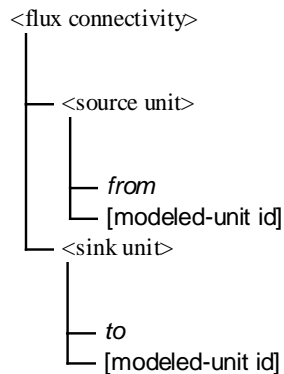


Figure 3-16: Flux Connectivity Production Tree

The source and sink modeled-units are identified by the semantic relationships *from* and *to*, respectively, as illustrated by the declarations:

```

REACTOR_Q  is-a flux
            from REACTOR_VESSEL
            to COOLING_JACKET
  
```

3.4.3 Material-Content Characterization

An elementary modeled-unit may be modeled as a blackbox, or as a unit with a material-content.

Physically, material-content refers to a region with no internal boundaries containing one or more thermodynamic phases at equilibrium. A material-content is represented in the mathematical model by relationships expressing thermal, physical, and chemical equilibrium. Assumptions necessary to fully define an instance of a material-content are characterized by the production tree for <material-content> illustrated in Figure 3-17.

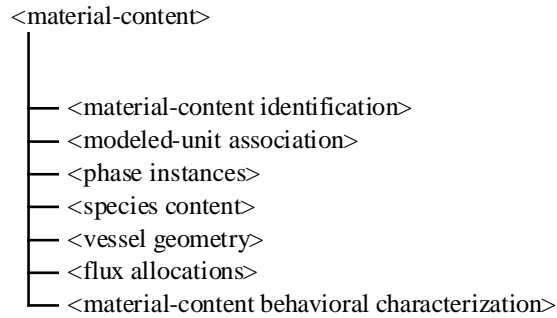


Figure 3-17: Material-Content Production Tree

- Material-Content Identification: The variable <material-content identification> identifies the modeling element as a material-content and the unique textual name used to refer to it. It is analogous to the variable <unit identification> of a modeled-unit. Declaration of a material-content in the phenomena-based model is represented using the semantic relationship *is-a*, as illustrated by the declaration:

FLASH_MATL *is-a* material-content

where FLASH_MATL is the name given to a particular instance of a material-content.

- Modeled-Unit Association: The variable <modeled-unit association> identifies the modeled-unit with which the material-content is associated with. It is declared using the semantic relationship *is-material-content-of* which is complementary to the semantic relationship *has-material-content* that relates a modeled-unit to a material-content, as illustrated by the declarations:

```

FLASH_MATL  is-a material-content
             is-material-content-in FLASH

FLASH       is-a modeled-unit
             has-material-content FLASH_MATL
  
```

- Phase Instances: The variable <phase instances> identifies the phases assumed to compose a material-content. These declarations are characterized by the production tree in Figure 3-18.

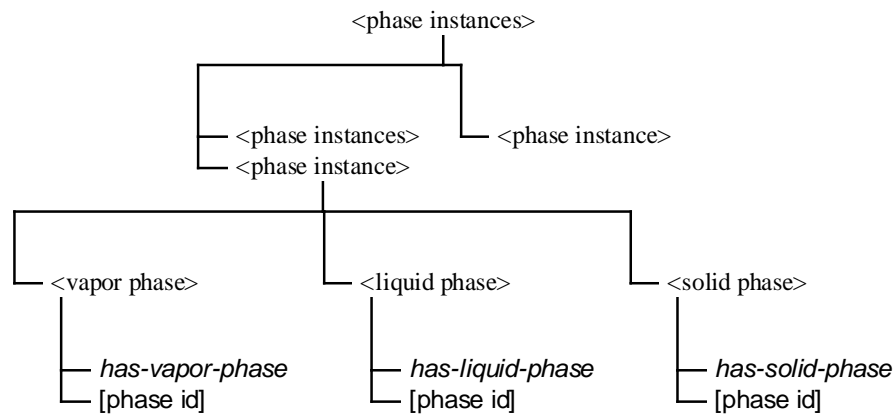


Figure 3-18: Phase Instance Production Tree

Vapor, liquid, and solid phases are declared using the semantic relationships *has-vapor-phase*, *has-liquid-phase*, and *has-solid-phase*, respectively, as illustrated by the declarations:

```

FLASH_MATL is-a material-content
           has-vapor-phase FLASH_MATL_V
           has-liquid-phase FLASH_MATL_L
  
```

where FLASH_MATL is a material-content composed of a vapor phase and a liquid phase at equilibrium. Note from the production tree for <phase instances> that a material-content must contain at least one phase.

- Species Content: The variable <species content> has already been discussed in the definition of a modeled-unit. Chemical species assumed to be present in the material-content are identified by the semantic relationship *has-species*, as illustrated by the declarations:

```

FLASH_MATL is-a material-content
           has-species BENZENE
           has-species TOLUENE
  
```

where FLASH_MATL contains two chemical species, BENZENE and TOLUENE.

- Vessel Geometry: The variable <vessel geometry> identifies the geometry of the vessel assumed to contain the material-content. This optional assumption, illustrated in the production tree of Figure 3-19, is used to derive an expression that relates the height of the contained phases as

a function of total volume.

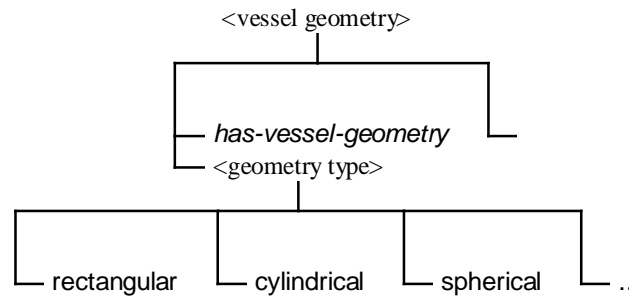


Figure 3-19: Vessel Geometry Production Tree

The geometry is identified using the semantic relationship, *has-vessel-geometry*, as illustrated by the declarations:

```
FLASH_MATL is-a material-content
           has-vessel-geometry spherical
```

- Flux Allocations: The variable *<flux allocations>* identifies the allocation of boundary fluxes to or from the associated modeled-unit to individual phases of the material-content. Primarily, this is used to relate to state of an outgoing convective flux to one of the phases of a material-content. Boundary fluxes may be allocated directly to a phase, or determined by geometry. Fluxes that do not need to be allocated are assigned to the material-content itself. The allocation of boundary fluxes, identified by semantic relationships *has-boundary-flux*, to the phases or geometry of a modeled-unit with a material-content are identified by the semantic relationship *allocated-to*, as illustrated by the production tree in Figure 3-20. Examples of these declarations are given below:

```
FLASH      is-a modeled-unit
           has-material-content FLASH_MATL
           has-convective-input FEED
           has-convective-output OVERHEAD
           has-convective-output BOTTOMS

FLASH_MATL is-a material-content
           has-vapor-phase FLASH_MATL_V
           has-liquid-phase FLASH_MATL_L
           has-boundary-flux FEED allocated-to self
```

has-boundary-flux OVERHEAD *allocated-to* FLASH_MATL_V
has-boundary-flux BOTTOMS *allocated-to* FLASH_MATL_L

In this example, there is a convective input FEED into the FLASH which has a material FLASH_MATL with a vapor phase, FLASH_MATL_V, and liquid phase, FLASH_MATL_L, at equilibrium. A convective stream OVERHEAD withdraws vapor material from the FLASH and another convective stream BOTTOMS withdraws liquid material from the FLASH.

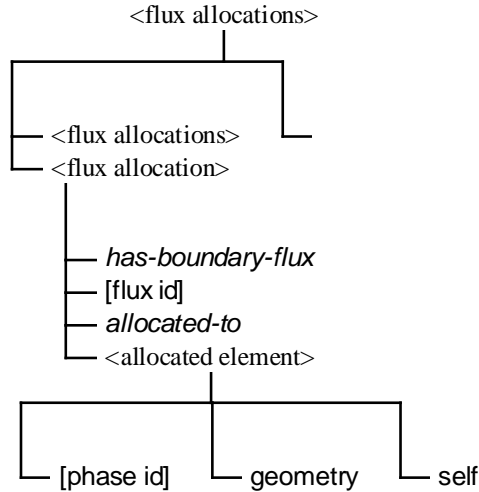


Figure 3-20: Flux Allocations Production Tree

- **Behavioral Characterization:** The behavioral characterization of the material-content involves specialized assumptions regarding its behavior (e.g., isobaric, isothermal, constant volume, etc.). These assumptions are made using the semantic relationship *is-modeled-as*, as illustrated by the declarations:

FLASH_MATL *is-a* material-content
is-modeled-as constant-volume

3.4.4 Phase Characterization

A phase represents a region with spatially uniform thermodynamic and physical properties. Each phase is represented in the mathematical model by relationships describing its physical and thermodynamic properties as functions of temperature, pressure, and composition. Assumptions required to fully define an instance of a phase are illustrated by the production tree in Figure 3-21.

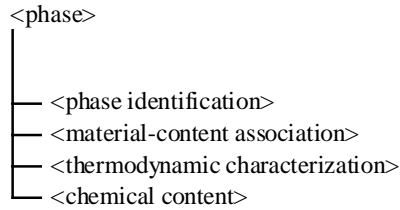


Figure 3-21: Phase Production Tree

- **Phase Identification:** The variable <phase identification> identifies the modeling element as a phase, the physical state of the phase, and the unique textual name used to refer it. This is accomplished using the semantic relationship *is-a*, as shown by the production tree for <phase identification> illustrated in Figure 3-22.

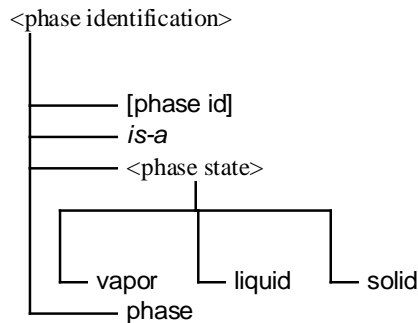


Figure 3-22: Phase Identification Production Tree

This results in a declaration in the phenomena-based model such as:

FLASH_MATL_L *is-a* liquid phase

where FLASH_MATL_L is the name given to a particular instance of a liquid phase.

- **Material-Content Association:** The variable <material-content association> identifies the material-content with which the phase is associated with. It is declared using the semantic relationship *is-phase-in* as illustrated by the declarations:

```

FLASH_MATL_V    is-a vapor phase
                 is-phase-in FLASH_MATL

FLASH_MATL_L    is-a liquid phase
                 is-phase-in FLASH_MATL

FLASH_MATL      is-a material-content
                 is-vapor-phase FLASH_MATL_V
  
```

has-liquid-phase FLASH_MATL_L

where FLASH_MATL_V and FLASH_MATL_L are names given to two separate instances of phases in a material-content FLASH_MATL.

- **Thermodynamic Characterization:** The variable <thermodynamic phase characterization> introduces a mechanistic characterization of a given phase. Either an equation of state or activity coefficient model may be selected, as illustrated in the production tree in Figure 3-23.

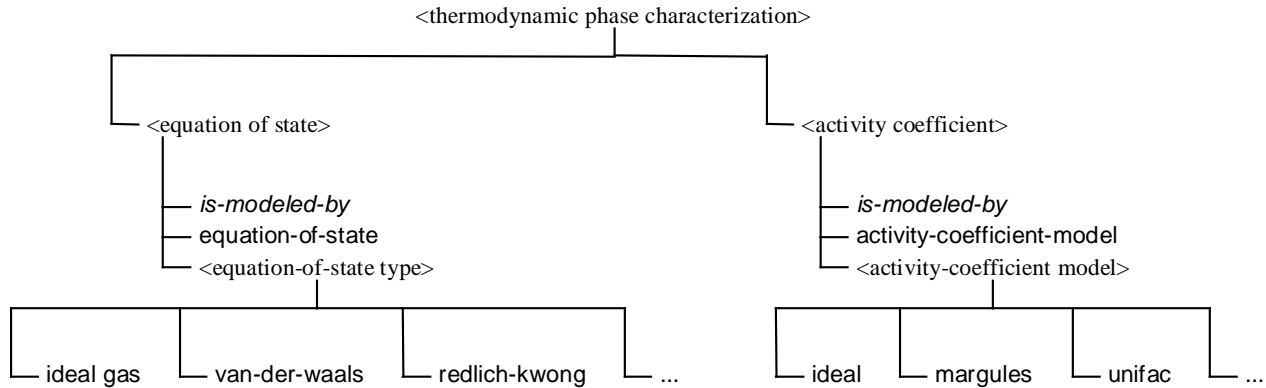


Figure 3-23: Thermodynamic Phase Characterization Production Tree

The selected mechanistic characterization is declared using the semantic relationship *is-modeled-by*, as illustrated by the declaration:

```
FLASH_MATL_V    is-a vapor phase
                is-modeled-by equation-of-state redlich-kwong
```

- **Chemical Content:** The variable <chemical content> has already been discussed in the definition of a modeled-unit. Chemical species and reaction assumed to be present in the phase are identified by the semantic relationships *has-species*, and *has-reaction*, respectively, as illustrated by the declarations:

```
VESSEL_MATL_L  is-a liquid phase
                has-species WATER
                has-species o_XYLENE
                has-species PHTHALIC_ANHYDRIDE
                has-species OXYGEN
                has-reaction RXN_101
```

where a phase, VESSEL_MATL_L, is assumed to contain four chemical species, WATER, o_XYLENE, PHTHALIC_ANHYDRIDE, and OXYGEN, and a chemical reaction, RXN_101.

3.4.5 Chemical Species Characterization

An instance of a chemical species in a phenomena-based model is characterized by the production tree illustrated in Figure 3-24.

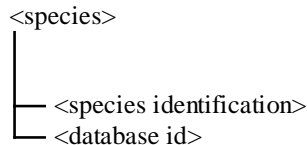


Figure 3-24: Species Production Tree

- **Species Identification:** The variable <species identification> identifies the modeling element as a chemical species and the unique textual name used to refer to it. It is analogous to the variable <unit identification> of a modeled-unit. Declaration of a species in the phenomena-based model is represented using the semantic relationship *is-a*, as illustrated by the declaration:

OXYGEN *is-a* species

where OXYGEN is the name given to a particular instance of a species.

- **Database Identification:** The variable <database id> identifies the unique identification tag used to access the properties of a chemical species from a database. This tag may be identified using a semantic relationship such as *has-database-id*, as illustrated by the declarations:

OXYGEN *is-a* species
 has-database-id 901

Here it is assumed that a database is used to access physical and thermodynamic property correlations for each species. Alternatively, more complex productions can be used for the chemical species that contain this data explicitly.

3.4.6 Chemical Reaction Characterization

An instance of a chemical species in a phenomena-based model is characterized by the production

tree illustrated in Figure 3-25.

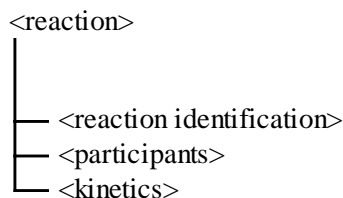


Figure 3-25: Reaction Production Tree

- **Reaction Identification:** The variable <reaction identification> identifies the modeling element as a chemical reaction and the unique textual name used to refer to it. It is analogous to the variable <unit identification> of a modeled-unit. Declaration of a reaction in the phenomena-based model is represented using the semantic relationship *is-a*, as illustrated by the declaration:

RXN_101 *is-a* reaction

where RXN_101 is the name given to a particular instance of a reaction.

- **Participants:** The variable <participants> identifies the reactants and products of the reaction, whether the reaction is modeled as irreversible, reversible, or equilibrium, and any relevant catalyst. These productions are illustrated in the production tree in Figure 3-26. For example, the declarations:

RXN_A *is-a* reaction

+ 1 ACETIC_ACID + 1 1_BUTANOL <=> + 1 n_BUTYL_ACETATE + 1 WATER

indicate a reversible reaction, RXN_A, of species ACETIC_ACID and 1_BUTANOL to form n_BUTYL_ACETATE and WATER.

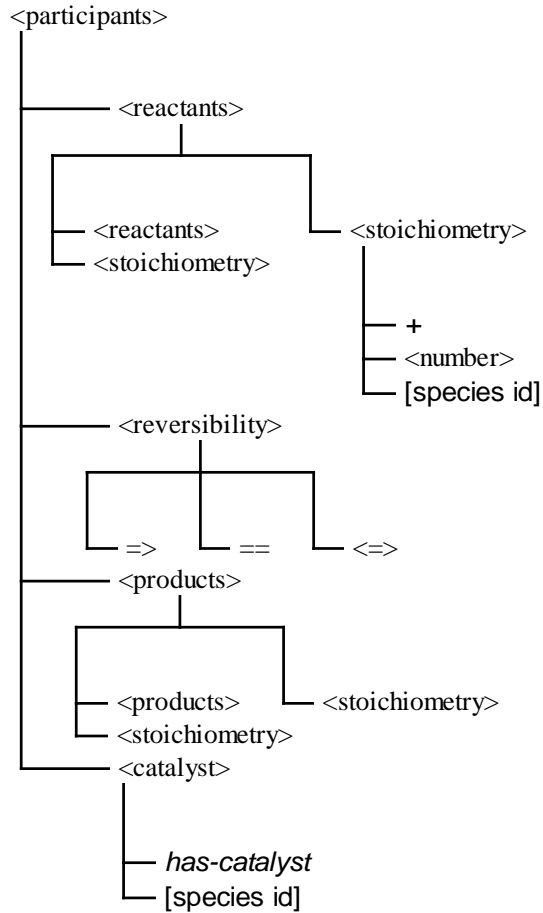


Figure 3-26: Reaction Participants Production Tree

- **Kinetics:** For rate-based reactions, the variable <kinetics> identifies the rate laws for the forward and, if the reaction is reversible, reverse rate laws. These forward and reverse rate laws are equations identified by the semantic relationships *has-forward-kinetics* and *has-reverse-kinetics*, as illustrated by the production tree in Figure 3-27.

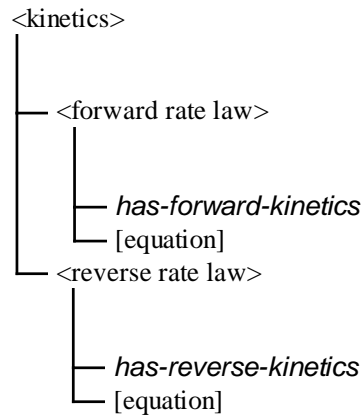


Figure 3-27: Reaction Kinetics Production Tree

3.5 Semantic Relationships

In the previous section, the assumptions that compose a phenomena-based model have been structured hierarchically using production rules that involve substitution of a variable with one or more non-terminal variables. In order to complete the grammar of the phenomena-based modeling language, production rules which substitute terminals in place of variables were then introduced. The format of these terminal substitutions were designed to consist of two parts: a semantic relationship and an identifier. These identifiers refer to the textual name of a modeling element or another elementary data type (i.e., integer, real, or string).

The semantic relationships unambiguously describe how the modeling elements are interrelated in forming a particular instance of a process model. In this section, these semantic relationships are summarized by organizing them into several categories.

- **Identification:** The *is-a* semantic relationship is used to identify the type of each modeling element. The *is-a* relationship links a unique textual name identifying the element to a string that identifies the type of the element.
- **Hierarchical Structure:** The parent-subunit relationship is captured by the symmetric semantic relationships *is-subunit-of* and *has-subunit*.

Spatially distributed modeled-units are modeling using a differential element approach. They are characterized by a coordinate system, identified by semantic relationship *has-spatial-distribution*, the distributed dimensions, which are identified by semantic relationship *has-distributed-dimension*, and the differential element subunits, which are identified by semantic relationship *has-differential-subunit*.

- **Material-Content:** The modeled-unit-material-content relationship is captured by the symmetric relationships *has-material-content* and *is-material-content-of*.

The phases of a material-content are identified by semantic relationships *has-vapor-phase*, *has-liquid-phase*, and *has-solid-phase*. The associated material-content of a phase is identified by semantic relationship *is-phase-in*.

The geometry of a material-content is identified by the semantic relationship *has-vessel-geometry*.

- **Topological Structure:** The boundary fluxes of a modeled-unit are identified by the semantic relationships *has-convective-input*, *has-convective-output*, *has-energy-input*, *has-energy-output*,

has-species-input, and *has-species-output*.

The allocation of boundary fluxes, identified by semantic relationships *has-boundary-flux*, to the phases or geometry of a modeled-unit with a material-content are identified by the semantic relationship *allocated-to*.

The modeled-units connected by a flux are identified by semantic relationships *from* and *to*. The type of flux is identified by semantic relationship *transports*.

- Chemical Characterization: The chemical species and reactions in a modeled-unit, material-content, or phase, are identified by semantic relationships *has-species* and *has-reaction*, respectively.
- Mechanistic Characterization: Mechanistic characterizations of thermodynamic property models, transport mechanisms, and reaction rate laws are identified by the semantic relationship *is-modeled-by*.
- Behavioral Characterization: Behavioral characterizations of modeled-units and material-contents are identified by the semantic relationship *is-modeled-as*.

3.6 Model Digraph

The introduction of semantic relationships into the modeling language allows a structured representation of phenomena-based process models using semantic networks. This semantic network may be depicted as a directed graph, where the vertices are labeled with names of modeling elements or another elementary data type, and the edges are labeled with semantic relationships. This representation organizes the knowledge behind the model and provides a structured and modular means of analyzing the phenomena-based model. For example, the declarations:

JACKET	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> JACKETED_CSTR <i>has-energy-input</i> q
q	<i>is-a</i> flux <i>transports</i> energy <i>to</i> JACKET <i>is-modeled-by</i> transport-mechanism surface-convection

JACKETED_CSTR *is-a* modeled-unit
 has-internal-unit JACKET

may be represented by the model digraph illustrated in Figure 3-28.

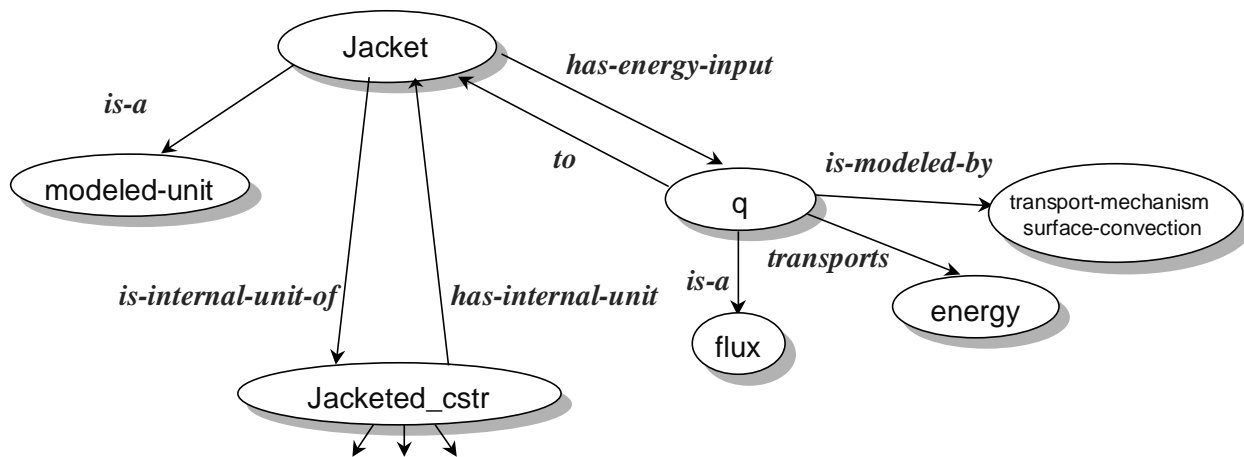


Figure 3-28: Example Model Digraph

3.7 Model Derivation Tree

The sequence of substitutions used to obtain a string (i.e., an instance of a model) in the language of MODEL.LA is characterized by the *derivation tree* of the model. The derivation tree is an extended production tree that includes details regarding which alternative decisions were made during the construction of a model. Similar to the production tree, the derivation tree captures the sequence of modeling decisions by recording the production rules and alternatives selected in a hierarchical manner. For example, the declarations:

HDA_Plant *is-a* modeled-unit
 has-internal-unit Separation_section
 has-internal-unit Reaction_section
 has-convective-input reactants
 has-convective-output products

are captured by the derivation tree in Figure 3-29.

3.8 Complete Context-Free Grammar Description

The complete context-free grammar of the MODEL.LA modeling language is given in Appendix A.

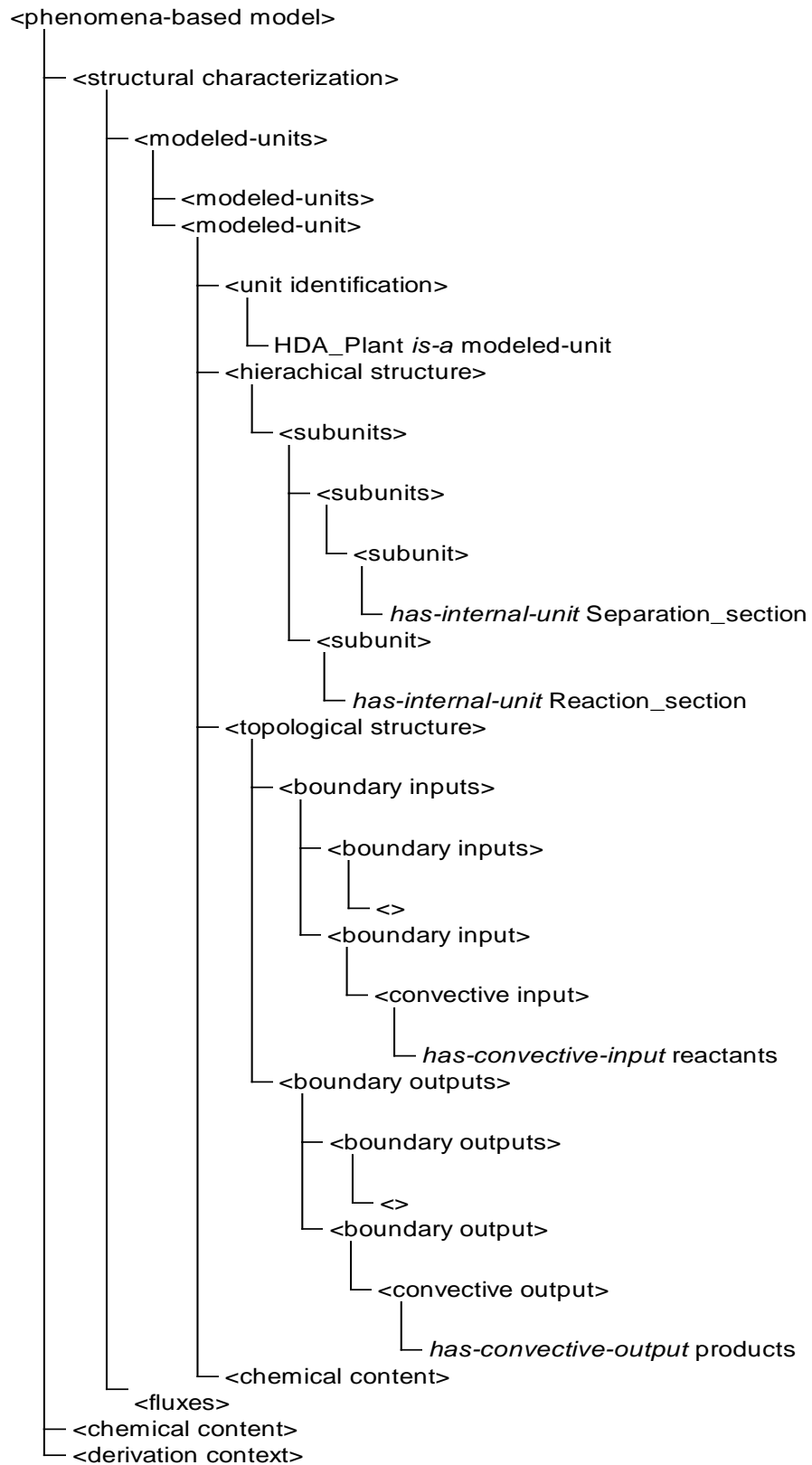


Figure 3-29: Example Model Derivation Tree

Chapter 4

Modeling Logic Framework

In the previous chapter, the context-free grammar of the MODEL.LA modeling language, which specifies the *syntax* of the modeling language, was introduced. This grammar formally describes the production rules that generate instances of the modeling elements and semantic relationships that compose a phenomena-based model. Essentially, the modeling elements and semantic relationships of the language provide a *vocabulary* that allows an engineer to articulate assumptions about the topological and hierarchical structure, physicochemical phenomena, and mechanistic characterizations of a chemical process. However, relying on syntax and common-sense or by-example explanations of the meaning of the modeling elements would leave the definition of the modeling language incomplete. In order to describe the *semantics* of the language, and to enable a computer to understand phenomena-based modeling assumptions and chemical process modeling principles, the underlying logic of model development must be elucidated.

To complete the specification of MODEL.LA, in this chapter a modeling logic framework is presented that describes the semantics of the modeling language (by formally describing the impact of phenomena-based assumptions on the resulting mathematical model) and makes it possible to systematize many aspects of the model development process. This systematization can enable a computer to comprehend the implication of the modeling assumptions, to assist the modeler in constructing the phenomena-based model description, to detect model inconsistencies and incompleteness, to automatically derive mathematical models, and to explain the terms and equations of the resulting mathematical model in terms of the modeling assumptions. Furthermore, the modeling logic framework provides a basis for introducing supervisory logic

into the modeling activity, which can guide inexperienced modelers toward completion of certain contextual modeling goals.

In order to assist engineers develop process models, a variety of modeling methodologies have been proposed (e.g., Aris, 1979, and Denn, 1986). These methodologies are commonly presented as flowcharts that provide very generic templates of the major tasks that a modeler tackles during model development. These flowcharts are frequently supplemented with several common-sense rules or heuristics that may also be employed under certain circumstances. Unfortunately, the level of granularity of guidance that these methodologies provide is too broad, the assistance they offer is too passive, and the knowledge they contain is too unstructured to offer real assistance to inexperienced modelers. This is because their lack of formalism restricts systematization of the modeling activity. As such, existing methodologies, while valid, cannot adequately address the modeling needs of either the chemical process industry or chemical engineering education.

The MODEL.LA modeling logic does not introduce new concepts of chemical engineering science and modeling expertise. Rather, through logical constructs based on the MODEL.LA modeling language, it makes it possible to express these chemical engineering concepts in a computational framework. In this manner, a computer can understand chemical process modeling principles. This allows systematization of the modeling activity, where such modeling knowledge makes possible varying degrees of computer-aided modeling support, including full automation, structured interaction, and explicit documentation.

4.1 Computational Logic

The MODEL.LA modeling logic presented in this chapter expresses chemical process modeling knowledge in a computational framework. Discussion of its implementation in this work is deferred until the subsequent chapter. The goal of this chapter is to present the embedded chemical process modeling knowledge in a concise, structured, and extendible manner that clearly identifies the principles of chemical engineering science applied in model development. This knowledge may subsequently be used to develop algorithms or programs through various implementations (e.g., expert systems, procedural subroutines, or object-oriented programs). However, decisions for implementation are considered separately from the modeling logic

framework itself.

4.2 Formal Description of Modeling Logic Operators

Logical operators, which represent tasks that comprise the modeling activity, form the basis of the MODEL.LA modeling logic. These logical modeling operators are classified as either *declarative* or *procedural*. Declarative operators allow assertions to be proposed about the state of the model (e.g., *Reactor-x is adiabatic*). Each declarative operator is characterized by three attributes:

$$\textit{declarative operator} = \langle \textit{arguments}, \textit{preconditions}, \textit{postconditions} \rangle$$

where

1. *Arguments* are the model elements on which the operator acts,
2. *Preconditions* are the necessary conditions that must be true before the operator is activated, and
3. *Postconditions* are the necessary conditions that are asserted by the activated operator.

Unlike declarative operators, procedural operators change the state of the model (e.g., *decompose Plant-Y into a Reaction-Section and a Separation-Section*). Each procedural operator is characterized by three attributes:

$$\textit{procedural operator} = \langle \textit{arguments}, \textit{preconditions}, \textit{suboperations} \rangle$$

where

1. *Suboperations* are the set of state operations and subtasks into which the operator is decomposed. This approach allows a hierarchical description of modeling tasks, where each modeling task can be decomposed into a set of smaller subtasks.

The arguments and preconditions for procedural operators are defined as for declarative operators.

To formally compose each operator, a variation of the notation of first-order predicate logic is used. Each operator is represented as an implication, expressed generically as:

$$\textit{antecedent} \Rightarrow \textit{consequent}$$

This may be interpreted as an *if-then* rule, where *if* the antecedent is true, *then* the consequent is asserted. For declarative operators, if the consequent also implies the antecedent, \Rightarrow is replaced by \Leftrightarrow . This may be interpreted as an *if-and-only-if* rule.

The antecedent is a *well-formed formula* composed of variable symbols, predicate symbols, connectives and quantifiers. Predicate symbols may be interpreted as *boolean functions*, which, based on the state values of their arguments, return a value of *true* or *false*. For example, the predicate $precond(arg_1, \dots, arg_n)$ returns a value of true or false based on the values of its n arguments arg_1, \dots, arg_n . The connectives include: \wedge (*and*), \vee (*or*), \neg (*not*), and the existential quantifier, \exists (*there exists*). The arguments of the operator appear as scoped variables and are identified at the head of the antecedent using the universal quantifier, \forall (*for all*). For example, in the formula $\forall x_1 \forall x_2 \dots \forall x_n$ variables x_1, x_2, \dots, x_n represent a list of n arguments. Variable type assertions for each variable are listed as predicate symbols in the antecedent. For example, the predicate symbols $type_id_1(x_1) \wedge type_id_2(x_2) \wedge \dots \wedge type_id_n(x_n)$, assert the types of variables x_1, x_2, \dots, x_n , respectively, where $type_id_i$ identifies the type of variable x_i . The variable type declarations in the antecedent are followed by the remaining preconditions, which are also represented by predicate symbols.

Postconditions of the declarative operators are expressed in the consequent, which is also a well-formed formula composed of predicate symbols, variable symbols, connectives and quantifiers. For procedural operators, the consequent contains the sequence of procedural suboperations, which are listed delimited by commas. Each suboperation may be another modeling logic operator, a statement containing other common mathematical, set, and assignment operations, or may contain function statements which introduce new state variables into the model. The latter appear generically as:

$$\Delta var(type_id(var))$$

where var is the new state variable introduced into the model, and $type_id$ is an asserted predicate identifying the type of the new state variable.

4.2.1 Modeling Logic Operators

As previously described, the semantic network digraph provides a structured, modular, and

organized view of an instance of a phenomena-based model. To exploit these features, the logical operators of the phenomena-based model will be defined in terms of the model digraph.

4.2.2 Elementary Graph Operators

The *state* of a phenomena-based model is represented formally by the model digraph. The model digraph is defined as:

$$M = (V, E)$$

where:

1. M is the model digraph,
2. V is the set of n labeled vertices $\{v_1, v_2, \dots, v_n\}$ in the graph, and
3. E is the set of m labeled, directed edges $\{e_1, e_2, \dots, e_m\}$ in the graph, where each edge is incident to and incident from a pair of vertices in the graph.

Elementary predicate symbols, which are assumed to be intrinsic declarative operators that make generic assertions about a model digraph are defined in Table 4-1.

Table 4-1: Intrinsic Declarative Graph Operators

<u><i>Operator</i></u>	<u><i>Actions</i></u>	<u><i>Preconditions</i></u>
<i>vertex(v)</i>	Identifies v is a vertex.	
<i>edge(e)</i>	Identifies e is a edge.	
<i>string(s)</i>	Identifies s is a string.	
<i>has_label(m, s)</i>	Asserts that the label of m is s .	$string(s) \wedge (edge(m) \vee vertex(m))$
<i>incident_from(e, v)</i>	Asserts that e is incident from v .	$edge(e) \wedge vertex(v)$
<i>incident_to(e, v)</i>	Asserts that e is incident to v .	$edge(e) \wedge vertex(v)$

Elementary predicate symbols, which are assumed to be intrinsic operators that generically access elements of the model digraph and allow their values to be assigned (e.g., $label(m) := \text{“Reactor-}x\text{”}$) are defined in Table 4-2.

Table 4-2: Intrinsic Graph Assignment Operators

<u><i>Operator</i></u>	<u><i>Actions</i></u>	<u><i>Preconditions</i></u>
<i>label(m)</i>	Accesses label of <i>m</i> .	$edge(m) \vee vertex(m)$
<i>incident_from(e)</i>	Accesses vertex that <i>e</i> is incident from.	$edge(e)$
<i>incident_to(e)</i>	Accesses vertex that <i>e</i> is incident to.	$edge(e)$

For convenience, several procedural operators that change the state of model graph *M* are defined in Table 4-3 in terms of these intrinsic graph operators.

4.3 Model Analysis Operators

In the previous section, a set of intrinsic and elementary graph operators were defined that provide means for analyzing and editing the model digraph. Since the state of the phenomena-based model is defined in terms of the model digraph, at the lowest level all modeling logic operators must be defined in terms of these intrinsic graph operators. However, by defining high-level operators in terms of these low-level operators, layers of abstraction (Abelson et al, 1996) are developed that separate the modeling activity from the actual underlying digraph representation. In turn, the high-level operators may be used to develop even more sophisticated modeling tasks. This use of abstraction enables the modeler to concentrate on modeling tasks such as “*define plant separation subsystem*” instead of “*add vertex and label to model digraph*”. With this methodology of abstraction in mind, several high-level declarative operators are now defined in terms of the model digraph to facilitate analysis of a phenomena-based model. In subsequent sections, these operators will provide the basis for describing operators for model construction, detection of model inconsistencies and incompleteness, and automated mathematical model generation. The model analysis operators are all defined as “if-and-only-if” implications, as denoted by the \Leftrightarrow symbol.

Table 4-3: Elementary Procedural Graph Operators

<u>Operator</u>	<u>Actions</u>
$\forall v \forall s [vertex(v) \wedge string(s) \wedge Add_vertex(v, s)$ \Rightarrow $label(v) := s,$ $V := V \cup v]$	<ol style="list-style-type: none"> 1.) Adds a vertex v to the model digraph, M, where $M = (V, E)$. 2.) Labels v with string s.
$\forall s [string(s) \wedge Add_new_vertex(s)$ \Rightarrow $\Delta v(vertex(v)),$ $Add_vertex(v, s)]$	<ol style="list-style-type: none"> 1.) Creates vertex v. 2.) Activates preceding operator to add v to model digraph with appropriate label.
$\forall e \forall v_1 \forall v_2 \forall s [edge(e) \wedge vertex(v_1) \wedge vertex(v_2)$ $\wedge string(s) \wedge Add_edge(e, v_1, v_2, s)$ \Rightarrow $label(e) := s,$ $incident_from(e) := v_1,$ $incident_to(e) := v_2,$ $E := E \cup e]$	<ol style="list-style-type: none"> 1.) Adds edge e to model digraph, M, where $M = (V, E)$. 2.) Sets incident from vertex of e to v_1. 3.) Sets incident to vertex of e to v_2. 4.) Labels e with string s.
$\forall v_1 \forall v_2 \forall s [vertex(v_1) \wedge vertex(v_2) \wedge string(s)$ $\wedge Add_new_edge(v_1, v_2, s)$ \Rightarrow $\Delta e(edge(e)),$ $Add_edge(e, v_1, v_2, s)]$	<ol style="list-style-type: none"> 1.) Creates edge e. 2.) Activates preceding operator to add e to model digraph with appropriate label.
$\forall v_1 \forall v_2 \forall s_1 \forall s_2 [vertex(v_1) \wedge vertex(v_2) \wedge string(s_1)$ $\wedge string(s_2)$ $\wedge Add_new_complementary_edges(v_1, v_2, s_1, s_2)$ \Rightarrow $Add_new_edge(v_1, v_2, s_1),$ $Add_new_edge(v_2, v_1, s_2)]$	<ol style="list-style-type: none"> 1.) Activates preceding operator to add an edge from v_1 to v_2 with appropriate label. 2.) Activates preceding operator to add an edge from v_2 to v_1 with appropriate label.
$\forall se \forall sv_1 \forall sv_2 [string(se) \wedge string(sv_1) \wedge string(sv_2)$ $\wedge Add_new_vertex_pair_and_edge(se, sv_1, sv_2)$ \Rightarrow $\Delta v_1(vertex(v_1)),$ $\Delta v_2(vertex(v_2)),$ $\Delta e(edge(e)),$ $Add_vertex(v_1, sv_1),$ $Add_vertex(v_2, sv_2),$ $Add_edge(e, v_1, v_2, s)]$	<ol style="list-style-type: none"> 1.) Creates vertex v_1. 2.) Creates vertex v_2. 3.) Creates edge e. 4.) Adds v_1, v_2, and e to model digraph with appropriate labels.

4.3.1 Modeling Element Identification

Every instance of a modeling element in the phenomena-based model is represented by a vertex in the model digraph. This vertex is labeled with the unique name, or identifier, of the modeling element. An edge labeled “is-a” that is incident from this vertex is incident to a vertex labeled with a string representing the type (e.g., “flux”) of the modeling element. Figure 4-1 illustrates an example of a modeled-unit modeling element named “Jacket” in a model digraph.

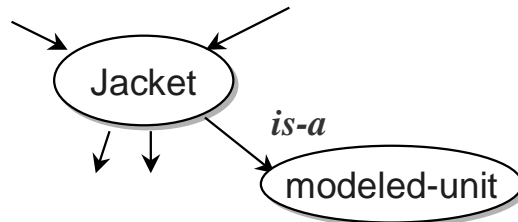


Figure 4-1: Modeled-Unit Digraph Representation

Several declarative operators that identify the particular type of a modeling element, which are represented by graph vertices with “is-a” edges incident from them, are defined in Table 4-4.

The declarative operators defined in Table 4-4 are also used by other operators that identify specialized (e.g., liquid flux) or generalized types of modeling elements. Several examples of these are listed in Table 4-5.

Table 4-4: Modeling Element Identification Operators

<u>Operator</u>	<u>Actions</u>
$\forall m$ [modeled_unit(m) \Leftrightarrow $vertex(m) \wedge \exists e \exists v (edge(e) \wedge vertex(v)$ $\wedge has_label(e, "is-a") \wedge has_label(v, "modeled-unit")$ $\wedge incident_from(e, m) \wedge incident_to(e, v))]$	Identifies m is a modeled-unit.
$\forall f$ [flux(f) \Leftrightarrow $vertex(f) \wedge \exists e \exists v (edge(e) \wedge vertex(v)$ $\wedge has_label(e, "is-a") \wedge has_label(v, "flux")$ $\wedge incident_from(e, f) \wedge incident_to(e, v))]$	Identifies f is a flux.
$\forall m$ [material-content(m) \Leftrightarrow $vertex(m) \wedge \exists e \exists v (edge(e) \wedge vertex(v)$ $\wedge has_label(e, "is-a") \wedge has_label(v, "material-content")$ $\wedge incident_from(e, m) \wedge incident_to(e, v))]$	Identifies m is a material-content.
$\forall p$ [vapor_phase(p) \Leftrightarrow $vertex(p) \wedge \exists e \exists v (edge(e) \wedge vertex(v)$ $\wedge has_label(e, "is-a") \wedge has_label(v, "vapor phase")$ $\wedge incident_from(e, p) \wedge incident_to(e, v))]$	Identifies p is a vapor phase.
$\forall p$ [liquid_phase(p) \Leftrightarrow $vertex(p) \wedge \exists e \exists v (edge(e) \wedge vertex(v)$ $\wedge has_label(e, "is-a") \wedge has_label(v, "liquid phase")$ $\wedge incident_from(e, p) \wedge incident_to(e, v))]$	Identifies p is a liquid phase.
$\forall p$ [solid_phase(p) \Leftrightarrow $vertex(p) \wedge \exists e \exists v (edge(e) \wedge vertex(v)$ $\wedge has_label(e, "is-a") \wedge has_label(v, "solid phase")$ $\wedge incident_from(e, p) \wedge incident_to(e, v))]$	Identifies p is a solid phase.
$\forall r$ [reaction(r) \Leftrightarrow $vertex(r) \wedge \exists e \exists v (edge(e) \wedge vertex(v)$ $\wedge has_label(e, "is-a") \wedge has_label(v, "reaction")$ $\wedge incident_from(e, r) \wedge incident_to(e, v))]$	Identifies r is a reaction.
$\forall s$ [species(s) \Leftrightarrow $vertex(s) \wedge \exists e \exists v (edge(e) \wedge vertex(v)$ $\wedge has_label(e, "is-a") \wedge has_label(v, "species")$ $\wedge incident_from(e, s) \wedge incident_to(e, v))]$	Identifies s is a species.

Table 4-5: Specialized Modeling Element Identification Operators

<i><u>Operator</u></i>	<i><u>Actions</u></i>
$\forall f$ [<i>vapor_flux</i> (<i>f</i>) \Leftrightarrow $flux(f) \wedge \exists e \exists v (edge(e) \wedge vertex(v) \wedge has_label(e, "transports")$ $\wedge has_label(v, "material\ vapor") \wedge incident_from(e, f)$ $\wedge incident_to(e, v))]$	Identifies <i>f</i> is a vapor convective flux
$\forall f$ [<i>liquid_flux</i> (<i>f</i>) \Leftrightarrow $flux(f) \wedge \exists e \exists v (edge(e) \wedge vertex(v) \wedge has_label(e, "transports")$ $\wedge has_label(v, "material\ liquid") \wedge incident_from(e, f)$ $\wedge incident_to(e, v))]$	Identifies <i>f</i> is a liquid convective flux
$\forall f$ [<i>solid_flux</i> (<i>f</i>) \Leftrightarrow $flux(f) \wedge \exists e \exists v (edge(e) \wedge vertex(v) \wedge has_label(e, "transports")$ $\wedge has_label(v, "material\ solid") \wedge incident_from(e, f)$ $\wedge incident_to(e, v))]$	Identifies <i>f</i> is a solid convective flux
$\forall f$ [<i>convective_flux</i> (<i>f</i>) $\Leftrightarrow vapor_flux(p) \vee liquid_flux(p) \vee solid_flux(p)]$	Identifies <i>f</i> is a convective flux
$\forall f$ [<i>energy_flux</i> (<i>f</i>) \Leftrightarrow $flux(f) \wedge \exists e \exists v (edge(e) \wedge vertex(v) \wedge has_label(e, "transports")$ $\wedge has_label(v, "energy") \wedge incident_from(e, f)$ $\wedge incident_to(e, v))]$	Identifies <i>f</i> is an energy flux
$\forall f \forall s$ [<i>species_flux</i> (<i>f</i> , <i>s</i>) \Leftrightarrow $flux(f) \wedge species(s) \wedge \exists e (\wedge edge(e) \wedge has_label(e, "transports")$ $\wedge incident_from(e, f) \wedge incident_to(e, s))]$	Asserts that species flux <i>f</i> transports species <i>s</i>
$\forall f$ [<i>species_flux</i> (<i>f</i>) $\Leftrightarrow flux(f) \wedge \exists s (species(s) \wedge species_flux(f, s))]$	Identifies <i>f</i> is a species flux
$\forall p$ [<i>phase</i> (<i>p</i>) $\Leftrightarrow vapor_phase(p) \vee liquid_phase(p) \vee solid_phase(p)]$	Identifies <i>p</i> is a phase
$\forall g$ [<i>geometry</i> (<i>g</i>) \Leftrightarrow $vertex(g) \wedge \exists e \exists v (edge(e) \wedge vertex(v) \wedge material_content(m)$ $\wedge has_label(e, "has-vessel-geometry") \wedge incident_from(e, m)$ $\wedge incident_to(e, g))]$	Identifies <i>g</i> is a geometry
$\forall s$ [<i>spatial_distribution</i> (<i>s</i>) \Leftrightarrow $vertex(s) \wedge \exists e \exists v (edge(e) \wedge vertex(v) \wedge modeled_unit(m)$ $\wedge has_label(e, "has-spatial-distribution") \wedge incident_from(e, m)$ $\wedge incident_to(e, s))]$	Identifies <i>s</i> is a spatial distribution.
$\forall m$ [<i>distributed_unit</i> (<i>m</i>) \Leftrightarrow $modeled_unit(m) \wedge \exists e \exists s (edge(e) \wedge spatial_distribution(s)$ $\wedge has_label(e, "has-spatial-distribution") \wedge incident_from(e, f)$ $\wedge incident_to(e, v))]$	Identifies <i>m</i> is a spatially distributed modeled-unit.

4.3.2 Hierarchical Structure

The hierarchical structure of modeled-units in a phenomena-based model is declared using the complementary semantic relationships *has-internal-unit* and *is-internal-unit-of*. Operators that provide for analysis of this hierarchical structure are listed in Table 4-6.

Table 4-6: Hierarchical Structure Analysis Operators

<u>Operator</u>	<u>Actions</u>
$\forall p [has_subunits(p)$ \Leftrightarrow $modeled_unit(p) \wedge \exists s \exists e (modeled_unit(s)$ $\wedge edge(e) \wedge has_label(e, "has-internal-unit")$ $\wedge incident_from(e, p) \wedge incident_to(e, s))]$	Asserts that modeled-unit p has subunits.
$\forall s [has_parent(s)$ \Leftrightarrow $modeled_unit(s) \wedge \exists p \exists e (modeled_unit(p)$ $\wedge edge(e) \wedge has_label(e, "is-internal-unit-of")$ $\wedge incident_from(e, s) \wedge incident_to(e, p))]$	Asserts that modeled-unit s has a parent unit.
$\forall p \forall s [has_subunit(p, s)$ \Leftrightarrow $modeled_unit(p) \wedge modeled_unit(s)$ $\wedge \exists e (edge(e) \wedge has_label(e, "has-internal-unit")$ $\wedge incident_from(e, p) \wedge incident_to(e, s))]$	Asserts that modeled-unit p has subunit s .
$\forall s \forall p [is_subunit_of(s, p)$ \Leftrightarrow $modeled_unit(s) \wedge modeled_unit(p)$ $\wedge \exists e (edge(e) \wedge has_label(e, "is-internal-unit-of")$ $\wedge incident_from(e, s) \wedge incident_to(e, p))]$	Asserts that s is a subunit of modeled-unit p
$\forall a \forall d [has_descendant(a, d)$ \Leftrightarrow $modeled_unit(a) \wedge modeled_unit(d)$ $\wedge (has_subunit(a, d) \vee \exists s (modeled_unit(s)$ $\wedge has_subunit(a, s) \wedge has_descendant(s, d))]$	Asserts that modeled-unit a has descendant d
$\forall d \forall a [has_ancestor(d, a)$ \Leftrightarrow $modeled_unit(d) \wedge modeled_unit(a)$ $\wedge (is_subunit_of(d, a) \vee \exists s (modeled_unit(s)$ $\wedge is_subunit_of(d, s) \wedge has_ancestor(s, a))]$	Asserts that modeled-unit d has ancestor a
$\forall p \forall s [has_differential_subunit(p, s)$ \Leftrightarrow $distributed_unit(p) \wedge modeled_unit(s) \wedge \exists e (edge(e)$ $\wedge has_label(e, "has-differential-subunit")$ $\wedge incident_from(e, p) \wedge incident_to(e, s))]$	Asserts that spatially distributed modeled-unit p has differential subunit s .

The fifth and sixth operators in Table 4-6, *has_descendant* and *has_ancestor*, are defined

recursively, where the operator is defined in terms of itself. When these recursive operators are evaluated, an assertion is made when the predicate disjunctive to the recursive predicate is true.

The following declarations will be used to illustrate use of these operators:

PLANT	<i>is-a</i> modeled-unit <i>has-internal-unit</i> REACTION_SECTION <i>has-internal-unit</i> SEPARATION_SECTION
REACTION_SECTION	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> PLANT <i>has-internal-unit</i> REACTION_PRETREAT <i>has-internal-unit</i> JACKETED_CSTR
JACKETED_CSTR	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> REACTION_SECTION <i>has-internal-unit</i> JACKET <i>has-internal-unit</i> VESSEL
JACKET	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> JACKETED_CSTR
VESSEL	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> JACKETED_CSTR

Examples of assertions that may be made from these declarations using the operators in Table 4-6 include: *has_subunits*(PLANT), *has_parent*(JACKETED_CSTR), *has_subunit*(REACTION_SECTION, REACTION_PRETREAT), *is_subunit_of*(JACKET, JACKETED_CSTR), *has_descendant*(PLANT, VESSEL), and *has_ancestor*(JACKET, REACTION_SECTION).

4.3.3 Topological Structure

The topological structure of a phenomena-based model reflects how modeled-units interact through transport of mass, energy, and chemical species between their boundaries. The boundary fluxes associated with a modeled-unit are identified by semantic relationships *has-convective-input*, *has-convective-output*, *has-energy-input*, *has-energy-output*, *has-species-input*, and *has-species-output*. The most refined modeled-units that a flux interconnects are identified by semantic relationships *to* and *from*. Operators that allow analysis of this topological structure are listed in Table 4-7.

Table 4-7: Topological Structure Analysis Operators

<u>Operator</u>	<u>Actions</u>
$\forall m \forall f [has_input_flux(m, f)$ \Leftrightarrow $modeled_unit(m) \wedge flux(f) \wedge \exists e (edge(e)$ $\wedge incident_from(e, m) \wedge incident_to(e, f)$ $\wedge (has_label(e, "has-convective-input")$ $\vee has_label(e, "has-energy-input")$ $\vee has_label(e, "has-species-input"))]$	Asserts that modeled-unit m has input flux f .
$\forall m \forall f [has_output_flux(m, f)$ \Leftrightarrow $modeled_unit(m) \wedge flux(f) \wedge \exists e (edge(e)$ $\wedge incident_from(e, m) \wedge incident_to(e, f)$ $\wedge (has_label(e, "has-convective-output")$ $\vee has_label(e, "has-energy-output")$ $\vee has_label(e, "has-species-output"))]$	Asserts that modeled-unit m has output flux f .
$\forall m \forall f [has_convective_input(m, f)$ \Leftrightarrow $has_input_flux(m, f) \wedge convective_flux(f)]$	Asserts that modeled-unit m has convective input flux f .
$\forall m \forall f [has_convective_output(m, f)$ \Leftrightarrow $has_output_flux(m, f) \wedge convective_flux(f)]$	Asserts that modeled-unit m has convective output flux f .
$\forall m \forall f [has_energy_input(m, f)$ \Leftrightarrow $has_input_flux(m, f) \wedge energy_flux(f)]$	Asserts that modeled-unit m has energy input flux f .
$\forall m \forall f [has_energy_output(m, f)$ \Leftrightarrow $has_output_flux(m, f) \wedge energy_flux(f)]$	Asserts that modeled-unit m has energy output flux f .
$\forall m \forall f [has_species_input(m, f)$ \Leftrightarrow $has_input_flux(m, f) \wedge species_flux(f)]$	Asserts that modeled-unit m has species input flux f .
$\forall m \forall f [has_species_output(m, f)$ \Leftrightarrow $has_output_flux(m, f) \wedge species_flux(f)]$	Asserts that modeled-unit m has species output flux f .
$\forall f \forall m [from(f, m)$ \Leftrightarrow $flux(f) \wedge modeled_unit(m) \wedge \exists e (edge(e)$ $\wedge has_label(e, "from") \wedge incident_from(e, f)$ $\wedge incident_to(e, m))]$	Asserts that f is a transport flux that originates incident from modeled-unit m .
$\forall f \forall m [to(f, m)$ \Leftrightarrow $flux(f) \wedge modeled_unit(m) \wedge \exists e (edge(e)$ $\wedge has_label(e, "to") \wedge incident_from(e, f)$ $\wedge incident_to(e, m))]$	Asserts that f is a transport flux that terminates incident to modeled-unit m .

The following declarations will be used to illustrate use of these operators:

JACKETED_CSTR	<i>is-a</i> modeled-unit <i>has-internal-unit</i> JACKET <i>has-internal-unit</i> VESSEL <i>has-convective-input</i> reactants <i>has-convective-output</i> products
VESSEL	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> JACKETED_CSTR <i>has-convective-input</i> reactants <i>has-convective-output</i> products <i>has-energy-input</i> q
q	<i>is-a</i> flux <i>transports</i> energy <i>from</i> VESSEL <i>to</i> JACKET
reactants	<i>is-a</i> flux <i>transports</i> material liquid <i>from</i> source <i>to</i> VESSEL

Examples of assertions that may be made from these declarations using the operators in Table 4-7 include: *has_energy_input*(VESSEL, q), *has_convective_input*(JACKETED_CSTR, reactants), *has_convective_input*(VESSEL, reactants), and *to*(q, VESSEL). Note that although *has_convective_input*(JACKETED_CSTR, reactants) is true, *to*(JACKETED_CSTR, q) cannot be asserted since the *to* operator only applies to the most-refined modeled-unit that the flux is incident to.

4.3.4 Material Characterization

The material characterization in a phenomena-based model is defined by instances of the modeling element material-content, which is declared for a modeled-unit using the complementary semantic relationships *has-material-content* and *is-material-content-in*. The phases that compose a material-content are identified by the semantic relationships *has-vapor-phase*, *has-liquid-phase*, and *has-solid-phase*. If a vessel geometry is assumed for the material-content, it is declared using the semantic

relationship *has-geometry*. Allocation of boundary fluxes, identified by semantic relationship *has-boundary-flux*, to the phases or geometry of a material-content are declared using the semantic relationship *is-allocated-to*. Operators that allow analysis of this material characterization are listed in Table 4-8.

The following declarations will be used to illustrate use of these operators:

FLASH	<i>is-a</i> modeled-unit <i>has-material-content</i> FLASH_MATL <i>has-convective-input</i> FEED <i>has-convective-output</i> OVERHEAD <i>has-convective-output</i> BOTTOMS
FLASH_MATL	<i>is-a</i> material-content <i>is-material-content-in</i> FLASH <i>has-vapor-phase</i> FLASH_MATL_V <i>has-liquid-phase</i> FLASH_MATL_L <i>has-geometry</i> vertical-cylinder <i>has-boundary-flux</i> FEED <i>allocated-to</i> self <i>has-boundary-flux</i> OVERHEAD <i>allocated-to</i> FLASH_MATL_V <i>has-boundary-flux</i> BOTTOMS <i>allocated-to</i> FLASH_MATL_L
FLASH_MATL_V	<i>is-a</i> vapor phase <i>is-phase-in</i> FLASH_MATL
FLASH_MATL_L	<i>is-a</i> liquid phase <i>is-phase-in</i> FLASH_MATL

Examples of assertions that may be made from these declarations using the operators in Table 4-8 include: *material_unit*(FLASH), *has_phase*(FLASH_MATL, FLASH_MATL_L), *has_geometry*(FLASH_MATL, vertical-cylinder), *has_boundary_flux*(FLASH_MATL, FEED), and *allocated_to*(FLASH_MATL, FLASH_MATL_L, BOTTOMS).

Table 4-8: Material Characterization Analysis Operators

<u>Operator</u>	<u>Actions</u>
$\forall u \forall m [has_material_content(u, m)$ \Leftrightarrow $modeled_unit(u) \wedge material_content(m)$ $\wedge \exists e (edge(e)$ $\wedge has_label(e, "has-material-content")$ $\wedge incident_from(e, u) \wedge incident_to(e, m))]$	Asserts that modeled-unit u has material-content m .
$\forall u [material_unit(u)$ \Leftrightarrow $modeled_unit(u) \wedge \exists m (material_content(m)$ $\wedge has_material_content(u, m))]$	Asserts that modeled-unit u has a material-content.
$\forall u [blackbox_unit(u)$ \Leftrightarrow $modeled_unit(u) \wedge \neg material_unit(u)$ $\wedge \neg has_subunits(u)]$	Asserts that modeled-unit u is modeled as a blackbox.
$\forall m \forall p [has_phase(m, p)$ \Leftrightarrow $material_content(m) \wedge phase(p) \wedge \exists e (edge(e)$ $\wedge (has_label(e, "has-vapor-phase")$ $\vee has_label(e, "has-vapor-phase")$ $\vee has_label(e, "has-vapor-phase"))$ $\wedge incident_from(e, m) \wedge incident_to(e, p))]$	Asserts that material-content m has phase p .
$\forall m \forall g [has_geometry(m, g)$ \Leftrightarrow $material_content(m) \wedge geometry(g)$ $\wedge \exists e (edge(e) \wedge has_label(e, "has-geometry")$ $\wedge incident_from(e, m) \wedge incident_to(e, g))]$	Asserts that material-content m has geometry g .
$\forall m \forall f [has_boundary_flux(m, f)$ \Leftrightarrow $material_content(m) \wedge flux(f) \wedge \exists e (edge(e)$ $\wedge has_label(e, "has-boundary-flux")$ $\wedge incident_from(e, m) \wedge incident_to(e, f))]$	Asserts that material-content m has boundary flux f .
$\forall m \forall p \forall f [allocated_to(m, p, f)$ \Leftrightarrow $material_content(m) \wedge phase(p) \wedge flux(f)$ $\wedge has_phase(m, p) \wedge has_boundary_flux(m, f)$ $\wedge \exists e (edge(e) \wedge has_label(e, "allocated-to")$ $\wedge incident_from(e, m) \wedge incident_to(e, p))]$	Asserts that material-content m has boundary flux f allocated to phase p .
$\forall m \forall g \forall f [allocated_to(m, g, f)$ \Leftrightarrow $material_content(m) \wedge geometry(g) \wedge flux(f)$ $\wedge geometry(m, g) \wedge has_boundary_flux(m, f)$ $\wedge \exists e (edge(e) \wedge has_label(e, "allocated-to")$ $\wedge incident_from(e, m) \wedge incident_to(e, g))]$	Asserts that material-content m has boundary flux f allocated to geometry g .
$\forall a [allocated_element(a)$ \Leftrightarrow $material_content(a) \vee phase(a) \vee geometry(a)]$	Asserts that a boundary flux can be allocated to modeling element a .

4.3.5 Chemical Content

The chemical content of a phenomena-based model identifies the chemical species and reactions assumed to be present in the modeled-units, material-contents, and phases of the model. The species and reaction present in these modeling elements are identified using the semantic relationships *has-species* and *has-reaction*, respectively. Operators that allow analysis of this chemical content are listed in Table 4-9. Additionally, the last operator determines if a particular chemical species is transported by a flux.

Table 4-9: Chemical Content Analysis Operators

<u>Operator</u>	<u>Actions</u>
$\forall m [species_element(m)$ \Leftrightarrow $modeled_unit(m) \vee material_content(m) \vee phase(m)]$	Asserts that species may be assigned to modeling element <i>m</i> .
$\forall m \forall s [has_species(m, s)$ \Leftrightarrow $species_element(m) \wedge species(s)$ $\wedge \exists e (edge(e) \wedge has_label(y, "has-species"))$ $\wedge incident_from(e, m) \wedge incident_to(e, s)]$	Asserts that species <i>s</i> is assigned to modeling element <i>m</i> .
$\forall m [reaction_element(m)$ \Leftrightarrow $modeled_unit(m) \vee phase(m)]$	Asserts that reactions may be assigned to modeling element <i>m</i> .
$\forall m \forall s [has_reaction(m, s)$ \Leftrightarrow $reaction_element(m) \wedge reaction(s)$ $\wedge \exists e (edge(e) \wedge has_label(y, "has-reaction"))$ $\wedge incident_from(e, m) \wedge incident_to(e, s)]$	Asserts that reaction <i>r</i> is assigned to modeling element <i>m</i> .
$\forall f \forall s [flux(f) \wedge species(s) \wedge transports_species(f, s)$ \Leftrightarrow $species_flux(f, s) \vee (convective_flux(f)$ $\wedge \exists u (modeled_unit(u) \wedge from(f, u)$ $\wedge ((blackbox_unit(u) \wedge has_species(u, s))$ $\vee \exists m (material_content(m) \wedge has_material_content(u, m)$ $\wedge ((has_geometry(m, g) \wedge allocated_to(m, g, f) \wedge has_species(m, s))$ $\vee (phase(p) \wedge has_phase(m, p) \wedge allocated_to(m, p, f)$ $\wedge has_species(p, s)))))))]$	Asserts that a flux transports a species if <i>i</i>) it is a corresponding species flux, <i>ii</i>) it is a convective flux from a blackbox unit that has the species, <i>iii</i>) it is allocated to the geometry of a material-content that has the species, or <i>iv</i>) it is allocated to a phase that has the species.

The following declarations will be used to illustrate use of these operators:

VESSEL_MATL_L *is-a* liquid phase
 has-species WATER
 has-species o_XYLENE

has-species PHTHALIC_ANHYDRIDE

has-species OXYGEN

has-reaction RXN_101

Examples of assertions that may be made from these declarations using the operators in Table 4-9 include: *species_element*(VESSEL_MATL_L) *has-species*(VESSEL_MATL_L, WATER), and *has_reaction*(VESSEL_MATL_L, RXN_101).

4.3.6 Mechanistic Characterization

The mechanistic characterization of a phenomena-based model identifies the transport mechanisms of fluxes, thermodynamic characterizations of phases, and kinetic rate laws of reactions. Transport mechanisms, equations of state, and activity coefficient models are identified by the semantic relationship *is-modeled-by*. Kinetic rate laws are identified by semantic relationships *has-forward-kinetics* and *has-reverse-kinetics*. Operators that allow analysis of such mechanistic characterizations are listed in Table 4-10.

The following declarations will be used to illustrate use of these operators:

VESSEL_MATL_V *is-a* vapor phase

has-equation-of-state ideal-gas

Examples of assertions that may be made from these declarations using the operators in Table 4-10 include: *equation_of_state*(ideal-gas) and *has_equation_of_state*(VESSEL_MATL_V, ideal-gas).

Table 4-10: Mechanistic Characterization Analysis Operators

<u>Operator</u>	<u>Actions</u>
$\forall m [transport_mechanism(m)$ \Leftrightarrow $vertex(m) \wedge has_label(m, "transport-mechanism *")$ $\wedge \exists f \exists e (flux(f) \wedge edge(e) \wedge has_label(e, "is-modeled-by"))$ $\wedge incident_from(e, f) \wedge incident_to(e, m)]$	Identifies m is a transport mechanism.
$\forall f \forall m [has_transport_mechanism(f, m)$ \Leftrightarrow $flux(f) \wedge transport_mechanism(m) \wedge \exists e (edge(e)$ $\wedge has_label(e, "is-modeled-by")) \wedge incident_from(e, f) \wedge incident_to(e, m)]$	Asserts that flux f has transport mechanism m .
$\forall s [equation_of_state(s)$ \Leftrightarrow $vertex(s) \wedge has_label(s, "equation-of-state *")$ $\wedge \exists p \exists e ((phase(p) \vee convective_flux(p)) \wedge edge(e)$ $\wedge has_label(e, "is-modeled-by")) \wedge incident_from(e, p) \wedge incident_to(e, s)]$	Identifies s is an equation of state.
$\forall p \forall s [has_equation_of_state(p, s)$ \Leftrightarrow $(phase(p) \vee convective_flux(p)) \wedge equation_of_state(s) \wedge \exists e (edge(e)$ $\wedge has_label(e, "is-modeled-by")) \wedge incident_from(e, p) \wedge incident_to(e, s)]$	Asserts that phase or convective flux p has equation-of-state e .
$\forall a [activity_coefficient_model(a)$ \Leftrightarrow $vertex(a) \wedge has_label(a, "activity-coefficient-model *")$ $\wedge \exists p \exists e (phase(p) \wedge edge(e) \wedge has_label(e, "is-modeled-by"))$ $\wedge incident_from(e, p) \wedge incident_to(e, a)]$	Identifies a is an activity coefficient model.
$\forall p \forall a [has_activity_coefficient_model(p, a)$ \Leftrightarrow $phase(p) \wedge activity_coefficient_model(a) \wedge \exists e (edge(e)$ $\wedge has_label(e, "is-modeled-by")) \wedge incident_from(e, p) \wedge incident_to(e, a)]$	Asserts that phase p has activity coefficient model a .
$\forall k [forward_rate_law(k)$ \Leftrightarrow $equation(k) \wedge \exists r \exists e (reaction(r) \wedge edge(e)$ $\wedge has_label(e, "has-forward-kinetics")) \wedge incident_from(e, r) \wedge incident_to(e, k)]$	Identifies k is a forward kinetic rate law for a reaction.
$\forall r \forall k [has_forward_rate_law(r, k)$ \Leftrightarrow $reaction(r) \wedge forward_rate_law(k) \wedge \exists e (edge(e)$ $\wedge has_label(e, "has-forward-kinetics")) \wedge incident_from(e, r) \wedge incident_to(e, k)]$	Asserts that reaction r forward kinetic rate law k .
$\forall k [reverse_rate_law(k)$ \Leftrightarrow $equation(k) \wedge \exists r \exists e (reaction(r) \wedge edge(e)$ $\wedge has_label(e, "has-reverse-kinetics")) \wedge incident_from(e, r) \wedge incident_to(e, k)]$	Identifies k is a reverse kinetic rate law for a reaction.
$\forall r \forall k [has_reverse_rate_law(r, k)$ \Leftrightarrow $reaction(r) \wedge reverse_rate_law(k) \wedge \exists e (edge(e)$ $\wedge has_label(e, "has-reverse-kinetics")) \wedge incident_from(e, r) \wedge incident_to(e, k)]$	Asserts that reaction r reverse kinetic rate law k .

4.4 Model Construction Operators

The model analysis operators defined in the previous section allow high-level assertions to be made regarding a phenomena-based model by examining the state of the underlying model digraph. These declarative operators abstract the details of the underlying representation, and provide the basis for knowledge-level analysis of a phenomena-based model. In a similar manner, it is intuitive to characterize the procedural development of a phenomena-based model as a sequence of hierarchical tasks (e.g., *refine modeled-unit*, *characterize phase behavior*, *define material geometry*, etc.), instead of elementary graph operations. In this section, the tasks comprising the modeling activity that dictate the creation and specification of the phenomena-based model are characterized as procedural operators. These procedural operators differ from the declarative analysis operators defined in the previous section because, when activated, they change the state of the phenomena-based model by modifying the underlying model digraph. These operators are initiated by their preconditions and decisions made by the modeler based on the context of a given engineering problems. In response to these decisions, the operators change the state of the model digraph to automatically generate the underlying MODEL.LA language-based description of the model.

Once deployed in a computer-aided environment, these operators can enable a computer to help the modeler define the model interactively and gradually, provide feedback on the validity of assumptions, capture the rationale of decisions made, and provide an explicit record of the modeling activity. Furthermore, these operators can provide a basis for defining high-level *supervisory logic* operators that integrate context-dependent modeling knowledge that guides the modeling activity based on the goals of an engineering problem.

4.4.1 Modeling Elements

When a new instance of a modeling element in a phenomena-based model is declared, a new vertex labeled with the unique name of the element is added to the model digraph. A new edge labeled “is-a” is then added to the digraph that is incident from the modeling element vertex and incident to a new vertex labeled with a string representing the type (e.g., “flux”) of the modeling element. Figure 4-2 illustrates an example of a new modeled-unit modeling element named “Jacket” added to a model digraph.

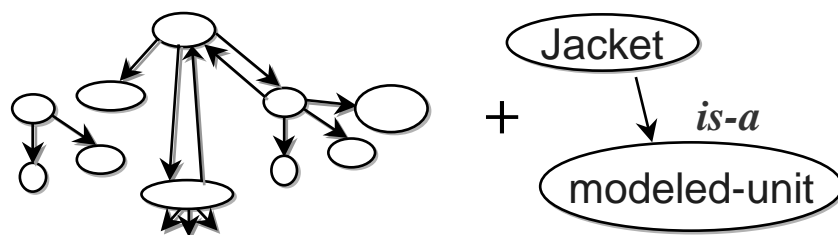


Figure 4-2: New Modeled-Unit Declaration

Several procedural operators that create new instances of modeling elements, which are represented by graph vertices with “is-a” edges incident from them, are defined in Table 4-11.

Table 4-11: Modeling Element Declaration Operators

<i>Operator</i>	<i>Actions</i>
$\forall s_1 \forall s_2 [string(s_1) \wedge string(s_2) \wedge Add_model_element(s_1, s_2)$ \Rightarrow $Add_new_vertex_pair_and_edge(“is-a”, s_1, s_2)]$	Creates modeling element of type s_1 , adds it to the model digraph and labels it with name s_2
$\forall s [string(s) \wedge Add_modeled_unit(s)$ \Rightarrow $Add_model_element(s, “modeled-unit”)]$	Adds a new modeled-unit named s to the model digraph.
$\forall s [string(s) \wedge Add_flux(s)$ \Rightarrow $Add_model_element(s, “flux”)]$	Adds a new flux named s to the model digraph.
$\forall s [string(s) \wedge Add_reaction(s)$ \Rightarrow $Add_model_element(s, “reaction”)]$	Adds a new reaction named s to the model digraph.
$\forall s [string(s) \wedge Add_species(s)$ \Rightarrow $Add_model_element(s, “species”)]$	Adds a new species named s to the model digraph.
$\forall s [string(s) \wedge Add_material_content(s)$ \Rightarrow $Add_model_element(s, “material-content”)]$	Adds a new material-content named s to the model digraph.
$\forall s [string(s) \wedge Add_vapor_phase(s)$ \Rightarrow $Add_model_element(s, “vapor phase”)]$	Adds a new vapor phase named s to the model digraph.
$\forall s [string(s) \wedge Add_liquid_phase(s)$ \Rightarrow $Add_model_element(s, “liquid phase”)]$	Adds a new liquid phase named s to the model digraph.
$\forall s [string(s) \wedge Add_solid_phase(s)$ \Rightarrow $Add_model_element(s, “solid phase”)]$	Adds a new solid phase named s to the model digraph.

The operators in Table 4-11 simply add new instances of modeling elements to the phenomena-based model. To fully define the model, additional operators that introduce semantic relationships that characterize and interrelate these modeling elements must be defined.

4.4.2 Topological Characterization

When a flux is declared to occur between two modeled-units, semantic relationships *to* and *from* identifying the sink and source modeled-unit must be added to the specification of the flux, along with semantic relationship *transports* identifying the type of transport. Furthermore, semantic relationship *has-convective-input*, *has-energy-input*, or *has-species-input* identifying the flux must be added to the specification of the sink modeled-unit, and semantic relationship *has-convective-output*, *has-energy-output*, or *has-species-output* identifying the flux must be added to the specification of the source modeled-unit. The following declarations:

INPUT-OUTPUT-PLANT	<i>is-a</i> modeled-unit <i>has-internal-unit</i> REACTION-SECTION <i>has-internal-unit</i> SEPARATION-SECTION
REACTION-SECTION	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> INPUT-OUTPUT-PLANT
SEPARATION-SECTION	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> INPUT-OUTPUT-PLANT
Source	<i>is-a</i> modeled-unit
Sink	<i>is-a</i> modeled-unit

are illustrated in the conceptual flowsheet shown in Figure 4-3.

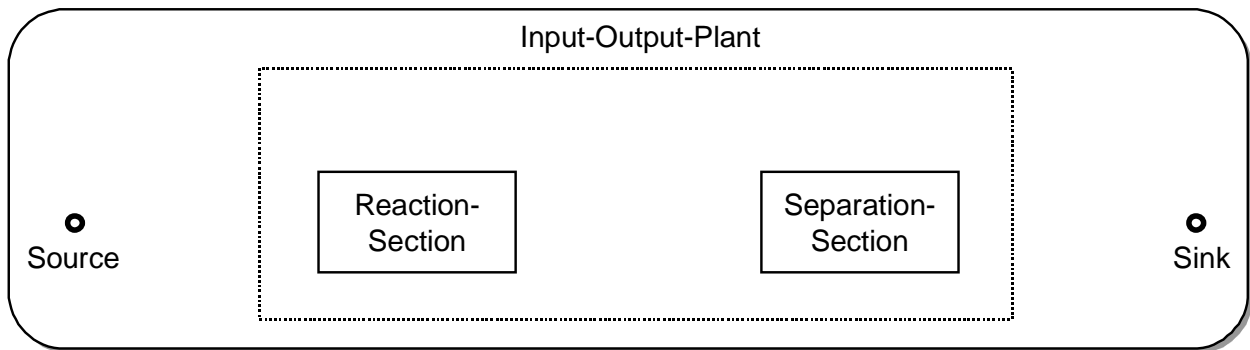


Figure 4-3: Declaration of Topological Structure

If a convective liquid effluent flux is declared between the Reaction-Section and the Separation-Section, as illustrated in Figure 4-4, the modeling element representing the flux is first created, resulting in declaration:

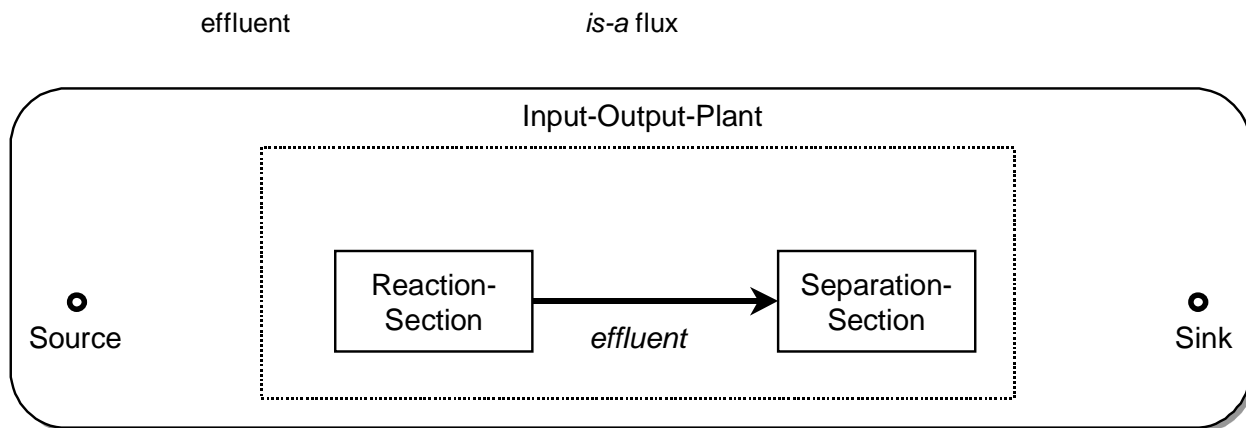


Figure 4-4: Declaration of Internal Flux

The relevant modeling elements are then modified as follows to reflect the connectivity of the flux:

REACTION-SECTION	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> INPUT-OUTPUT-PLANT <i>has-convective-output</i> effluent
SEPARATION-SECTION	<i>is-a</i> modeled-unit <i>is-internal-unit-of</i> INPUT-OUTPUT-PLANT <i>has-convective-input</i> effluent
effluent	<i>is-a</i> flux <i>transports</i> material liquid <i>from</i> REACTION-SECTION <i>to</i> SEPARATION-SECTION

In addition, when the flux crosses the boundary of any ancestor of the sink or source modeled-unit, the appropriate modifications must be made to the specifications of the ancestor modeled-units. For example, If a convective liquid raw-materials flux is declared between the Source and the REACTION-SECTION, as illustrated in Figure 4-5, the resulting declarations are as follows:

INPUT-OUTPUT-PLANT	<i>is-a</i> modeled-unit <i>has-internal-unit</i> REACTION-SECTION <i>has-internal-unit</i> SEPARATION-SECTION <i>has-convective-input</i> raw-materials
REACTION-SECTION	<i>is-a</i> modeled-unit

	<i>is-internal-unit-of</i> INPUT-OUTPUT-PLANT
	<i>has-convective-output</i> effluent
	<i>has-convective-input</i> raw-materials
Source	<i>is-a</i> modeled-unit
	<i>has-convective-output</i> raw-materials
effluent	<i>is-a</i> flux
	<i>transports</i> material liquid
	<i>from</i> Source
	<i>to</i> REACTION-SECTION

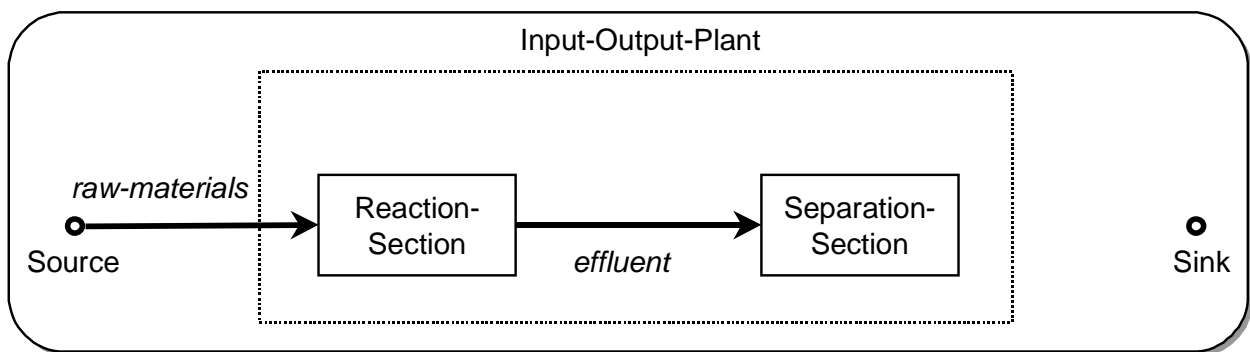


Figure 4-5: Declaration of Multi-Level Flux

In this manner, the topological structure declaration operators, listed in Table 4-12, represent high-level modeling tasks (e.g., *Add_convective_flux*) that abstract the details which refine the state of the model, establish necessary semantic relationships, and enforce topological model consistency.

Table 4-12: Topological Structure Declaration Operators

<u>Operator</u>	<u>Actions</u>
$\forall m_1 \forall m_2 \forall f [\text{modeled_unit}(m_1) \wedge \text{modeled_unit}(m_2) \wedge \text{flux}(f)$ $\wedge \text{Add_convective_flux}(m_1, m_2, f)$ \Rightarrow $\Delta v(\text{vertex}(v)),$ $\text{Add_vertex}(v, \text{"material"}),$ $\text{Add_new_edge}(f, v, \text{"transports"}),$ $\text{Add_new_complementary_edges}(m_1, f, \text{"has-convective-output"}, \text{"from"}),$ $\text{Add_new_complementary_edges}(m_2, f, \text{"has-convective-input"}, \text{"to"}),$ $\forall a [\text{modeled_unit}(a) \wedge \text{has_ancestor}(m_1, a) \wedge \neg \text{has_ancestor}(m_2, a)$ $\Rightarrow \text{Add_new_edge}(a, f, \text{"has-convective-input"}),$ $\forall a [\text{modeled_unit}(a) \wedge \text{has_ancestor}(m_2, a) \wedge \neg \text{has_ancestor}(m_1, a)$ $\Rightarrow \text{Add_new_edge}(a, f, \text{"has-convective-output"})]]$	<ol style="list-style-type: none"> 1. Specifies convective flux f from modeled-unit m_1 to modeled-unit m_2. 2. Adds input flux to ancestors of m_1 where appropriate. 3. Adds output flux to ancestors of m_2 where appropriate.
$\forall m_1 \forall m_2 \forall f [\text{modeled_unit}(m_1) \wedge \text{modeled_unit}(m_2) \wedge \text{flux}(f)$ $\wedge \text{Add_energy_flux}(m_1, m_2, f)$ \Rightarrow $\Delta v(\text{vertex}(v)),$ $\text{Add_vertex}(v, \text{"energy"}),$ $\text{Add_new_edge}(f, v, \text{"transports"}),$ $\text{Add_new_complementary_edges}(m_1, f, \text{"has-energy-output"}, \text{"from"}),$ $\text{Add_new_complementary_edges}(m_2, f, \text{"has-energy-input"}, \text{"to"})$ $\forall a [\text{modeled_unit}(a) \wedge \text{has_ancestor}(m_1, a) \wedge \neg \text{has_ancestor}(m_2, a)$ $\Rightarrow \text{Add_new_edge}(a, f, \text{"has-energy-input"}),$ $\forall a [\text{modeled_unit}(a) \wedge \text{has_ancestor}(m_2, a) \wedge \neg \text{has_ancestor}(m_1, a)$ $\Rightarrow \text{Add_new_edge}(a, f, \text{"has-energy-output"})]]$	<ol style="list-style-type: none"> 1. Specifies energy flux f from modeled-unit m_1 to modeled-unit m_2. 2. Adds input flux to ancestors of m_1 where appropriate. 3. Adds output flux to ancestors of m_2 where appropriate.
$\forall m_1 \forall m_2 \forall f \forall s [\text{modeled_unit}(m_1) \wedge \text{modeled_unit}(m_2) \wedge \text{flux}(f) \wedge \text{species}(s)$ $\wedge \text{Add_species_flux}(m_1, m_2, f, s)$ \Rightarrow $\Delta v(\text{vertex}(v)),$ $\Delta t(\text{string}(t)),$ $t := \text{"species"} + \text{label}(s),$ $\text{Add_vertex}(v, t),$ $\text{Add_new_edge}(f, v, \text{"transports"}),$ $\text{Add_new_complementary_edges}(m_1, f, \text{"has-species-output"}, \text{"from"}),$ $\text{Add_new_complementary_edges}(m_2, f, \text{"has-species-input"}, \text{"to"}),$ $\forall a [\text{modeled_unit}(a) \wedge \text{has_ancestor}(m_1, a) \wedge \neg \text{has_ancestor}(m_2, a)$ $\Rightarrow \text{Add_new_edge}(a, f, \text{"has-species-input"}),$ $\forall a [\text{modeled_unit}(a) \wedge \text{has_ancestor}(m_2, a) \wedge \neg \text{has_ancestor}(m_1, a)$ $\Rightarrow \text{Add_new_edge}(a, f, \text{"has-species-output"})]]$	<ol style="list-style-type: none"> 1. Specifies species s flux f from modeled-unit m_1 to modeled-unit m_2. 2. Adds input flux to ancestors of m_1 where appropriate. 3. Adds output flux to ancestors of m_2 where appropriate.
$\forall m \forall p \forall f [\text{material_content}(m) \wedge \text{allocated_element}(p) \wedge \text{flux}(f)$ $\wedge \text{Allocate_flux}(m, p, f)$ \Rightarrow $\text{Add_new_edge}(m, f, \text{"has-boundary-flux"}),$ $\text{Add_new_edge}(f, p, \text{"allocated-to"})]]$	<p>Allocates boundary flux f of material-content m to modeling element p.</p>

4.4.3 Chemical Content

The assumption that a chemical species is present in a modeled-unit, material-content, or phase in a phenomena-based model requires that the semantic relationship *has-species* identifying the species be added to the specification of the corresponding modeling element. Similarly, the assumption that a chemical reaction is present in a modeled-unit or phase requires that the semantic relationship *has-reaction* identifying the reaction be added to the specification of the corresponding modeled-unit or phase. Additionally, for consistency, when a species is declared to be present in a phase, it must also be present in the associated material-content, the modeled-unit containing the material-content, and all ancestors of the modeled-unit. This is required so that the phenomena-based model, viewed at any level of detail, will be consistent with all other levels. A similar hierarchical consistency is required for the declaration of chemical reactions. Chemical content declaration operators that make these desired modifications are listed in Table 4-13.

Use of these operators will be illustrated by the following declarations:

SEPARATION_SECTION	<i>is-a</i> modeled-unit <i>has-internal-unit</i> FLASH <i>has-internal-unit</i> DISTILLATION_TRAIN
FLASH	<i>is-a</i> modeled-unit <i>has-material-content</i> FLASH_MATL
FLASH_MATL	<i>is-a</i> material-content <i>is-material-content-in</i> FLASH <i>has-vapor-phase</i> FLASH_MATL_V <i>has-liquid-phase</i> FLASH_MATL_L
FLASH_MATL_V	<i>is-a</i> vapor phase <i>is-phase-in</i> FLASH_MATL
FLASH_MATL_L	<i>is-a</i> liquid phase <i>is-phase-in</i> FLASH_MATL

Table 4-13: Chemical Content Declaration Operators

<u>Operator</u>	<u>Actions</u>
$\forall m \forall r [\text{modeled-unit}(m) \wedge \text{reaction}(r) \wedge \text{Add_reaction}(m, r)$ \Rightarrow $\text{Add_new_edge}(m, r, \text{"has-reaction"}),$ $\forall a [\text{modeled_unit}(a) \wedge \text{has_ancestor}(m, a) \wedge \neg \text{has_reaction}(a, r)$ \Rightarrow $\text{Add_reaction}(a, r)]]$	<ol style="list-style-type: none"> 1. Adds reaction <i>r</i> to modeled-unit <i>m</i>. 2. Adds <i>r</i> to ancestors of <i>m</i> where appropriate.
$\forall p \forall r [\text{phase}(p) \wedge \text{reaction}(r) \wedge \text{Add_reaction}(p, r)$ \Rightarrow $\text{Add_new_edge}(p, r, \text{"has-reaction"}),$ $\exists m \exists u [\text{material_content}(m) \wedge \text{modeled_unit}(u) \wedge \text{has_phase}(m, p)$ $\wedge \text{has_material_content}(u, m) \wedge \neg \text{has_reaction}(u, r)$ \Rightarrow $\text{Add_reaction}(u, r)]]$	<ol style="list-style-type: none"> 1. Adds reaction <i>r</i> to phase <i>p</i>. 2. Adds <i>r</i> to modeled-unit containing material-content associated with <i>p</i> when necessary.
$\forall u \forall s [\text{modeled-unit}(u) \wedge \text{species}(s) \wedge \text{Add_species}(u, s)$ \Rightarrow $\text{Add_new_edge}(u, s, \text{"has-species"}),$ $\forall a [\text{modeled_unit}(a) \wedge \text{has_ancestor}(u, a) \wedge \neg \text{has_species}(a, s)$ \Rightarrow $\text{Add_species}(a, s),$ $\exists m [\text{material_content}(m) \wedge \text{has_material_content}(u, m)$ $\wedge \neg \text{has_species}(m, s)$ \Rightarrow $\text{Add_species}(m, s)]]$	<ol style="list-style-type: none"> 1. Adds species <i>s</i> to modeled-unit <i>m</i>. 2. Adds <i>s</i> to ancestors of <i>m</i> where appropriate. 3. Adds <i>s</i> to material-content of <i>m</i> if when necessary.
$\forall m \forall s [\text{material_content}(m) \wedge \text{species}(s) \wedge \text{Add_species}(m, s)$ \Rightarrow $\text{Add_new_edge}(m, s, \text{"has-species"}),$ $\exists u [\text{modeled_unit}(u) \wedge \text{has_material_content}(u, m)$ $\wedge \neg \text{has_species}(u, s)$ \Rightarrow $\text{Add_species}(u, s)]]$	<ol style="list-style-type: none"> 1. Adds species <i>s</i> to material-content <i>m</i>. 2. Adds <i>s</i> to modeled-unit associated with <i>m</i> where appropriate..
$\forall p \forall r [\text{phase}(p) \wedge \text{species}(s) \wedge \text{Add_species}(m, s)$ \Rightarrow $\text{Add_new_edge}(p, s, \text{"has-reaction"}),$ $\exists m [\text{material_content}(m) \wedge \text{has_phase}(m, p) \wedge \neg \text{has_species}(m, s)$ \Rightarrow $\text{Add_species}(m, s)]]$	<ol style="list-style-type: none"> 1. Adds species <i>s</i> to phase <i>p</i>. 2. Adds <i>s</i> to material-content associated with <i>p</i> when necessary.

The operation *Add_species*(FLASH_MATL_V, BENZENE) will result in declarations:

SEPARATION_SECTION	<i>is-a</i> modeled-unit
	<i>has-internal-unit</i> FLASH
	<i>has-internal-unit</i> DISTILLATION_TRAIN
	<i>has-species</i> BENZENE
FLASH	<i>is-a</i> modeled-unit

	<i>has-material-content</i> FLASH_MATL
	<i>has-species</i> BENZENE
FLASH_MATL	<i>is-a</i> material-content
	<i>is-material-content-in</i> FLASH
	<i>has-vapor-phase</i> FLASH_MATL_V
	<i>has-liquid-phase</i> FLASH_MATL_L
	<i>has-species</i> BENZENE
FLASH_MATL_V	<i>is-a</i> vapor phase
	<i>is-phase-in</i> FLASH_MATL
	<i>has-species</i> BENZENE
FLASH_MATL_L	<i>is-a</i> liquid phase
	<i>is-phase-in</i> FLASH_MATL

For consistency, note that not only is the specification of FLASH_MATL_V modified, but also the specifications of FLASH_MATL, FLASH, and SEPARATION_SECTION. However, after application of these operators, inconsistencies may still be present. For example, the operation *Add_species*(FLASH, TOLUENE) would leave the model inconsistent, as no phase in the material-content would contain the species TOLUENE. An additional declaration would be required by the modeler that made such an assignment. If it were not made, such an inconsistency would be detected by a model inconsistency operator, as defined in the subsequent section.

4.4.4 Hierarchical Characterization

The decomposition of a composite modeled-unit into a set of more refined modeled-units, or the aggregation of a set of modeled-units into an abstract modeled-unit, requires that complementary semantic relationships *has-internal-unit* and *is-internal-unit-of* identifying the necessary interrelations be added to the specifications of the corresponding modeled-units. Additionally, for consistency, all species and reactions in the subunits must also be present in the abstract composite unit. Any non-internal fluxes crossing the boundary of the abstract unit must also added to the specification of the composite unit. Hierarchical structure characterization operators that make these desired modifications are listed in Table 4-14.

Table 4-14: Hierarchical Structure Characterization Operators

<u>Operator</u>	<u>Actions</u>
$\forall p \forall s [modeled_unit(p) \wedge modeled_unit(s) \wedge Add_subunit(p, s)$ \Rightarrow $Add_new_complementary_edges$ $(p, s, "has-internal-unit", "is-internal-unit-of"),$ $\forall a [has_species(s, a) \wedge \neg has_species(p, a)$ $\Rightarrow Add_species(p, a)]$ $\forall r [has_reaction(s, r) \wedge \neg has_reaction(p, r)$ $\Rightarrow Add_reaction(p, r)]]$	<ol style="list-style-type: none"> 1. Specifies s is a subunit of modeled-unit p. 2. Adds species in s to p when necessary. 3. Adds reactions in s to p when necessary.
$\forall p \forall s_1 \forall s_2 \dots \forall s_N [modeled_unit(p) \wedge modeled_unit(s_1)$ $\wedge modeled_unit(s_2) \wedge \dots \wedge modeled_unit(s_N)$ $\wedge Abstract_subunits(p, s_1, s_2, \dots, s_N)$ \Rightarrow $S = \{s_1, s_2, \dots, s_N\},$ $\forall m [m \in S \Rightarrow Add_subunit(p, m)],$ $\forall f \forall m [flux(f) \wedge m \in S \wedge has_input(m, f) \wedge \neg \exists n (n \in S \wedge has_output(n, f))$ \Rightarrow $convective_flux(f)$ $\Rightarrow Add_new_edge(m, f, "has-convective-input"),$ $energy_flux(f)$ $\Rightarrow Add_new_edge(m, f, "has-energy-input"),$ $species_flux(f)$ $\Rightarrow Add_new_edge(m, f, "has-species-input")]$ $\forall f \forall m [flux(f) \wedge m \in S \wedge has_output(m, f) \wedge \neg \exists n (n \in S \wedge has_input(n, f))$ \Rightarrow $convective_flux(f)$ $\Rightarrow Add_new_edge(m, f, "has-convective-output"),$ $energy_flux(f)$ $\Rightarrow Add_new_edge(m, f, "has-energy-output"),$ $species_flux(f)$ $\Rightarrow Add_new_edge(m, f, "has-species-output")]$	<ol style="list-style-type: none"> 1. Activates previous operator to add set of subunits to parent unit. 2. Identifies fluxes entering abstract parent system. 3. Identifies fluxes leaving abstract parent system.
$\forall p \forall s [distributed_unit(p) \wedge modeled_unit(s)$ $\wedge Add_differential_subunit(p, s)$ \Rightarrow $Add_new_complementary_edges$ $(p, s, "has-differential-subunit", "is-internal-unit-of"),$ $\forall a [has_species(s, a) \wedge \neg has_species(p, a)$ $\Rightarrow Add_species(p, a)]$ $\forall r [has_reaction(s, r) \wedge \neg has_reaction(p, r)$ $\Rightarrow Add_reaction(p, r)]]$	<ol style="list-style-type: none"> 1. Specifies s is a differential subunit of spatially distributed modeled-unit p. 2. Adds species in s to p when necessary. 3. Adds reactions in s to p when necessary.

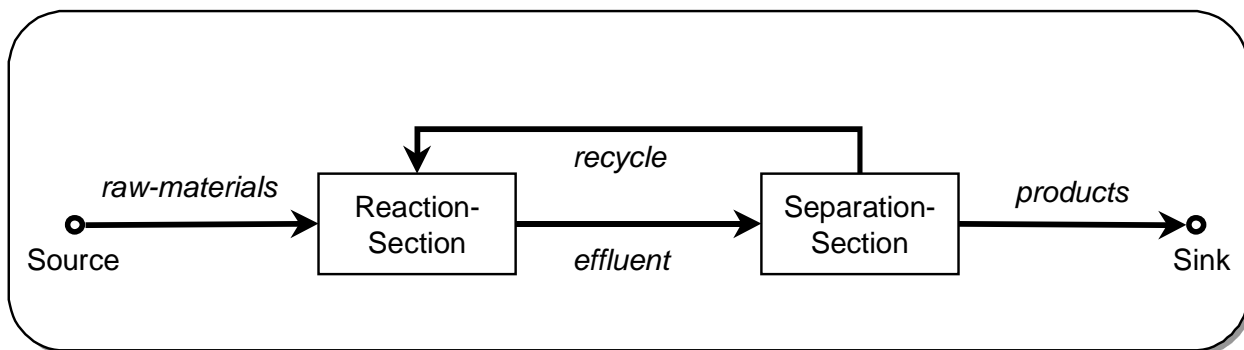


Figure 4-6: Hierarchical Structure Characterization Example

Use of these operators will be illustrated by the following declarations, which are illustrated in Figure 4-6:

REACTION-SECTION	<i>is-a</i> modeled-unit <i>has-convective-input</i> raw-materials <i>has-convective-input</i> recycle <i>has-convective-output</i> effluent <i>has-reaction</i> RXN_101 <i>has-species</i> A <i>has-species</i> B <i>has-species</i> C
SEPARATION_SECTION	<i>is-a</i> modeled-unit <i>has-convective-input</i> effluent <i>has-convective-output</i> products <i>has-convective-output</i> recycle <i>has-species</i> A <i>has-species</i> B <i>has-species</i> C

The operations *Add_modeled_unit*("INPUT-OUTPUT-PLANT") and *Abstract_subunits*(INPUT-OUTPUT-PLANT, REACTION-SECTION, SEPARATION-SECTION) will result in following declarations, illustrated in Figure 4-7:

INPUT-OUTPUT-PLANT	<i>is-a</i> modeled-unit <i>has-internal-unit</i> REACTION-SECTION <i>has-internal-unit</i> SEPARATION-SECTION <i>has-convective-input</i> raw-materials
--------------------	---

	<i>has-convective-output</i> product
	<i>has-reaction</i> RXN_101
	<i>has-species</i> A
	<i>has-species</i> B
	<i>has-species</i> C
REACTION-SECTION	<i>is-a</i> modeled-unit
	<i>is-internal-unit-of</i> INPUT-OUTPUT-PLANT
	<i>has-convective-input</i> raw-materials
	<i>has-convective-input</i> recycle
	<i>has-convective-output</i> effluent
	<i>has-reaction</i> RXN_101
	<i>has-species</i> A
	<i>has-species</i> B
	<i>has-species</i> C
SEPARATION-SECTION	<i>is-a</i> modeled-unit
	<i>is-internal-unit-of</i> INPUT-OUTPUT-PLANT
	<i>has-convective-input</i> effluent
	<i>has-convective-output</i> products
	<i>has-convective-output</i> recycle
	<i>has-species</i> A
	<i>has-species</i> B
	<i>has-species</i> C

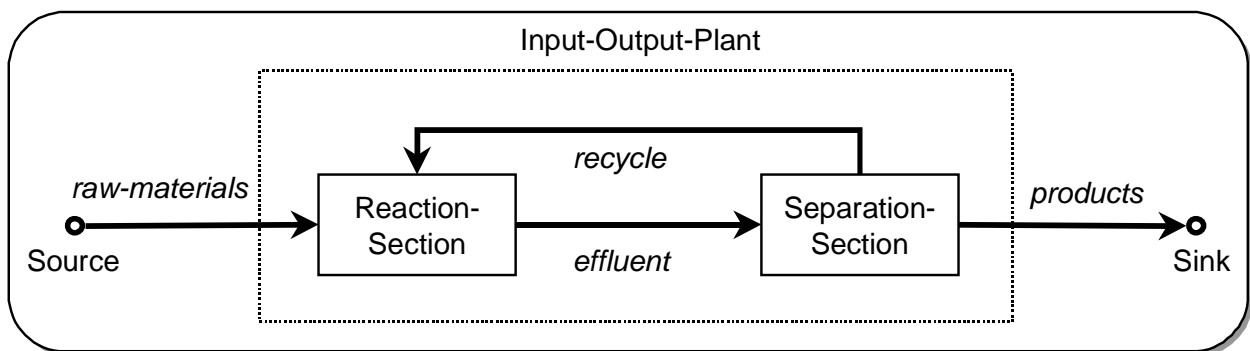


Figure 4-7: Hierarchical Abstraction Example

Whenever possible, the modeling operators propagate assumptions throughout the hierarchical structure of the phenomena-based model. When modeled-units are aggregated into an abstract composite system, a consistent representation may be made automatically. However, when a

modeled-unit is disaggregated into a set of modeled-units, additional declarations are required by the modeler that allocate the species, reactions, and fluxes of the abstract unit to its subunits.

4.4.5 Material Characterization

When a material-content is associated with a modeled-unit, the complementary semantic relationships *has-material-content* and *is-material-content-of* interrelating the modeled-units must be added to the corresponding specifications. Additional assignments of chemical species and boundary fluxes may also be made for consistency. When a phase is associated with a material-content, the semantic relationship *has-vapor-phase*, *has-liquid-phase*, or *has-solid-phase* identifying the phase must be added to the specification of the material-content, and the semantic relationship *is-phase-in* identifying the material-content must be added to the specification of the phase. Additional assignments of chemical species and reactions may also be made for consistency. Declaration of a vessel geometry for a material-content requires addition of the semantic relationship *has-vessel-geometry* identifying the type of geometry to the specification of the material-content. Material characterization operators that make these desired modifications are listed in Table 4-15.

Use of these operators will be illustrated by the following declarations:

```
FLASH          is-a modeled-unit
               has-species BENZENE
               has-species TOLUENE
               has-convective-input feed
               has-convective-output overhead
               has-convective-output bottoms
```

The operations *Add_material_content*("FLASH_MATL"), *Add_material_content*(FLASH, FLASH_MATL), *Add_vapor_phase*("FLASH_MATL_V"), *Add_phase*(FLASH_MATL, FLASH_MATL_V), *Add_liquid_phase*("FLASH_MATL_L"), and *Add_phase*(FLASH_MATL, FLASH_MATL_L) will result in the following declarations:

Table 4-15: Material Characterization Operators

<u>Operator</u>	<u>Actions</u>
$\forall u \forall m [modeled_unit(u) \wedge material_content(m)$ $\wedge Add_material_content(u, m)$ \Rightarrow $Add_new_complementary_edges$ $(u, m, "has-material-content", "is-material-content-of"),$ $\forall s [has_species(u, s) \wedge \neg has_species(m, s)$ \Rightarrow $Add_species(m, s),$ $\forall s [has_species(m, s) \wedge \neg has_species(u, s)$ \Rightarrow $Add_species(u, s),$ $\forall f [flux(f) \wedge (has_input_flux(u, f) \vee has_output_flux(u, f))$ \Rightarrow $Allocate_flux(m, f, m)]]$	<ol style="list-style-type: none"> 1. Specifies m is material-content of modeled-unit u. 2. Adds species in u to m if necessary. 3. Adds species in m to u if necessary. 4. Adds boundary fluxes of u to m.
$\forall m \forall p [material_content(m) \wedge phase(p) \wedge Add_phase(m, p)$ \Rightarrow $vapor_phase(p) \Rightarrow Add_new_edge(m, p, "has-vapor-phase"),$ $liquid_phase(p) \Rightarrow Add_new_edge(m, p, "has-liquid-phase"),$ $solid_phase(p) \Rightarrow Add_new_edge(m, p, "has-solid-phase"),$ $Add_new_edge(p, m, "is-phase-in"),$ $\forall s [has_species(p, s) \wedge \neg has_species(m, s)$ \Rightarrow $Add_species(m, s)]]$ $\exists u [modeled_unit(u) \wedge has_material_content(u, m)$ $\wedge \neg has_reaction(u, r)$ \Rightarrow $Add_reaction(u, r)]]$	<ol style="list-style-type: none"> 1. Specifies p is phase in material-content m. 2. Adds species in p to m if necessary. 3. Adds reactions in p to modeled-unit associated with m if necessary.
$\forall s \forall m [string(s) \wedge material_content(m) \wedge Add_geometry(m, s)$ \Rightarrow $\Delta v(vertex(v)),$ $label(v) := s,$ $Add_new_edge(m, v, "has-vessel-geometry")]$	<p>Specifies geometry of material-content m.</p>

FLASH

is-a modeled-unit

has-species BENZENE

has-species TOLUENE

has-convective-input feed

has-convective-output overhead

has-convective-output bottoms

FLASH_MATL

is-a material-content

is-material-content-in FLASH

```

                                has-vapor-phase FLASH_MATL_V
                                has-liquid-phase FLASH_MATL_L
                                has-boundary-flux feed allocated-to self
                                has-boundary-flux overhead allocated-to self
                                has-boundary-flux bottoms allocated-to self
                                has-species BENZENE
                                has-species TOLUENE

FLASH_MATL_V                    is-a vapor phase
                                is-phase-in FLASH_MATL

FLASH_MATL_L                    is-a liquid phase
                                is-phase-in FLASH_MATL

```

Note that additional declarations must be made by the modeler allocating the chemical species and the outgoing convective fluxes to the phases of the material-content.

4.4.6 Mechanistic Characterization

Mechanistic characterizations in a phenomena-based model regarding transport mechanisms of fluxes and thermodynamic characterizations of phases require that the semantic relationship *is-modeled-by* identifying the assumed mechanism be added to the specifications of the corresponding modeling elements. Kinetic rate laws of reactions require specification of rate equations using semantic relationships *has-forward-kinetics* and *has-reverse-kinetics*. Mechanistic characterization operators that make these desired modifications are listed in Table 4-16.

Use of these operators will be illustrated by the following declarations:

```

FLASH_MATL_V                    is-a vapor phase
                                is-phase-in FLASH_MATL
                                has-species BENZENE
                                has-species TOLUENE

```

The operation *Specify_equation_of_state*(FLASH_MATL_V, “redlich-kwong”) will result in the following declarations:

```

FLASH_MATL_V                    is-a vapor phase
                                is-phase-in FLASH_MATL
                                has-species BENZENE

```


has-species TOLUENE

is-modeled-by equation-of-state redlich-kwong

Table 4-16: Mechanistic Characterization Operators

<u>Operator</u>	<u>Actions</u>
$\forall f \forall s [flux(f) \wedge string(s) \wedge Specify_mechanism(f, s)$ \Rightarrow $\Delta v(vertex(v)),$ $\Delta t(string(t)),$ $t := "transport-mechanism" + s,$ $Add_vertex(v, t),$ $Add_new_edge(f, v, "is-modeled-by")]$	Specifies transport mechanism of flux f .
$\forall p \forall s [(phase(p) \vee convective_flux(p)) \wedge string(s)$ $\wedge Specify_equation_of_state(p, s)$ \Rightarrow $\Delta v(vertex(v)),$ $\Delta t(string(t)),$ $t := "equation-of-state" + s,$ $Add_vertex(v, t),$ $Add_new_edge(f, v, "is-modeled-by")]$	Specifies equation of state for phase or convective flux p .
$\forall p \forall s [phase(p) \wedge string(s) \wedge Specify_activity_coefficient_model(p, s)$ \Rightarrow $\Delta v(vertex(v)),$ $\Delta t(string(t)),$ $t := "activity-coefficient-model" + s,$ $Add_vertex(v, t),$ $Add_new_edge(f, v, "is-modeled-by")]$	Specifies activity coefficient model of phase p .
$\forall r \forall e [reaction(r) \wedge equation(e) \wedge Specify_forward_kinetics(r, e) \Rightarrow$ $Add_new_edge(r, e, "has-forward-kinetics")]$	Specifies forward kinetic rate law of reaction r .
$\forall r \forall e [reaction(r) \wedge equation(e) \wedge Specify_reverse_kinetics(r, e) \Rightarrow$ $Add_new_edge(r, e, "has-reverse-kinetics")]$	Specifies reverse kinetic rate law of reaction r .

4.4.7 Behavioral Characterization

Behavioral characterizations of modeled-units and materials in a phenomena-based model require that the semantic relationship *is-modeled-as* identifying the assumed behavior be added to the specifications of the corresponding modeling elements. Behavioral characterization operators that make these desired modifications are listed in Table 4-17.

Use of these operators will be illustrated by the following declarations:

FLASH_MATL

is-a material-content

is-material-content-in FLASH

has-vapor-phase FLASH_MATL_V

has-liquid-phase FLASH_MATL_L

The operation *Specify_behavior*(FLASH_MATL, “constant-pressure”) will result in the following declarations:

FLASH_MATL *is-a* material-content
 is-material-content-in FLASH
 has-vapor-phase FLASH_MATL_V
 has-liquid-phase FLASH_MATL_L
 is-modeled-as constant-pressure

Table 4-17: Behavioral Characterization Operators

<u>Operator</u>	<u>Actions</u>
$\forall m \forall s [modeled_unit(m) \wedge string(s) \wedge Specify_behavior(m, s)$ \Rightarrow $\Delta v(vertex(v)),$ $Add_vertex(v, s),$ $Add_new_edge(m, v, "is-modeled-as")]$	Specifies a behavioral assumption for modeled-unit <i>m</i> .
$\forall m \forall s [material_content(m) \wedge string(s) \wedge Specify_behavior(m, s)$ \Rightarrow $\Delta v(vertex(v)),$ $Add_vertex(v, s),$ $Add_new_edge(m, v, "is-modeled-as")]$	Specifies a behavioral assumption for material-content <i>m</i> .

4.5 Model Consistency Operators

Before a mathematical model is derived from the physicochemical description, the consistency of the phenomena-based model should be verified. An inconsistency is detected when logical operators make conflicting assertions about the state of the phenomena-based model. For example, consider the following logical statements:

$$\forall x [A(x) \Rightarrow B(x)]$$

$$\forall x [C(x) \Leftrightarrow B(x)]$$

If, for a given *x*, *A(x)* is true and *C(x)* is false, then a logical inconsistency exists since *B(x)* (asserted by the first implication) and $\neg B(x)$ (asserted by the second if-and-only-if implication)

cannot be true simultaneously.

The most readily ascertained inconsistencies in a phenomena-based model description involve the hierarchical and topological allocation of fluxes, chemical species, and chemical reactions in the model. In this section, operators that express necessary conditions for the state of a phenomena-based model will be described. When these operators make assertions that conflict with the hierarchical structure, topological structure, material-content, and chemical content analysis operators defined in an earlier section, the existence of modeling inconsistencies is detected. In a computer-aided environment, these operators may be used to provide feedback to the modeler on the consistency of the phenomena-based model description, and possible alternatives to resolve the problem. As an extension, additional operators may be defined that invoke “chemical engineering judgement” for reviewing model simulation results in order to evaluate and critique mechanistic assumptions made during model development.

4.5.1 Hierarchical Consistency

The hierarchical structure of a phenomena-based model allows the model to be viewed at multiple levels of detail, where composite systems may be viewed as abstract control volumes or as aggregates of more refined control volumes. For conservation principles to be expressed consistently among the varying levels of detail, certain conditions must be true. Several rules for the hierarchical consistency of a phenomena-based model are listed in Table 4-18. These operators are applied iteratively throughout the hierarchical tree of modeled-units, ensuring consistency among all levels of modeling detail.

The first operator in Table 4-18 states that any chemical species assumed to be present in an composite parent unit must also be assumed to be present at least one subunit of the parent. Conversely, the second operator states that any chemical species assumed to be present in a subunit must also be present in the parent unit. Essentially, these operators state that the set of species assumed for the parent unit must equal the union of the sets of species assumed for each of the subunits, or

$$S = \bigcup_{\text{subunits}} S_i$$

where

1. S is the set of species assumed to be present in the composite unit, and

2. S_i is the set of species assumed to be present in the i^{th} subunit of the composite unit.

Similarly, in the case of chemical reactions, the third and fourth operators state that

$$R = \bigcup_{\text{subunits}} R_i$$

where

1. R is the set of reactions assumed to occur in the composite unit, and
2. R_i is the set of reactions assumed to occur in the i^{th} subunit of the composite unit.

These rules enforce that consistent conservation equations are derived, regardless of the level of abstraction or refinement.

The fifth and sixth operators in Table 4-18 state that any flux entering (or leaving) the boundary of a composite unit must enter (or leave) the boundary of one of the subunits of the composite unit. Conversely, the seventh and eighth operators state that any flux entering (or leaving) the boundary of a subunit of a composite unit that does not leave (or enter) the boundary of another subunit of the composite unit (i.e., it is not an internal flux) must enter (or leave) the boundary of the composite unit. The final related operator states that any internal flux between two subunits of a composite unit cannot be a boundary flux of the composite unit. Essentially, these rules state that

$$F^{in} = \left(\bigcup_{\text{subunits}} F_i^{in} \right) - \left(\bigcup_{\text{subunits}} F_i^{in} \right) \cap \left(\bigcup_{\text{subunits}} F_i^{out} \right)$$

and

$$F^{out} = \left(\bigcup_{\text{subunits}} F_i^{out} \right) - \left(\bigcup_{\text{subunits}} F_i^{out} \right) \cap \left(\bigcup_{\text{subunits}} F_i^{in} \right)$$

where

1. F^{in} is the set of fluxes assumed to enter the boundaries of the composite unit,
2. F^{out} is the set of fluxes assumed to leave the boundaries of the composite unit,
3. F_i^{in} is the set of fluxes assumed to enter the boundaries of the i^{th} subunit of the composite unit, and
4. F_i^{out} is the set of fluxes assumed to leave the boundaries of the i^{th} subunit of the composite unit.

These rules enforce that the net flux of a conserved quantity into (or out of) a composite system

will equal the sum of the net fluxes of the conserved quantity into (or out of) each of its subunits.

If a phenomena-based model were to be viewed as a static entity, these hierarchical consistency operators might seem irrelevant, since this abstraction is a many-to-one mapping that can be automated. For example, why maintain the species and reaction assignment assumptions for the parent unit separately, when they may be inferred as needed from the species and reaction assignment assumptions for the subunits? Similarly, why maintain the boundary flux assumptions when they may similarly be inferred from the boundary fluxes of the subunits? However, model development is in fact an evolutionary process. During hierarchical model development of a complex process, modelers can readily overlook these details, neglecting to carry over assumptions (such as the presence of species and reactions) from one level to the next refined level. By tracking and maintaining these assumptions at each level independently, such oversights may be detected and pointed out to the modeler for rectification.

Table 4-18: Hierarchical Consistency Operators

<u>Operator</u>	<u>Actions</u>
$\forall m \forall s [modeled_unit(m) \wedge species(s) \wedge has_species(m, s) \wedge has_subunits(m) \Rightarrow \exists x (has_subunit(m, x) \wedge has_species(x, s))]$	Verifies that all species assigned to a parent unit are also assigned to at least one subunit of the parent unit.
$\forall m \forall p \forall s [modeled_unit(m) \wedge modeled_unit(p) \wedge species(s) \wedge has_subunit(p, m) \wedge has_species(m, s) \Rightarrow has_species(p, s)]$	Verifies that all species assigned to any subunit of a parent unit are also assigned to the parent unit.
$\forall m \forall r [modeled_unit(m) \wedge reaction(r) \wedge has_reaction(m, r) \wedge has_subunits(m) \Rightarrow \exists x (has_subunit(m, x) \wedge has_reaction(x, r))]$	Verifies that all reaction assigned to a parent unit are also assigned to at least one subunit of the parent unit.
$\forall m \forall p \forall r [modeled_unit(m) \wedge modeled_unit(p) \wedge reaction(r) \wedge has_subunit(p, m) \wedge has_reaction(m, r) \Rightarrow has_reaction(p, r)]$	Verifies that all reactions assigned to any subunit of a parent unit are also assigned to the parent unit.
$\forall m \forall f [modeled_unit(m) \wedge flux(f) \wedge has_input_flux(m, f) \wedge has_subunits(m) \Rightarrow \exists s (has_subunit(m, s) \wedge has_input_flux(s, f))]$	Verifies that any flux to a parent unit is also a flux to a subunit of the parent unit.
$\forall m \forall f [modeled_unit(m) \wedge flux(f) \wedge has_output_flux(m, f) \wedge has_subunits(m) \Rightarrow \exists s (has_subunit(m, s) \wedge has_output_flux(s, f))]$	Verifies that any flux from a parent unit is also a flux from a subunit of the parent unit.
$\forall f \forall s \forall m \forall p [flux(f) \wedge modeled_unit(s) \wedge modeled_unit(m) \wedge modeled_unit(p) \wedge from(f, s) \wedge to(f, m) \wedge has_subunit(p, s) \wedge \neg has_subunit(p, m) \Rightarrow has_output_flux(p, f)]$	Verifies that any flux from a subunit of a parent unit to a unit that is not a subunit of the parent unit is also a flux from the parent unit.
$\forall f \forall s \forall m \forall p [flux(f) \wedge modeled_unit(s) \wedge modeled_unit(m) \wedge modeled_unit(p) \wedge from(f, m) \wedge to(f, s) \wedge has_subunit(p, s) \wedge \neg has_subunit(p, m) \Rightarrow has_input_flux(p, f)]$	Verifies that any flux to a subunit of a parent unit to a unit that is not a subunit of the parent unit is also a flux to the parent unit.
$\forall f \forall p \forall s_1 \forall s_2 [flux(f) \wedge modeled_unit(p) \wedge modeled_unit(s_1) \wedge modeled_unit(s_2) \wedge from(f, s_1) \wedge to(f, s_2) \wedge has_subunit(p, s_1) \wedge has_subunit(p, s_2) \Rightarrow \neg has_input_flux(p, f) \wedge \neg has_output_flux(p, f)]$	Verifies that any flux between two subunits of a parent unit is not a flux to or from the parent unit.

4.5.2 Material Characterization Consistency

The conditions for consistency of the material characterization of a phenomena-based model are related to those for hierarchical consistency. The associated rules are listed in Table 4-19.

The first two operators in Table 4-19 states that any chemical species assumed to be present in an material unit (i.e., a modeled-unit with a material-content) must also be present in its material-content and that, conversely, and chemical species assumed to be present in the material-content must also be present in the associated modeled-unit. Essentially, these operators state that

$$S^u = S$$

where

1. S^u is the set of species assumed to be present in the material unit, and
2. S is the set of species assumed to be present in material-content of the material unit.

The third operator in Table 4-19 states that any chemical species assumed to be present in an material-content must also be assumed to be present at least one phase of the material-content. Conversely, the second operator states that any chemical species assumed to be present in a phase must also be present in the material-content. Essentially, these operators state that

$$S = \bigcup_{\text{phases}} S_i$$

where

1. S is the set of species assumed to be present in the material-content, and
2. S_i is the set of species assumed to be present in the i^{th} phase of the material-content.

Similarly, in the case of chemical reactions, the fifth and sixth operators state that

$$R = \bigcup_{\text{phases}} R_i$$

where

1. R is the set of reactions assumed to occur in the modeled-unit associated with the material-content, and
2. R_i is the set of reactions assumed to occur in the i^{th} phase of the material-content.

The seventh and eighth operators state that any flux entering or leaving the boundaries of a material unit must be a boundary flux of its material-content. Essentially,

$$B = F^{in} \cup F^{out}$$

where

1. B is the set of boundary fluxes of a material-content,
2. F^{in} is the set of fluxes assumed to enter the boundaries of the associated modeled-unit, and
3. F^{out} is the set of fluxes assumed to leave the boundaries of the associated modeled-unit.

The above material characterization consistency operators may be viewed as extensions to the hierarchical consistency operators, where a material-content is viewed as the single subsystem of the modeled-unit, and the phases are viewed as subsystems of the material-content.

The final two rules in Table 4-19 require that convective or species boundary fluxes leaving a material-content must either be “allocated” as leaving a particular phase of the material-content, or this allocation must be determined from an associated vessel geometry and the port position of the flux relative to the geometry. For example, a convective flux leaving a vapor-liquid equilibrium system must be allocated to the vapor or liquid phase, determining the state of the material transported by the convective flux. Since conservation equations will be derived for the overall modeled-unit, and not the individual phases, the fluxes between the phases of a material-content are not modeled. Therefore, all energy fluxes, and convective and species fluxes entering a material-content, do not need to be allocated to a particular phase. This method of model derivation is possible since phases in a material-content are by definition assumed to be in thermodynamic equilibrium. Deriving conservation equations in this manner also avoids the generation of high-index DAE models (that are difficult to initialize consistently for numerical solution) associated with dynamic models of equilibrium systems (Ponton and Gawthrop, 1991).

Table 4-19: Material Characterization Consistency Operators

<u>Operator</u>	<u>Actions</u>
$\forall u \forall m \forall s [modeled_unit(u) \wedge material_content(m) \wedge species(s) \wedge has_material_content(u, m) \wedge has_species(u, s) \Rightarrow has_species(m, s)]$	Verifies that any species assigned to a material unit is also assigned to the associated material-content.
$\forall u \forall m \forall s [modeled_unit(u) \wedge material_content(m) \wedge species(s) \wedge has_material_content(u, m) \wedge has_species(m, s) \Rightarrow has_species(u, s)]$	Verifies that any species assigned to a material-content is also assigned to the associated material unit.
$\forall m \forall s [material_content(m) \wedge species(s) \wedge has_species(m, s) \Rightarrow \exists p(phase(p) \wedge has_phase(m, p) \wedge has_species(p, s))]$	Verifies that any species assigned to a material-content is also assigned to at least one phase of the material-content.
$\forall p \forall s \forall m [phase(p) \wedge species(s) \wedge material_content(m) \wedge has_species(p, s) \wedge has_phase(m, p) \Rightarrow has_species(m, s)]$	Verifies that any species assigned to a phase is also assigned to the associated material-content.
$\forall u \forall m \forall r [modeled_unit(u) \wedge material_content(m) \wedge reaction(r) \wedge has_material_content(u, m) \wedge has_reaction(m, r) \Rightarrow \exists p(phase(p) \wedge has_phase(m, p) \wedge has_reaction(p, r))]$	Verifies that any reaction assigned to a material unit is also assigned to at least one phase of the associated material-content.
$\forall p \forall r \forall m \forall u [phase(p) \wedge reaction(r) \wedge material_content(m) \wedge has_reaction(p, r) \wedge has_phase(m, p) \wedge modeled_unit(u) \wedge has_material_content(u, m) \Rightarrow has_reaction(u, s)]$	Verifies that any reaction assigned to a phase is also assigned to modeled-unit of the associated material-content.
$\forall u \forall m \forall f [modeled_unit(u) \wedge material_content(m) \wedge flux(f) \wedge has_boundary_flux(u, f) \Rightarrow has_boundary_flux(m, f)]$	Verifies that any boundary flux of a material unit is a boundary flux of the associated material-content.
$\forall u \forall m \forall f [modeled_unit(u) \wedge material_content(m) \wedge flux(f) \wedge has_boundary_flux(m, f) \Rightarrow has_boundary_flux(u, f)]$	Verifies that any boundary flux of a material-content is a boundary flux of the associated material unit.
$\forall u \forall m \forall f [modeled_unit(u) \wedge material_content(m) \wedge flux(f) \wedge has_material_content(u, f) \wedge has_convective_output(u, f) \Rightarrow \exists p(phase(p) \wedge has_phase(m, p) \wedge allocated_to(m, p, f)) \vee \exists g(geometry(g) \wedge has_geometry(m, g) \wedge allocated_to(m, g, f))]$	Verifies that any convective flux from a material unit is allocated to a phase or the geometry of the associated material-content.
$\forall u \forall m \forall f [modeled_unit(u) \wedge material_content(m) \wedge flux(f) \wedge has_material_content(u, f) \wedge has_species_input(u, f) \Rightarrow \exists p(phase(p) \wedge has_phase(m, p) \wedge allocated_to(m, p, f)) \vee \exists g(geometry(g) \wedge has_geometry(m, g) \wedge allocated_to(m, g, f))]$	Verifies that any species flux from a material unit is allocated to a phase or the geometry of the associated material-content.

4.5.3 Species Topology Rules

In sequential modular flowsheet simulators, the common paradigm for declaring relevant chemical species is to assume a flowsheet wide basis. In other words, a set of chemical species is declared

for the entire flowsheet, and these species are assumed to occur in every system is the flowsheet. Therefore, trivial balance equations are calculated for those species that are not present (i.e., the flow rates of the species in all streams connected to the unit is zero). In an equation-based model, however, such an assumption would unnecessarily increase the number of and complexity of the model equations, making subsequent structural analysis and numerical solution less robust and more time consuming. Furthermore, the inclusion of trivial equations often leads to numerical singularities during solution. To avoid these difficulties, the MODEL.LA modeling language allows a finer allocation of chemical species to individual modeled-units and even individual phases within the phenomena-based model. This prevents the derivation of unnecessary model equations (e.g., balances for species that are not present in a particular modeled-unit) and reduces the complexity of equations that are derived (e.g., terms representing the contribution of a zero concentration species to an equation that describes the specific enthalpy of a phase are not included).

The methodology of localized species allocation does introduce the possibility of inconsistencies in the phenomena-based model definition. Operators that detect these inconsistencies are listed in Table 4-20. The first operator state that all species entering a system due to transport by a flux must appear in that system. For species fluxes, the transported species is assigned directly. For convective fluxes, the species transported are determined by species present in the upstream, or source, unit. If the source unit is a blackbox, all species in the modeled-unit are assumed to be transported by the flux. If the source unit is a material unit and the flux is allocated to the geometry of the material-content, all species in the material-content are assumed to be transported by the flux. If the source unit is a material unit and the flux is allocated to a phase of the material-content, all species in the phase are assumed to be transported by the flux. The second operator states that for any species flux, the source unit must also contain the species transported by the flux. The third operator states that any species participating in a reaction that is assumed to occur in a modeled-unit or phase, must appear in that modeled-unit or phase. Finally, the last operator is applied for steady-state models only. Since there is no initial holdup of species considered in the modeled-units of a steady-state model, any species appearing in a modeled-unit must be transported to the modeled-unit by a flux, or appear there due to a chemical reaction.

Table 4-20: Species Topology Consistency Operators

<i><u>Operator</u></i>	<i><u>Actions</u></i>
$\forall f \forall s \forall m [flux(f) \wedge species(s) \wedge modeled_unit(m) \wedge has_convective_input(m, f) \wedge transports_species(f, s) \Rightarrow has_species(m, s)]$	For convective fluxes from a blackbox source unit, verifies that any species in the source unit appears in the sink unit.
$\forall f \forall s \forall m [flux(f) \wedge species(s) \wedge species_flux(f, s) \wedge modeled_unit(m) \wedge has_species_output(m, s) \Rightarrow has_species(m, s)]$	For species fluxes, verifies that the transported species appears in the source unit.
$\forall r \forall m \forall s [reaction(r) \wedge reaction_element(m) \wedge species(s) \wedge has_participant(r, s) \wedge has_reaction(m, r) \Rightarrow has_species(m, s)]$	Verifies that all participating species in a reaction that occurs in a modeled-unit or phase appears in the modeled-unit or phase.
$\forall s \forall m [species(s) \wedge modeled_unit(m) \wedge has_species(m, s) \Rightarrow \exists f (flux(f) \wedge transports_species(f, s) \wedge has_input_flux(m, f) \vee \exists r (reaction(r) \wedge has_product(r, s) \wedge has_reaction(m, r))]$	In steady-state models, verifies any species in a modeled-unit is transported to the modeled-unit by a flux, or appears as a product of a reaction assigned to the modeled-unit.

The methodology of localized species allocation does increase the number of declarations that a modeler must make during model formulation. However, this burden is usually outweighed by the benefits of reduced computational complexity, memory, and time required during derivation, analysis, and solution of the resulting model equations. Finally, if localized allocation does not apply (i.e., all species are assumed to occur in all systems in the process model) a specification in the derivation context of the phenomena-based model definition may assume the global species declaration paradigm, thus making localized unit-by-unit allocations unnecessary.

4.6 Model Completeness Operators

In addition to inconsistencies, a phenomena-based model should be checked for model incompleteness. Various aspects regarding the complete specification of a phenomena-based model are dictated by the context-free grammar that describes the syntax of the MODEL.LA modeling language. In the context of the phenomena-based model digraph, these specifications can be expressed as logical operators that convey necessary decisions and assumptions for each of the modeling elements. Completeness operators also verify that complementary relationships exist between associated modeling elements. Selected examples of such operators are listed in Table 4-21.

Table 4-21: Model Completeness Operators

<u>Operator</u>	<u>Actions</u>
$\forall p \forall s [modeled_unit(p) \wedge modeled_unit(s) \wedge has_subunit(p, s)$ \Leftrightarrow $is_subunit_of(m, u)]$	Verifies that the complementary associations are made for composite unit and subunits.
$\forall m [distributed_unit(m) \Rightarrow$ $\exists s (coordinate_system(s) \wedge has_coordinate_system(m, s))$ $\wedge \exists d (distributed_dimension(d) \wedge$ $has_distributed_dimension(m, d))]$	Verifies that every distributed unit has a coordinate system selected with at least one distributed dimension
$\forall m [material_content(x)$ \Rightarrow $\exists u (modeled_unit(u) \wedge is_material_content_of(m, u))$ $\wedge \exists p (phase(p) \wedge has_phase(m, p))$ $\wedge \exists s (species(s) \wedge has_species(m, s))]$	Verifies that every material-content is associated with a modeled-unit and has at least one phase and one chemical species.
$\forall m \forall u [material_content(m) \wedge modeled_unit(u)$ $\wedge has_material_content(u, m)$ \Leftrightarrow $is_material_content_of(m, u)]$	Verifies that the complementary associations are made for materials and modeled-units.
$\forall m \forall p [material_content(m) \wedge phase(p) \wedge has_phase(m, p)$ \Leftrightarrow $is_phase_in(p, m)]$	Verifies that the complementary associations are made for all materials and phases.
$\forall p [phase(p)$ \Rightarrow $\exists m (material_content(m) \wedge is_phase_in(p, m))$ $\wedge \exists s (species(s) \wedge has_species(m, s))$ $\wedge (\exists e (equation_of_state(e) \wedge has_equation_of_state(p, e))$ $\vee \exists a (activity_coefficient_model(a)$ $\wedge has_activity_coefficient_model(p, a)))]$	Verifies that every phase is associated with a material-content, has at least one chemical species, and is characterized mechanistically.
$\forall f [flux(f)$ \Rightarrow $\exists m (modeled_unit(m) \wedge from(f, m))$ $\wedge \exists m (modeled_unit(m) \wedge to(f, m))$ $\wedge \exists t (transport_mechanism(t) \wedge has_transport_mechanism(f, t))]$	Verifies that every flux has a source and sink unit, and is characterized mechanistically.
$\forall f [convective_flux(f)$ \Rightarrow $\exists e (equation_of_state(e) \wedge has_equation_of_state(f, e))]$	Verifies that the material transported by every convective flux is characterized mechanistically.
$\forall f [species_flux(f) \Rightarrow$ $\exists s (species(s) \wedge transports_species(f, s))]$	Verifies that every species flux has a species associated with it
$\forall f \forall m [flux(f) \wedge modeled_unit(m) \wedge to(f, m)$ \Leftrightarrow $has_input_flux(m, f)]$	Verifies that the complementary associations are made for all input fluxes and modeled-units.
$\forall f \forall m [flux(f) \wedge modeled_unit(m) \wedge from(f, m)$ \Leftrightarrow $has_output_flux(m, f)]$	Verifies that the complementary associations are made for all output fluxes and modeled-units.
$\forall r [reaction(r) \Rightarrow$ $\exists s (species(s) \wedge has_reactant(r, s))$ $\wedge \exists s (species(s) \wedge has_product(r, s))]$	Verifies that every reaction has reactant and product species selected.

Similar to model inconsistency operators, model incompleteness is detected when the completeness operators make assertions about the state of the phenomena-based model that conflict with the hierarchical structure, topological structure, material-content, and chemical content analysis operators. In a computer-aided environment, these operators may be used to provide feedback to the modeler on the completeness of the phenomena-based model description.

4.7 Model Derivation Operators

The semantics of the MODEL.LA modeling language are best characterized by the set of logical operators that map the phenomena-based model description into the corresponding mathematical equation-based representation. In this manner, the procedural operators defined in this section formally describe the impact of individual modeling assumptions made on the resulting mathematical model. With these formalisms, the modeling logic of MODEL.LA makes it possible to not only automatically derive a mathematical models from the phenomena-based description, but also to explain the basis of the resulting equations, terms, and variables of the model in terms of the operators and the assumptions that produced it.

In this section, elements that represent the mathematical model are also introduced. The set of mathematical equations composing the model is identified by the predicate *model_equations(x)*. An equation is identified by the predicate *equation(x)*. The predicate *generic_variable(x)* identifies any elementary variable (e.g., *a*), or any composite variable (e.g., *sin(a)+b*c*) resulting from any combination of elementary or binary mathematical operations. Finally, variables appearing in *courier* font represent elementary variables associated with each modeling element (e.g., *temperature(x)*). These mathematical modeling elements and operations are assumed to be intrinsic.

4.7.1 Chemical Species Conservation Equation Derivation

Each modeled-unit in a phenomena-based model represents a control volume defined by the modeler. Consequently, equations expressing the conservation of mass for all relevant chemical species are derived for each modeled-unit. This relationship is expressed generically for species *i* as:

$$\frac{dN_i}{dt} = \sum_{\substack{\text{input} \\ \text{fluxes}}} n_{ij} - \sum_{\substack{\text{output} \\ \text{fluxes}}} n_{ij} + \sum_{\text{reactions}} v_{ij} \xi_j$$

where:

1. N_i is the molar holdup of species i within the boundaries of the modeled-unit,
2. n_{ij} is the molar flux of species i crossing the boundaries of the modeled-unit due to the j^{th} flux into or out the modeled-unit,
3. v_{ij} is stoichiometry of species i in reaction j , and
4. ξ_j is the molar extent of the j^{th} reaction that is assumed to occur within the boundaries of the modeled-unit.

This derivation holds at any level of modeling detail, for both composite and elementary modeled-units. The logical operators that characterize this mathematical model derivation based on chemical species conservation are listed in Table 4-22.

Table 4-22: Species Conservation Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall m \forall s [model_equations(X) \wedge modeled_unit(m) \wedge species(s) \wedge has_species(m, s) \Rightarrow include_species_balance(X, m, s)]$	Identifies when an equation expressing the conservation of mass of species s in modeled-unit m is required in the set of model equations X .
$\forall X \forall m \forall s [model_equations(X) \wedge modeled_unit(m) \wedge species(s) \wedge include_species_balance(X, m, s) \Rightarrow \Delta e(equation(e)), species_balance(m, s, e), X := X \cup e]$	Creates equation e , which expresses conservation of mass of species s in modeled-unit m , and adds e to the set of model equations X .
$\forall m \forall s \forall e [modeled_unit(m) \wedge species(s) \wedge equation(e) \wedge species_balance(m, s, e) \Rightarrow \Delta v_1(generic_variable(v_1)), species_accumulation_term(m, s, v_1), \Delta v_2(generic_variable(v_2)), species_boundary_flux_terms(m, s, v_2), \Delta v_3(generic_variable(v_3)), species_source_terms(m, s, v_3), e := v_1 = v_2 + v_3]$	Creates terms of equation e , which express conservation of mass of species s in modeled-unit m .
$\forall m \forall s \forall v [modeled_unit(m) \wedge species(s) \wedge generic_variable(v) \wedge species_accumulation_term(m, s, v) \Rightarrow v := d(molar_holdup(m, s))/dt]$	Defines term v , which expresses accumulation of mass of species s in modeled-unit m .
$\forall m \forall s \forall v_1 [modeled_unit(m) \wedge species(s) \wedge generic_variable(v_1) \wedge species_boundary_flux_terms(m, s, v_1) \Rightarrow \forall f [flux(f) \wedge transports_species(f, s) \wedge has_input_flux(m, f) \Rightarrow v_1 := v_1 + species_mole_flux(f, s)] \forall f [flux(f) \wedge transports_species(f, s) \wedge has_output_flux(m, f) \Rightarrow v_1 := v_1 - species_mole_flux(f, s)]]$	Defines composite term v_1 , which expresses transport of mass of species s across boundaries of modeled-unit m .
$\forall m \forall s \forall v [modeled_unit(m) \wedge species(s) \wedge generic_variable(v) \wedge species_source_terms(m, s, v) \Rightarrow \forall r [reaction(r) \wedge has_reaction(m, r) \wedge has_species(r, s) \Rightarrow v := v + stoichiometry(r, s) * reaction_extent(m, r)]]$	Defines composite term v , which expresses generation and consumption of mass of species s within boundaries of modeled-unit m due to chemical reactions.

Alternatively, for a composite modeled-unit, the net molar holdup for any species may be expressed as a summation of the individual molar holdups for each of its subunits:

$$N_i = \sum_{\text{subunits}} N_{ij}$$

where:

1. N_i is the molar holdup of species i within the boundaries of the composite modeled-unit, and
2. N_{ij} is the molar holdup of species i within the boundaries of the j^{th} subunit of the composite modeled-unit.

The logical operators that characterize this mathematical model derivation expressing the aggregation of control volumes into a composite control volume are listed in Table 4-23.

Table 4-23: Species Aggregation Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall m \forall s [model_equations(X) \wedge modeled_unit(m) \wedge species(s)$ $\wedge has_subunits(m) \wedge has_species(m, s)$ \Rightarrow $\Delta e(equation(e)),$ $sum_composite_species_holdup(m, s, e),$ $X := X \cup e]$	<p>Creates equation e, which expresses holdup of mass of species s in composite modeled-unit m, and adds e to set of model equations X.</p>
$\forall m \forall s \forall e [modeled_unit(m) \wedge species(s) \wedge equation(e)$ $\wedge sum_composite_species_holdup(m, s, e)$ \Rightarrow $\Delta v_1(generic_variable(v_1)),$ $v_1 = molar_holdup(m, s),$ $\Delta v_2(generic_variable(v_2)),$ $\forall u [modeled_unit(u) \wedge has_subunit(m, u)$ $\wedge has_species(u, s)$ \Rightarrow $v_2 := v_2 + molar_holdup(u, s)]$ $e := v_1 = v_2]$	<p>Creates terms of equation e, which express holdup of mass of species s of composite modeled-unit m, by summing over the individual holdups of species s for each subunit of m.</p>

For each convective flux, the molar flux of species i is expressed as:

$$n_i = x_i * n$$

where:

1. n_i is the molar flux of species i due to the convective flux,
2. x_i is the mole fraction of species i in the material transported by the convective flux,
and
3. n is the total molar flux due to the convective flux.

The logical operator that characterizes this mathematical model derivation is listed in Table 4-24.

Table 4-24: Convective Species Transport Derivation Operator

<u>Operator</u>	<u>Actions</u>
$\forall X \forall f \forall s [model_equations(X) \wedge convective_flux(f) \wedge species(s)$ $\wedge transports_species(f, s)$ \Rightarrow $\Delta e(equation(e)),$ $e := species_mole_flux(f, s)$ $= species_mole_fraction(f, s) * mole_flux(f),$ $X := X \cup e]$	<p>Creates equation e, which expresses transport of mass of species s due to convective flux f as fraction of total mass transport, and adds e to set of model equations X.</p>

4.7.2 Energy Conservation Equation Derivation

Equations expressing the conservation of energy are also derived for each modeled-unit. This relationship is expressed generically as:

$$\frac{dU}{dt} = \sum_{\substack{input \\ fluxes}} e_i - \sum_{\substack{output \\ fluxes}} e_i$$

where:

1. U is the holdup of internal energy within the boundaries of the modeled-unit, and
2. e_i is the flux of energy crossing the boundaries of the modeled-unit due to the i^{th} flux into or out of the modeled-unit.

This derivation holds at any level of modeling detail, for both composite and elementary modeled-units. The form of the accumulation term assumes that changes in the potential and kinetic energy of the system are negligible. If this assumption is not valid, the derivation can be readily extended to account for these effects. The logical operators that characterize this mathematical model derivation based on energy conservation are listed in Table 4-25.

Table 4-25: Energy Conservation Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall m [model_equations(X) \wedge modeled_unit(m)$ \Rightarrow $include_energy_balance(X, m)]$	Identifies when an equation expressing the conservation of energy in modeled-unit m is required in the set of model equations X .
$\forall X \forall y [model_equations(X) \wedge modeled_unit(m)$ $\wedge include_energy_balance(X, m)$ \Rightarrow $\Delta e(equation(e)),$ $energy_balance(m, e),$ $X := X \cup e]$	Creates equation e , which expresses conservation of energy for modeled-unit m , and adds e to set of model equations X .
$\forall m \forall e [modeled_unit(m) \wedge equation(e) \wedge energy_balance(m, e)$ \Rightarrow $\Delta v_1(generic_variable(v_1)),$ $energy_accumulation_term(m, v_1),$ $\Delta v_2(generic_variable(v_2)),$ $energy_boundary_flux_terms(m, v_2),$ $e := v_1 = v_2]$	Creates terms of equation e , which express conservation of energy in modeled-unit m .
$\forall m \forall v [modeled_unit(m) \wedge generic_variable(v)$ $\wedge energy_accumulation_term(m, v)$ \Rightarrow $y := d(internal_energy(m))/dt]$	Defines term v , which expresses accumulation of internal energy in modeled-unit m .
$\forall m \forall v_1 [modeled_unit(m) \wedge generic_variable(v_1)$ $\wedge energy_boundary_flux_terms(m, v)$ \Rightarrow $\forall f [flux(f) \wedge has_input_flux(m, f)$ \Rightarrow $v_1 := v_1 + energy_flux(f)]$ $\forall f [flux(f) \wedge has_output_flux(m, f)$ \Rightarrow $v_1 := v_1 - energy_flux(f)]]$	Defines composite term v_1 , which expresses transport of energy across boundaries of modeled-unit m .

Alternatively, for a composite modeled-unit, the net holdup of internal energy may be expressed as a summation of the individual holdups of internal energy for each of its subunits:

$$U = \sum_{\text{subunits}} U_i$$

where:

1. U is the holdup of internal energy within the boundaries of the composite modeled-unit, and
2. U_i is the holdup of internal energy within the boundaries of the i^{th} subunit of the composite modeled-unit.

The logical operators that characterize this mathematical model derivation expressing the aggregation of control volumes into a composite control volume are listed in Table 4-26:

Table 4-26: Internal Energy Aggregation Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall m [model_equations(X) \wedge modeled_unit(m) \wedge has_subunits(m)$ \Rightarrow $\Delta e(equation(e)),$ $sum_composite_internal_energy_holdup(m, e),$ $X := X \cup e]$	<p>Creates equation e, which expresses holdup of internal energy of composite modeled-unit m, and adds e to set of model equations X.</p>
$\forall m \forall e [modeled_unit(m) \wedge equation(e)$ $\wedge sum_composite_internal_energy_holdup(m, e)$ \Rightarrow $\Delta v_1(generic_variable(v_1)),$ $v_1 = internal_energy(m),$ $\Delta v_2(generic_variable(v_2)),$ $\forall u [modeled_unit(u) \wedge has_subunit(m, u)$ \Rightarrow $v_2 := v_2 + internal_energy(u)]$ $e := v_1 = v_2]$	<p>Creates terms of equation e, which express holdup of internal energy of composite modeled-unit m, by summing over the individual holdups of internal energy for each subunit of m.</p>

For each convective and species flux, the energy flux is expressed as:

$$e = h * n$$

where:

1. e is the energy flux due to the flux,
2. h is the specific enthalpy of the material transported by the flux, and
3. n is the total molar flux due to the flux.

Again, effects due to potential and kinetic energy are assumed to be negligible, but can readily be appended. The logical operator that characterizes this mathematical model derivation is listed in Table 4-27:

Table 4-27: Energy Transport Derivation Operator

<u>Operator</u>	<u>Actions</u>
$\forall X \forall f \forall v [model_equations(X)$ $\wedge (convective_flux(f) \vee species_flux(f))$ \Rightarrow $\Delta e(equation(e)),$ $e := energy_flux(f) =$ $specific_enthalpy(f) * mole_flux(f)$ $X := X \cup e]$	<p>Creates equation e, which expresses transport of energy due to convective of species flux f as product of specific enthalpy and total mass transport, and adds e to set of model equations X.</p>

4.7.3 Chemical Reaction Rate Equation Derivation

For each rate-based volumetric reaction in a phase, where the rate of reaction is defined as a function of the intensive properties of the phase, the extent of reaction is expressed as:

$$\xi = (r_{forward} - r_{reverse})V$$

where:

1. ξ is the net extent of reaction in the phase,
2. $r_{forward}$ and $r_{reverse}$ are the forward and reverse rates of reaction, respectively, and
3. V is the total volume of the phase.

The logical operators that characterize this mathematical model derivation are listed in Table 4-28.

Table 4-28: Chemical Reaction Rate Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall p \forall r [model_equations(X) \wedge phase(p) \wedge reaction(r) \wedge has_reaction(p, r) \Rightarrow \Delta e(equation(e)), \Delta v(generic_variable(v)), net_rate_of_reaction(p, r, v), e := extent_of_reaction(p, r) = v * volume(p) X := X \cup e]$	Creates equation e , which expresses net rate of reaction of reaction r in phase p , and adds e to set of model equations X .
$\forall p \forall r \forall v [phase(p) \wedge reaction(r) \wedge generic_variable(v) \wedge reversible(r) \wedge net_rate_of_reaction(p, r, v) \Rightarrow v := forward_rate_of_reaction(p, r) - reverse_rate_of_reaction(p, r)]$	Creates variable v , which expresses net rate of reaction for reversible reaction r .
$\forall p \forall r \forall v [phase(p) \wedge reaction(r) \wedge generic_variable(v) \wedge irreversible(r) \wedge net_rate_of_reaction(p, r, v) \Rightarrow v := forward_rate_of_reaction(p, r)]$	Creates variable v , which expresses net rate of reaction for irreversible reaction r .

4.7.4 Material-Content Characterization Equation Derivation

An elementary modeled-unit modeled as a blackbox is not associated with any internal intensive quantities. However, when an elementary modeled-unit is assumed to contain a material-content, the control volume defined by the modeled-unit is assumed to encompass a region with no internal boundaries containing one or more phases at equilibrium. For such a system, additional equations are derived to capture the intensive characterization of these phases. Furthermore, each extensive quantity characterizing a modeled-unit with a material-content is derived as a summation of the corresponding quantity for each phase in the material-content. This is expressed generically as:

$$B = \sum_{phases} B_i$$

where:

1. B is the holdup in the material-content of extensive quantity B (e.g., volume, internal energy, species moles, etc.), and
2. B_i is the corresponding holdup of quantity B in the i^{th} phase of the material-content.

The logical operators that characterize this mathematical model derivation are listed in Table 4-29.

Table 4-29: Material-Content Aggregation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall m [model_equations(X) \wedge material_content(m)$ \Rightarrow $\Delta e(equation(e)),$ $decompose_volume(m, e),$ $X := X \cup e]$	Creates equation e , which decomposes volume of material-content m as summation of volumes of individual phase, and adds e to set of model equations X .
$\forall m \forall e [material_content(m) \wedge equation(e) \wedge decompose_volume(m, e)$ \Rightarrow $\Delta v_1(generic_variable(v_1)),$ $v_1 := volume(m),$ $\Delta v_2(generic_variable(v_2)),$ $sum_phase_volumes(m, v_2),$ $e := v_1 = v_2]$	Creates terms of equation e , which decompose volume of material-content m as summation of volumes of individual phases.
$\forall m \forall v_1 [material_content(m) \wedge generic_variable(v)$ $\wedge sum_phase_volumes(m, v)$ \Rightarrow $\forall p [phase(p) \wedge has_phase(m, p)$ \Rightarrow $v := v + volume(p)]]$	Creates composite term v , which expresses summation of volumes for each phase of material-content m .
$\forall X \forall m [model_equations(X) \wedge material_content(m)$ \Rightarrow $\Delta e(equation(e)),$ $decompose_internal_energy(m, e),$ $X := X \cup e]$	Creates equation e , which decomposes internal energy of material-content m as summation of internal energies of individual phase, and adds e to set of model equations X .
$\forall m \forall e [material_content(m) \wedge equation(e)$ $\wedge decompose_internal_energy(m, e)$ \Rightarrow $\Delta v_1(generic_variable(v_1)), v_1 := internal_energy(m),$ $\Delta v_2(generic_variable(v_2)), sum_phase_internal_energy(m,$ $v_2),$ $e := v_1 = v_2]$	Creates terms of equation e , which decomposes internal energy of material-content m as summation of internal energies of individual phases.
$\forall m \forall v_1 [material_content(m) \wedge generic_variable(v)$ $\wedge sum_phase_internal_energy(m, v)$ \Rightarrow $\forall p [phase(p) has_phase(m, p)$ \Rightarrow $v := v + internal_energy(p)]]$	Creates composite term v , which expresses summation of internal energies for each phase of material-content m .
$\forall X \forall m \forall s [model_equations(X) \wedge material_content(m) \wedge species(s)$ $\wedge has_species(m, s)$ \Rightarrow $\Delta e(equation(e)),$ $decompose_species_holdup(m, s, e),$ $X := X \cup e]$	Creates equation e , which decomposes holdup of species s of material-content m as summation of holdups of species s of individual phases, and adds e to set of model equations X .
$\forall m \forall s \forall e [material_content(m) \wedge species(s) \wedge equation(e)$ $\wedge decompose_species_holdup(m, s, e)$ \Rightarrow $\Delta v_1(generic_variable(v_1)),$ $v_1 := species_holdup(m, s),$	Creates terms of equation e , which decomposes holdup of species s of material-content m as summation of holdups of species s of individual phases.

$\Delta v_2(\text{generic_variable}(v_2)),$ $\text{sum_phase_species_holdup}(m, s, v_2),$ $e := v_1 = v_2]$	
$\forall m \forall s \forall v_1 [\text{material_content}(m) \wedge \text{species}(s) \wedge \text{generic_variable}(v)$ $\wedge \text{sum_phase_species_holdup}(m, s, v)$ \Rightarrow $\forall p [\text{phase}(p) \wedge \text{has_phase}(m, p) \wedge \text{has_species}(s)$ \Rightarrow $v := v + \text{species_holdup}(p, s)]]]$	<p>Creates composite term v, which expresses summation of holdups of species s for each phase of material-content m.</p>

Equations expressing physical, thermal, and chemical equilibria among phases of a material-content are derived by equating the corresponding equilibrium quantities for each phase. This relationship is expressed generically as:

$$B = B_i$$

where:

1. B is the equilibrium quantity of the material-content (e.g., temperature, pressure, chemical species fugacity), and
2. B_i is the corresponding quantity of the i^{th} phase of the material-content.

The logical operators that characterize this mathematical model derivation based on thermodynamic equilibrium are listed in Table 4-30.

Table 4-30: Phase Equilibrium Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall m \forall p [model_equations(X) \wedge material_content(m) \wedge phase(p) \wedge has_phase(m, p) \Rightarrow \Delta e(equation(e)), thermal_equilibrium(m, p, e), X := X \cup e]$	Creates equation e , which expresses thermal equilibrium for phase p of material-content m , and adds e to set of model equations X .
$\forall m \forall p \forall e [material_content(m) \wedge phase(p) \wedge equation(e) \wedge thermal_equilibrium(m, p, e) \Rightarrow e := temperature(m) = temperature(p)]$	Creates terms of equation e , which express thermal equilibrium for phase p of material-content m , by equating temperature of phase to overall material-content temperature.
$\forall X \forall m \forall p [model_equations(X) \wedge material_content(m) \wedge phase(p) \wedge has_phase(m, p) \Rightarrow \Delta e(equation(e)), physical_equilibrium(m, p, e), X := X \cup e]$	Creates equation e , which expresses physical equilibrium for phase p of material-content m , and adds e to set of model equations X .
$\forall m \forall p \forall e [material_content(m) \wedge phase(p) \wedge equation(e) \wedge physical_equilibrium(m, p, e) \Rightarrow e := pressure(m) = pressure(p)]$	Creates terms of equation e , which express physical equilibrium for phase p of material-content m , by equating pressure of phase to overall material-content pressure.
$\forall X \forall m \forall p \forall s [model_equations(X) \wedge material_content(m) \wedge phase(p) \wedge species(s) \wedge has_phase(m, p) \wedge has_species(p, s) \Rightarrow \Delta e(equation(e)), chemical_equilibrium(m, p, s, e), X := X \cup e]$	Creates equation e , which expresses chemical equilibrium of species s for phase p of material-content m , and adds e to set of model equations X .
$\forall m \forall p \forall s \forall e [material_content(m) \wedge phase(p) \wedge species(s) \wedge equation(e) \wedge chemical_equilibrium(m, p, s, e) \Rightarrow \Delta v_2(generic_variable(v_1)), fugacity_model(p, s, v_1), e := fugacity(m, s) = v_1]$	Creates terms of equation e , which express chemical equilibrium of species s for phase p of material-content m , by equating fugacity of species s for phase to overall material-content fugacity of species s .
$\forall p \forall s \forall e \forall v [phase(p) \wedge species(s) \wedge equation_of_state(e) \wedge has_equation_of_state(p, e) \wedge generic_variable(v) \wedge fugacity_model(p, s, v) \Rightarrow v_1 := mole_fraction(p, s) * fugacity_coefficient(p, s) * pressure(p)]$	Creates composite term v , which expresses fugacity of species s for phase p modeled using equation of state e .
$\forall p \forall s \forall a \forall v [phase(p) \wedge species(s) \wedge activity_coefficient(a) \wedge has_activity_coefficient(p, a) \wedge generic_variable(v) \wedge fugacity_model(p, s, v) \Rightarrow v_1 := mole_fraction(p, s) * activity_coefficient(p, s) * vapor_pressure(p, s)]$	Creates composite term v , which expresses fugacity of species s for phase p modeled using activity coefficient model a .

4.7.5 Phase Characterization Equation Derivation

The total molar holdup of a phase is derived as a summation over the individual species molar holdups in the phase. This is expressed generically as:

$$N = \sum_{species} N_i$$

where:

1. N is the total molar holdup in the phase, and
2. N_i is the total molar holdup of the i^{th} species in the phase.

The logical operators that characterize this mathematical model derivation are listed in Table 4-31.

Table 4-31: Phase Species Aggregation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall p [model_equations(X) \wedge phase(p)$ \Rightarrow $\Delta e(equation(e)),$ $decompose_molar_holdup(p, e),$ $X := X \cup e]$	Creates equation e , which expresses total molar holdup of phase p as summation of individual species molar holdups, and adds e to set of model equations X .
$\forall p \forall e [phase(p) \wedge equation(e) \wedge decompose_molar_holdup(m, e)$ \Rightarrow $\Delta v_1(generic_variable(v_1)),$ $v_1 := molar_holdup(p),$ $\Delta v_2(generic_variable(v_2)),$ $\forall s [species(s) has_species(p, s)$ \Rightarrow $v_2 := v_2 + species_holdup(p, s)]$ $e := v_1 = v_2]$	Creates terms of equation e , which expresses total molar holdup of phase p as summation of individual species molar holdups.

The sum of species mole fractions of a phase must equal unity. This is expressed generically as:

$$\sum_{species} x_i = 1$$

where:

1. x_i is the mole fraction of the i^{th} species in the phase.

The logical operators that characterize this mathematical model derivation are listed in Table 4-32.

Table 4-32: Species Fraction Summation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall p [model_equations(X) \wedge phase(p)$ \Rightarrow $\Delta e(equation(e)),$ $sum_species_mole_fractions(p, e),$ $X := X \cup e]$	Creates equation e , which expresses summation of individual species molar fraction of phase p as equal to unity, and adds e to set of model equations X .
$\forall p \forall e [phase(p) \wedge equation(e) \wedge sum_species_mole_fractions(m, e)$ \Rightarrow $\Delta v(generic_variable(v)),$ $\forall s [species(s) has_species(p, s)$ \Rightarrow $v := v + species_mole_fraction(p, s)]$ $e := v = I]$	Creates equation e , which expresses summation of individual species molar fraction of phase p as equal to unity, and adds e to set of model equations X .

The total species molar holdup of a phase is equal to the product of the species mole fraction and the total molar holdup of the phase. This is expressed generically as:

$$N_i = x_i N$$

where:

1. N_i is the total molar holdup of the i^{th} species in the phase,
2. x_i is the mole fraction of the i^{th} species in the phase, and
3. N is the total molar holdup in the phase.

The logical operator that characterizes this mathematical model derivation is listed in Table 4-33

Table 4-33: Species Holdup Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall p \forall s [model_equations(X) \wedge phase(p) \wedge species(s)$ $\wedge has_species(p, s)$ \Rightarrow $\Delta e(equation(e)),$ $e := species_molar_holdup(p, s) =$ $species_mole_fraction(p, s) * molar_holdup(p),$ $X := X \cup e]$	Creates equation e , which expresses total molar holdup of phase p as product of molar fraction and molar holdup of species s , and adds e to set of model equations X .

Note that for a given phase, the set of equations formed by the operator in Table 4-33, the operators in Table 4-32, and the operators in Table 4-31 contains one redundant equation that must be excluded from the mathematical model. With regard to the structural analysis of the degrees of freedom of the model equations, it is best to include both summation equations, and

eliminate one of the equations formed by the operator in Table 4-33.

The total species molar holdup of a phase is equal to the product of the species concentration and the total volume of the phase. This is expressed generically as:

$$N_i = c_i V$$

where:

1. N_i is the total molar holdup of the i^{th} species in the phase,
2. c_i is the molar concentration of the i^{th} species in the phase, and
3. V is the total volume of the phase.

The logical operator that characterize this mathematical model derivation is listed in Table 4-34.

Table 4-34: Species Concentration Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall p \forall s [model_equations(X) \wedge phase(p) \wedge species(s)$ $\wedge has_species(p, s)$ \Rightarrow $\Delta e(equation(e)),$ $e := species_molar_holdup(p, s)$ $\quad = species_concentration(p, s) * volume(p),$ $X := X \cup e]$	Creates equation e , which expresses total molar holdup of species s for phase p as product of molar species concentration and volume, and adds e to set of model equations X .

The equations derived by the operator in Table 4-34 introduce the definition of molar species concentrations into the model, which are frequently used in mechanistic characterizations of species transport mechanisms and reaction rate laws.

The total molar holdup of a phase is equal to the product of the density and the total volume of the phase. This is expressed generically as:

$$N = \rho V$$

where:

1. N is the total molar holdup of the phase,
2. ρ is the molar density of the phase, and
3. V is the volume of the phase.

The operators that characterize this mathematical model derivation are listed in Table 4-35.

Table 4-35: Phase Density Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall p \forall s [model_equations(X) \wedge phase(p)$ \Rightarrow $\Delta e(equation(e)),$ $e := molar_holdup(p)$ $\qquad\qquad\qquad = molar_density(p) * volume(p),$ $X := X \cup e]$	Creates equation e , which expresses total molar holdup of phase p as product of molar density and volume, and adds e to set of model equations X .

The equations derived by the operator in Table 4-35 introduce the definition of molar density (or its inverse, specific volume) into the mathematical model.

The total internal energy of a phase is equal to the product of the specific internal energy and the total molar holdup of the phase. This is expressed generically as:

$$U = uN$$

where:

1. U is the total internal energy of the phase,
2. u is the specific internal energy of the phase, and
3. N is the total molar holdup of the phase.

The logical operators that characterize this mathematical model derivation are listed in Table 4-36.

Table 4-36: Phase Internal Energy Derivation Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall p \forall s [model_equations(X) \wedge phase(p)$ \Rightarrow $\Delta e(equation(e)),$ $e := internal_energy(p)$ $\qquad\qquad\qquad = specific_internal_energy(p)$ $\qquad\qquad\qquad\qquad\qquad\qquad * molar_holdup(p),$ $X := X \cup e]$	Creates equation e , which expresses total internal energy of phase p as product of specific internal energy and molar holdup, and adds e to set of model equations X .

4.7.6 Mechanistic Characterization Equation Derivation

Mechanistic characterizations of fluxes and reactions introduce additional relationships into the mathematical model. Constitutive equations for flux transport mechanisms are used to derive the net rate of flux of the transported material, energy, or species as a function of the properties of the

two interconnected modeled-units. Reaction rate laws are used to derive the rate of reaction in a phase as a function of properties (e.g., temperature, species concentrations, partial pressures, etc) of that phase. Operators that characterize these mathematical model derivations are listed in Table 4-37.

Table 4-37: Mechanistic Characterization Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall p \forall r \forall l [model_equations(X) \wedge phase(p) \wedge reaction(r)$ $\wedge rate_law(l) \wedge has_reaction(p, r) \wedge has_rate_law(r, l)$ \Rightarrow $\Delta e(equation(e)),$ $reaction_rate_law_of_reaction(p, r, e),$ $X := X \cup e]$	Creates equation e , which expresses rate of reaction for reaction r in phase p using an assumed kinetic rate law, and adds e to set of model equations X .
$\forall X \forall p \forall m [model_equations(X) \wedge flux(p)$ $\wedge transport_mechanism(m)$ $\wedge has_transport_mechanism(f, ml)$ \Rightarrow $\Delta e(equation(e)),$ $transport_mechanism_rate_law(f, m, e),$ $X := X \cup e]$	Creates equation e , which expresses rate of transport for flux f using an assumed transport mechanism, and adds e to set of model equations X .
$\forall f \forall m \forall e [energy_flux(f) \wedge mechanism(m) \wedge equation(e)$ $\wedge has_mechanism(m, f) \wedge surface_convection(m)$ $\wedge transport_mechanism_rate_law(f, m, e)$ \Rightarrow $\Delta v_1(variable(v_1)),$ $source_temperature(f, v_1),$ $\Delta v_2(variable(v_2)),$ $sink_temperature(f, v_2),$ $e := energy_flux(f) = U_coeff(f)*Area(f)*(v_1-v_2)]$	Creates terms of equation e , which expresses rate of energy transport for flux f modeled using constitutive equation for surface convection .

The final operator in Table 4-37 illustrates the use of an assumed transport mechanism in generating the form of a particular constitutive equation that characterizes an energy flux which is modeled as driven by a surface convection mechanism.

4.7.7 Thermodynamic and Physical Properties of Phases Equation Derivation

Physical and thermodynamic property correlations for each phase in the model are expressed generically as:

$$B = f(T, P, \underline{x})$$

where:

1. B is the thermodynamic or physical property of the phase,
2. T is the temperature of the phase, and

3. P is the pressure of the phase.
4. \underline{x} is the vector of species mole fractions in the phase.

Operators that characterize these mathematical model derivations are listed in Table 4-38.

Table 4-38: Physical and Thermodynamic Phase Property Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall p [model_equations(X) \wedge phase(p)$ \Rightarrow $\Delta e(equation(e)),$ $molar_density(p, e),$ $X := X \cup e]$	Creates equation e , which expresses molar density of phase p as function of properties of p , and adds e to set of model equations X .
$\forall X \forall p [model_equations(X) \wedge phase(p)$ \Rightarrow $\Delta e(equation(e)),$ $specific_enthalpy(p, e),$ $X := X \cup e]$	Creates equation e , which expresses specific enthalpy of phase p as function of properties of p , and adds e to set of model equations X .
$\forall X \forall p [model_equations(X) \wedge phase(p)$ \Rightarrow $\Delta e(equation(e)),$ $specific_internal_energy(p, e),$ $X := X \cup e]$	Creates equation e , which expresses specific internal energy of phase p as function of properties of p , and adds e to set of model equations X .
$\forall X \forall p [model_equations(X) \wedge phase(p)$ \Rightarrow $\Delta e(equation(e)),$ $heat_capacity(p, e),$ $X := X \cup e]$	Creates equation e , which expresses heat capacity of phase p as function of properties of p , and adds e to set of model equations X .
$\forall X \forall p \forall e \forall s [model_equations(X) \wedge phase(p) \wedge species(s)$ $\wedge has_species(p, s) \wedge equation_of_state(e)$ $\wedge has_equation_of_state(p, e)$ \Rightarrow $\Delta e(equation(e)),$ $fugacity_coefficient(p, s, e),$ $X := X \cup e]$	Creates equation e , which expresses fugacity coefficient of species s for phase p as function of properties of p , and adds e to set of model equations X .
$\forall X \forall p \forall a \forall s [model_equations(X) \wedge phase(p) \wedge species(s)$ $\wedge has_species(p, s) \wedge activity_coefficient(a)$ $\wedge has_activity_coefficient(p, a)$ \Rightarrow $\Delta e(equation(e)),$ $activity_coefficient(p, s, e),$ $X := X \cup e]$	Creates equation e , which expresses activity coefficient of species s for phase p as function of properties of p , and adds e to set of model equations X .
$\forall X \forall p \forall s [model_equations(X) \wedge liquid_phase(p) \wedge species(s)$ $\wedge has_species(p, s)$ \Rightarrow $\Delta e(equation(e)),$ $vapor_pressure(p, s, e),$ $X := X \cup e]$	Creates equation e , which expresses vapor pressure of species s for phase p as function of properties of p , and adds e to set of model equations X .

To complete this specification, additional operators are required that access a database of pure chemical species properties used to form the requisite property correlation of the phase.

4.7.8 Thermodynamic and Physical Properties of Fluxes Equation Derivation

Physical and thermodynamic property correlations for the transported material of each convective flux in the model are expressed generically as:

$$B = f(T, P, \underline{x})$$

where:

1. B is the thermodynamic or physical property of the transported material,
2. T is the temperature of the transported material,
3. P is the pressure of the transported material, and
4. \underline{x} is the vector of species mole fractions of the transported material.

Logical operators that characterize these mathematical model derivations are listed in Table 4-39.

Table 4-39: Physical and Thermodynamic Flux Property Operators

<u>Operator</u>	<u>Actions</u>
$\forall X \forall f [model_equations(X) \wedge convective_flux(f)$ \Rightarrow $\Delta e(equation(e)),$ $density(f, e),$ $X := X \cup e]$	Creates equation e , which expresses molar density of convective flux f as function of properties of f , and adds e to set of model equations X .
$\forall X \forall f [model_equations(X) \wedge convective_flux(f)$ \Rightarrow $specific_enthalpy(f, e),$ $X := X \cup e]$	Creates equation e , which expresses specific enthalpy of convective flux f as function of properties of f , and adds e to set of model equations X .

To complete this specification, additional operators are required that access a database of pure chemical species properties used to form the requisite property correlation of the transported material.

4.8 Model Explanation

The process of mathematical model generation from a phenomena-based description is illustrated conceptually in Figure 4-8. The state of the model digraph (determined by the phenomena-based modeling assumptions) activates operators that create mathematical model equations. Suboperations of these operators then create the individual terms and variables of these equations.

The record of operators applied and the preconditions that activated them during model derivation provide means for explaining the reasoning behind the derivation of a mathematical model. As a result, this provides a direct link between the assumptions made by the modeler in creating a phenomena-based model description, and the equations and terms of the resulting mathematical model.

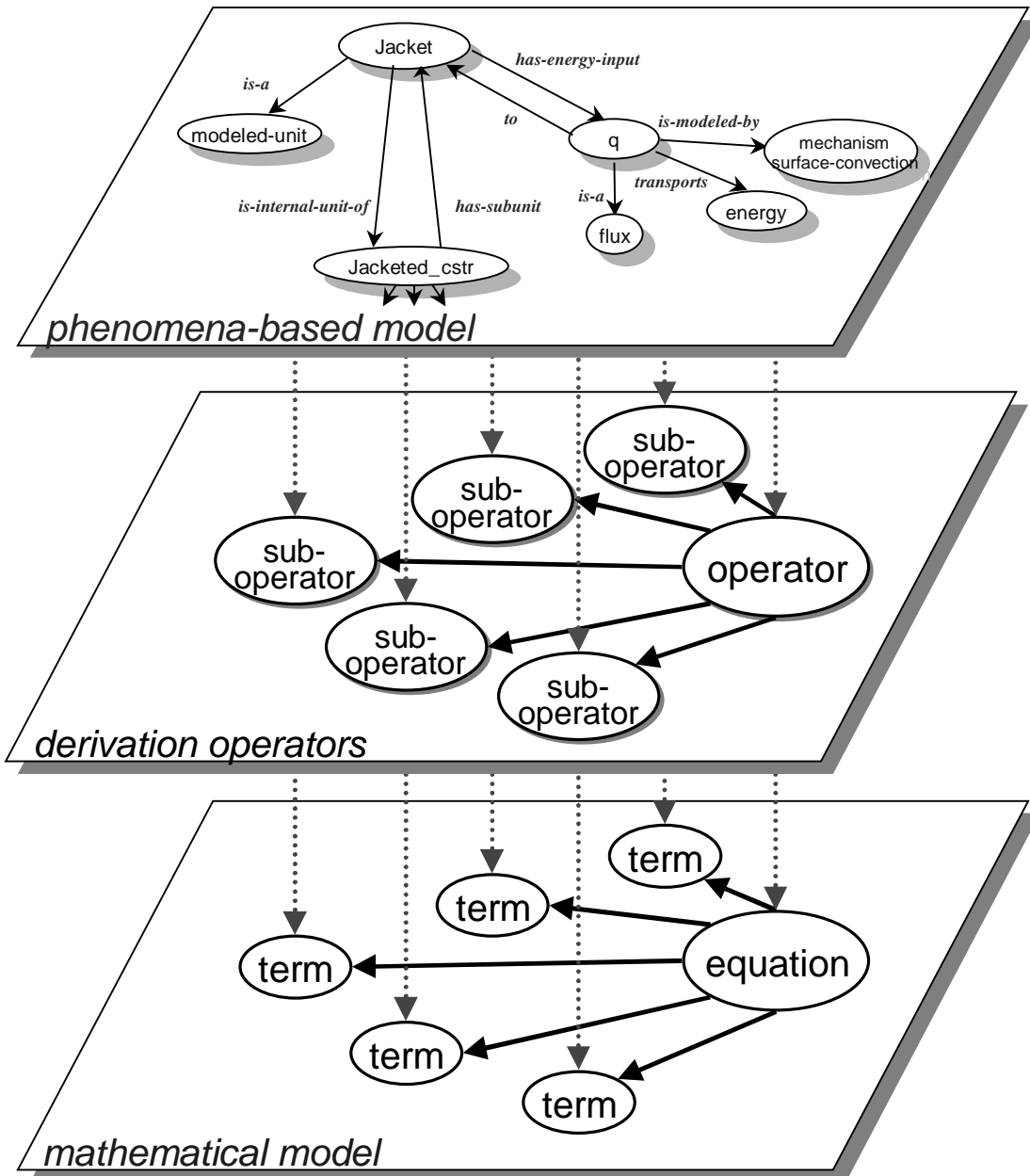


Figure 4-8: Mathematical Model Derivation

4.9 Extensions to Modeling Logic Operators

The declarative and procedural operators introduced in this chapter, which are all based on low-level intrinsic operators that act on the underlying phenomena-based model digraph, provide the basis for the description of high-level modeling logic. This logic may be readily extended to encompass additional operators for model analysis, construction, consistency, completeness, and derivation. For example, a high-level operator may be defined that asserts when a system is adiabatic:

$$\forall x [\text{adiabatic}(x) \Leftrightarrow \text{modeled_unit}(x) \wedge \neg \exists y (\text{flux}(y) \wedge (\text{has_energy_input}(x, y) \vee \text{has_energy_output}(x, y)))]$$

Other high-level analysis operators may be defined that build upon low-level operators. In a similar manner, operators that introduce additional mechanistic characterizations, along with their impact on the resulting mathematical model equations, may be defined. Furthermore, these logical operators, through knowledge of the underlying modeling assumptions, may be extended to analyze numerical results from the solution of mathematical model equations to detect possible inconsistencies in mechanistic characterizations using chemical engineering guidelines. Most importantly, supervisory logic operators may even be defined that interact with the modeler in a given context to guide him or her toward completion of certain modeling goals.

4.10 Supervisory Logic Operators

The modeling logic of MODEL.LA provides a formal basis for describing the analysis, construction, consistency and completeness of a phenomena-based model, and the derivation and explanation of the mathematical model from the phenomena-based description. This modeling logic may be readily extended to capture other domain-dependent modeling knowledge that guides the modeling activity based on goals of engineering problem. This knowledge can be integrated into the modeling logic framework as supervisory logic which interacts with the engineer to guide and structure the decisions made during model development. The process of supervisory logic interacting with the modeler in constructing a phenomena-based model is illustrated conceptually in Figure 4-9. The context of a given modeling problem, declared by the engineer, activates supervisory operators that initiate applicable modeling tasks. These operators interact with the modeler and activate model construction operators that refine the description of

the phenomena-based model.

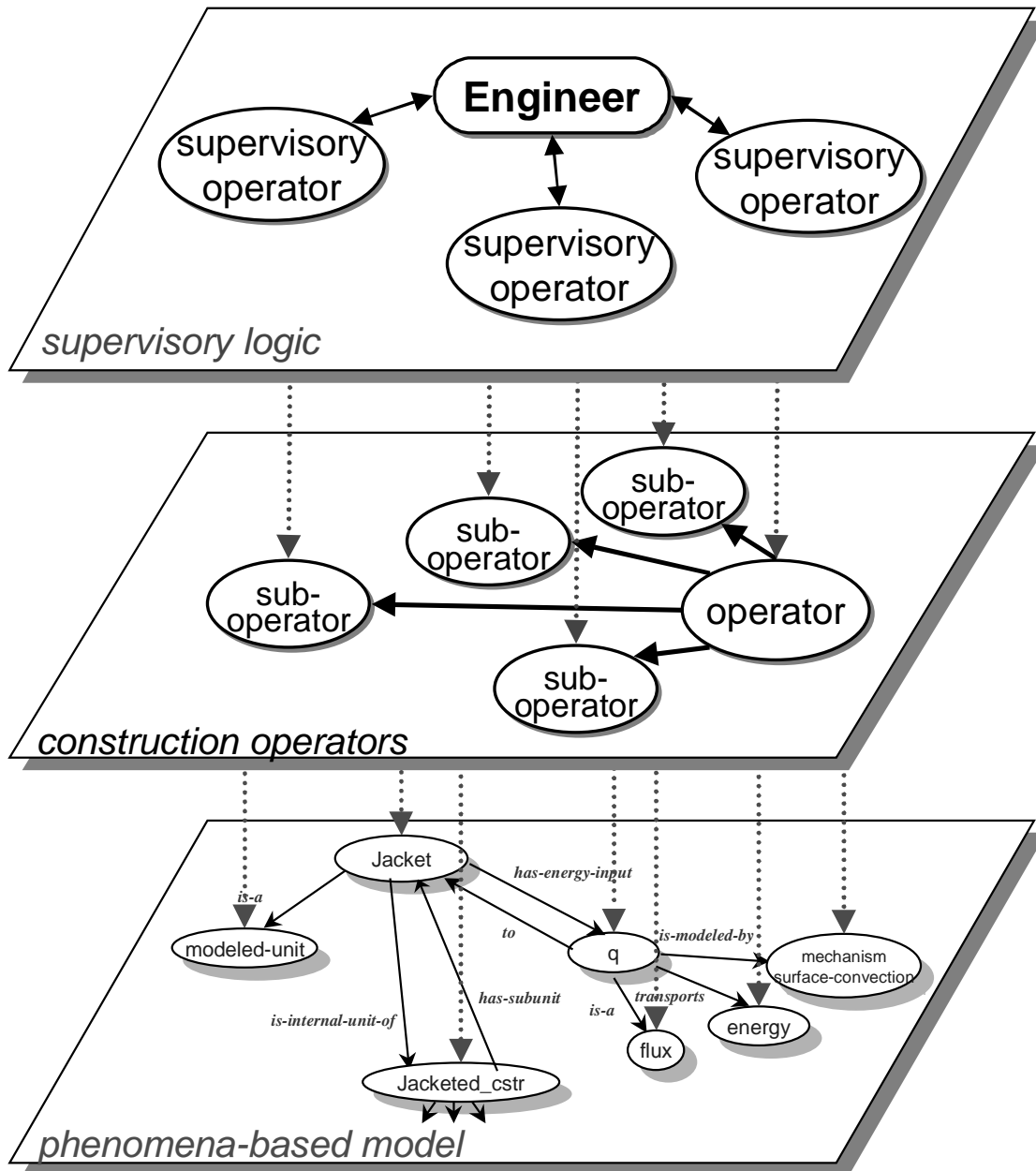


Figure 4-9: Supervisory Logic Operators

These supervisory logic operators may lend varying degrees of support to the modeling activity, including:

1. Full automation, where, based on assertions made from preconditions by the modeling logic operators, the logic can automatically refine the description of a phenomena-based model.

2. Structured interaction, where, by presenting the modeler with possible decisions and querying him/her about which assertions should be made, the logic can proceed to refine the description of a phenomena-based model.
3. Integrated documentation, where the modeler is responsible for guiding the direction of the modeling activity by selecting assertions and modeling tasks carried out, while the logic records and integrates the rationale stated by the modeler for carrying out the tasks into a record of the modeling activity.

For example, the supervisory logic operator listed in Table 4-1 is based on a methodology for the hierarchical design of continuous chemical plants (Douglas, 1985, 1988). The operator provides an explicit framework for the modeling task of creating the input-output description of a continuous chemical plant. The first two suboperators (which are examples of full automation) create a modeled-unit and label it "*input_output_plant*". In the next two suboperators (which are examples of structured interaction), the modeler defines the chemical species and reactions assumed to occur within the plant. The fifth and sixth suboperators automatically assign the species and reactions to the plant. In the seventh suboperator, for each species that is designated as a raw material by the modeler, a feed stream is added to the plant from the surroundings. Similarly, in the eighth suboperator, for each species that is designated as a product by the modeler, a convective product stream is added from the plant to the surroundings. In the ninth suboperator, for each species that is designated as a byproduct by the modeler, a waste stream is added from the plant to the surroundings. In the final suboperation (which is an example of integrated documentation), the modeler provides a rationale for the decisions made at this level.

Table 4-40: Level-1 Supervisory Operator

<u>Supervisory Operator</u>	<u>Suboperation</u>
[<i>Level_1()</i>	
\Rightarrow	
Δm (<i>modeled_unit(m)</i>),	(1)
<i>label(m) := "input_output_plant"</i>),	(2)
<i>Define_species()</i> ,	(3)
<i>Define_reactions()</i> ,	(4)
$\forall s$ [<i>species(s)</i>]	(5)
\Rightarrow	
<i>Add_species(m, s)</i> ,	
$\forall r$ [<i>reaction(r)</i>]	(6)
\Rightarrow	
<i>Add_reaction(m, r)</i> ,	
$\forall s$ [<i>species(s) \wedge raw_material(s)</i>]	(7)
\Rightarrow	
Δn (<i>source(n)</i>),	
Δf (<i>convective_flux(f)</i>), <i>label(f) := label(s) + "_feed"</i> ,	
<i>Add_convective_flux(f, n, m)</i> ,	
$\forall s$ [<i>species(s) \wedge product(s)</i>]	(8)
\Rightarrow	
Δn (<i>sink(n)</i>),	
Δf (<i>convective_flux(f)</i>), <i>label(f) := label(s) + "_product"</i> ,	
<i>Add_convective_flux(f, m, n)</i> ,	
$\exists s$ [<i>species(s) \wedge raw_material(s)</i>]	(9)
\Rightarrow	
Δn (<i>sink(n)</i>),	
Δf (<i>convective_flux(f)</i>), <i>label(f) := "byproducts"</i> ,	
<i>Add_convective_flux(f, n, m)</i> ,	
<i>Document_decisions()</i>]	(10)

The next *Level-2* for the hierarchical design of the plant may be represented by the operator in Table 4-41. The first six suboperations refine the input-output plant into two subunits, representing a reaction section and a separation section. All species are added to both subunits, then all reactions are added to only the reaction section. A convective effluent stream is then added from the reaction section to the separation section. The raw material input streams from *level-1* to the plant are then allocated to the reaction section, and the product and byproduct from *level-1* from the plant are then allocated from the separation section. Recycle streams are then added from the separation section to the reaction section for each raw material that is not completely reacted (as decided by the modeler). The modeler then documents the decisions

made.

Table 4-41: Level-2 Supervisory Operator

<u>Supervisory Operator</u>	<u>Suboperation</u>
$\exists m [Level_2() \wedge modeled_unit(m) \wedge label(m) := "input_output_plant"]$	
\Rightarrow	
$\Delta x (modeled_unit(x)),$	(1)
$label(x) := "reaction_section",$	(2)
$\Delta p (modeled_unit(p)),$	(3)
$label(p) := "separation_section",$	(4)
$Add_subunit(m, x);$	(5)
$Add_subunit(m, p);$	(6)
$\forall s [species(s)$	(7)
\Rightarrow	
$Add_species(r, s),$	
$Add_species(p, s),$	
$\forall r [reaction(r)$	(8)
\Rightarrow	
$Add_reaction(m, r),$	
$\Delta f (convective_flux(f)),$	(9)
$label(f) := "effluent",$	(10)
$Add_convective_flux(f, x, p),$	(11)
$\forall f [flux(f) \wedge has_input(m, f)$	(12)
\Rightarrow	
$Allocate_flux(f, x),$	
$\forall f [flux(f) \wedge has_output(m, f)$	(13)
\Rightarrow	
$Allocate_flux(f, p),$	
$\forall s [species(s) \wedge raw_material(s) \wedge \neg completely_reacted(s, x)$	(14)
\Rightarrow	
$\Delta f (convective_flux(f)),$	
$label(f) := label(s) + "_recycle",$	
$Add_convective_flux(f, p, x),$	
$Document_decisions()]$	(15)

The applicability of these supervisory logic operators in this simple example are dependent on a particular modeling context, the hierarchical design of a continuous chemical plant. A wide variety of such examples may be formulated for various modeling goals. Thus, while these operators are treated as distinct from the core MODEL.LA modeling logic, they can provide the most powerful means for assisting the process of model development.

Chapter 5

The MODEL.LA Modeling Environment

In the previous two chapters, the context-free grammar of the MODEL.LA modeling language, which specifies the *syntax* of the modeling language, and the framework of modeling logic operators, which specifies the *semantics* of the modeling language elements, were introduced. The modeling elements and semantic relationships of the modeling language provide the vocabulary that allows an engineer to articulate assumptions about the structure, physicochemical phenomena, and mechanistic characterizations of a process. The modeling logic operators provide a framework that formally describes the impact of these phenomena-based assumptions on the resulting mathematical model, and makes it possible to systematize the modeling process. This framework enables the computer to understand the implication of the modeling assumptions, to assist the modeler in constructing the phenomena-based model description, to detect model inconsistencies and incompleteness, to automatically derive mathematical models, and to explain the terms and equations of the resulting mathematical model in terms of the modeling assumptions. The phenomena-based modeling language and logical framework has been described independently of any computer-aided implementation. However, without such an implementation, it would be impossible to test and evaluate these phenomena-based concepts in a meaningful manner.

In this chapter, the implementation of the modeling language and logical framework of MODEL.LA in a computer-aided modeling environment is presented through discussion of its functionality, graphical user interface, and overall structure. This software acts as an interface between the modeler and the underlying modeling language and logic and provides an interactive environment for phenomena-based modeling of dynamic or static systems of arbitrary structure

(with lumped or spatially distributed properties), hierarchical levels of detail, and multi-context depictions.

5.1 Software Structure

The MODEL.LA modeling environment is designed for personal computers running 32-bit Microsoft Windows operating systems. The overall software structure of MODEL.LA is depicted graphically in Figure 5-1.

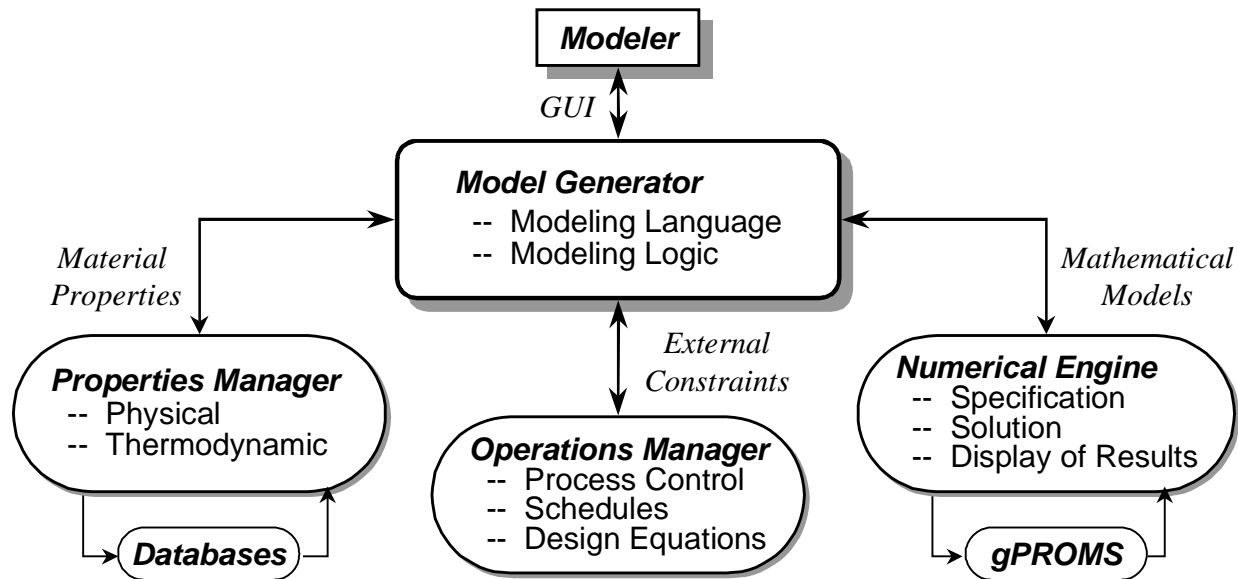


Figure 5-1: MODEL.LA Modeling Environment Software Structure

The four primary components of MODEL.LA are the

1. *Model Generator*,
2. *Properties Manager*,
3. *Operations Manager*, and
4. *Numerical Engine*.

The details of each of these components will be discussed in the remainder of this chapter.

5.2 Model Generator

The key component of the MODEL.LA modeling environment is the *Model Generator*. The *Model Generator* integrates the modeling language, which provides a basis for description of phenomena-based models, and the modeling logic operators, which enable the computer to understand and analyze the modeling assumptions, assist the modeler in constructing the

phenomena-based model description, detect model inconsistencies and incompleteness, automatically derive mathematical models, and explain the terms and equations of the resulting mathematical model. The modeler interacts with the modeling environment through a graphical user interface (GUI).

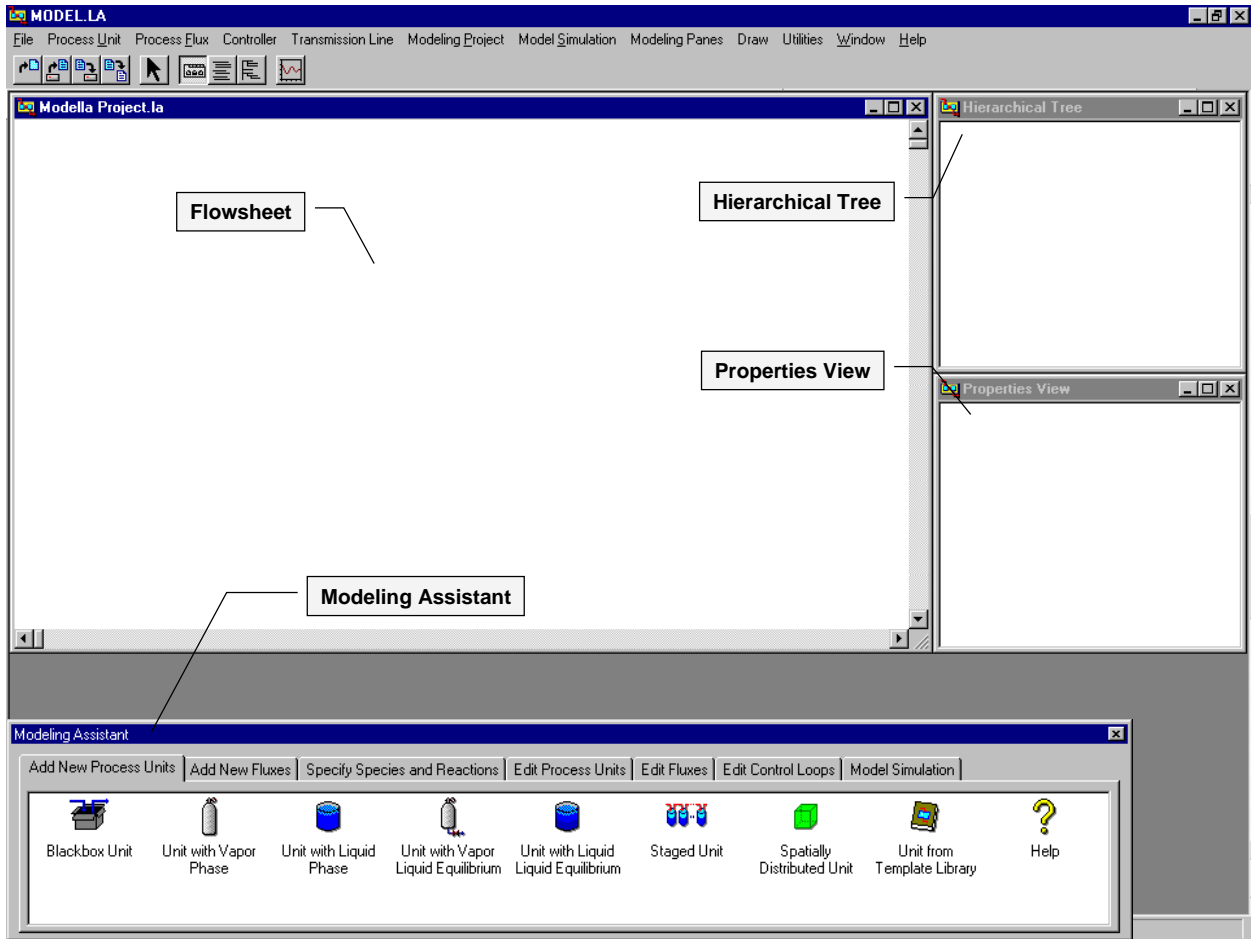


Figure 5-2: MODEL.LA Graphical User Interface

The primary GUI of MODEL.LA is illustrated in Figure 5-2. It consists of four key components:

1. *Flowsheets* provide graphical means of declaring and depicting topological and hierarchical structure,
2. *Hierarchical Tree* provides a graphical overview of the hierarchical structure of a phenomena-based model,
3. *Properties View* displays the textual language-based assumptions that characterize the phenomena-based modeling elements, and

4. *Modeling Assistant* provides the modeler with a palette of modeling options and decisions available for declaring, characterizing, and analyzing the elements of a phenomena-based model during the course of the modeling activity.

In addition to these elements, the GUI of the *Model Generator* uses a rich set of contextual menus and dialogs for specification of the topological structure, hierarchical structure, chemical characterization, and the mechanistic description of a phenomena-based model. The interactive nature of the modeling environment provides the modeler with immediate feedback during model development. Also, the corresponding *MODEL.LA* language-based description is automatically generated after each interaction and displayed in the *Properties View*.

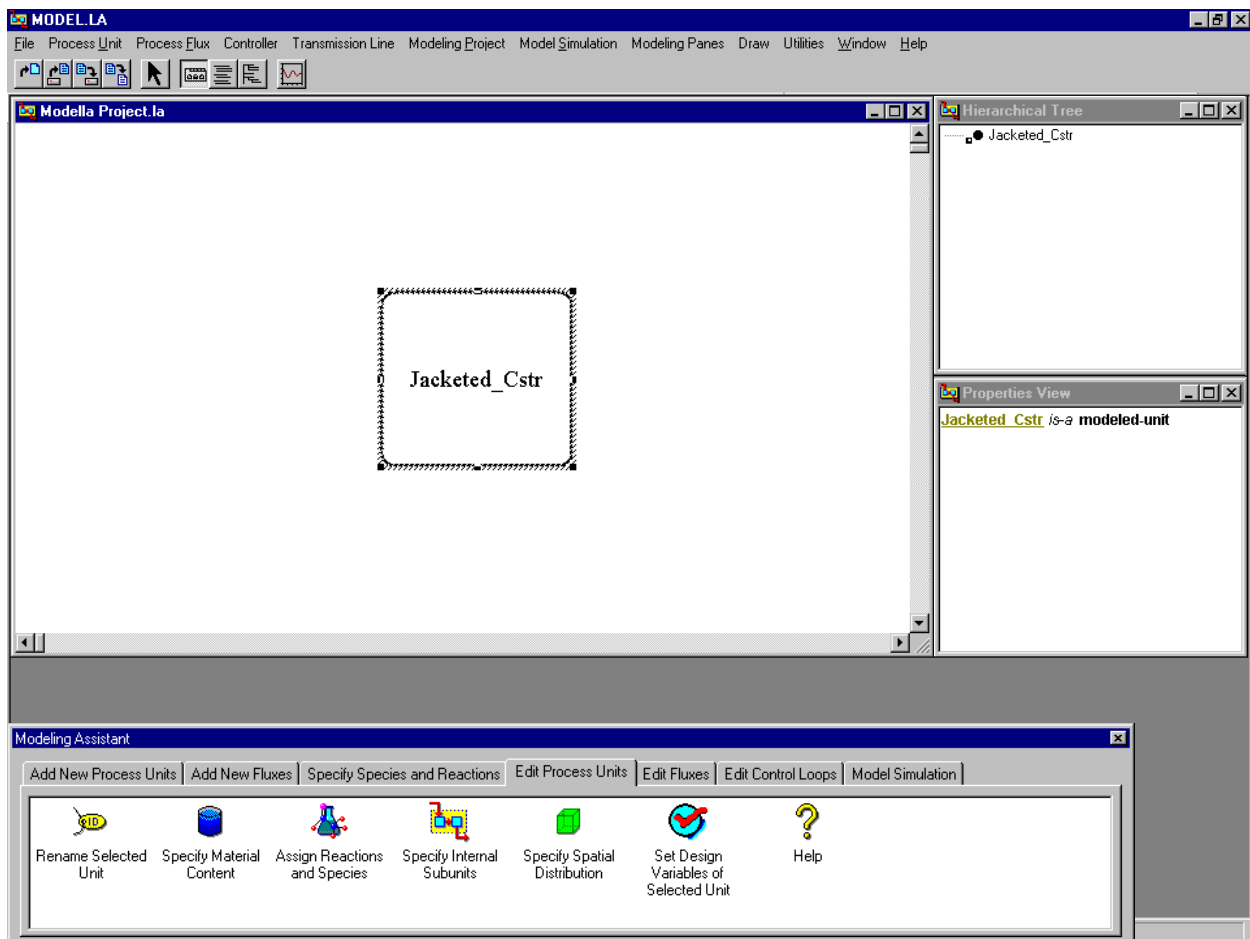


Figure 5-3: Declaration of a Modeled-Unit

5.2.1 Topological Structure

The topological structure of a phenomena-based model is declared using an intuitive flowsheet

approach where icons, depicting modeled-units, are interconnected by arrows, depicting fluxes between the modeled-units. Flux arrows are color-coded indicating the transport of material (blue), energy (red), or a selected chemical species (green).

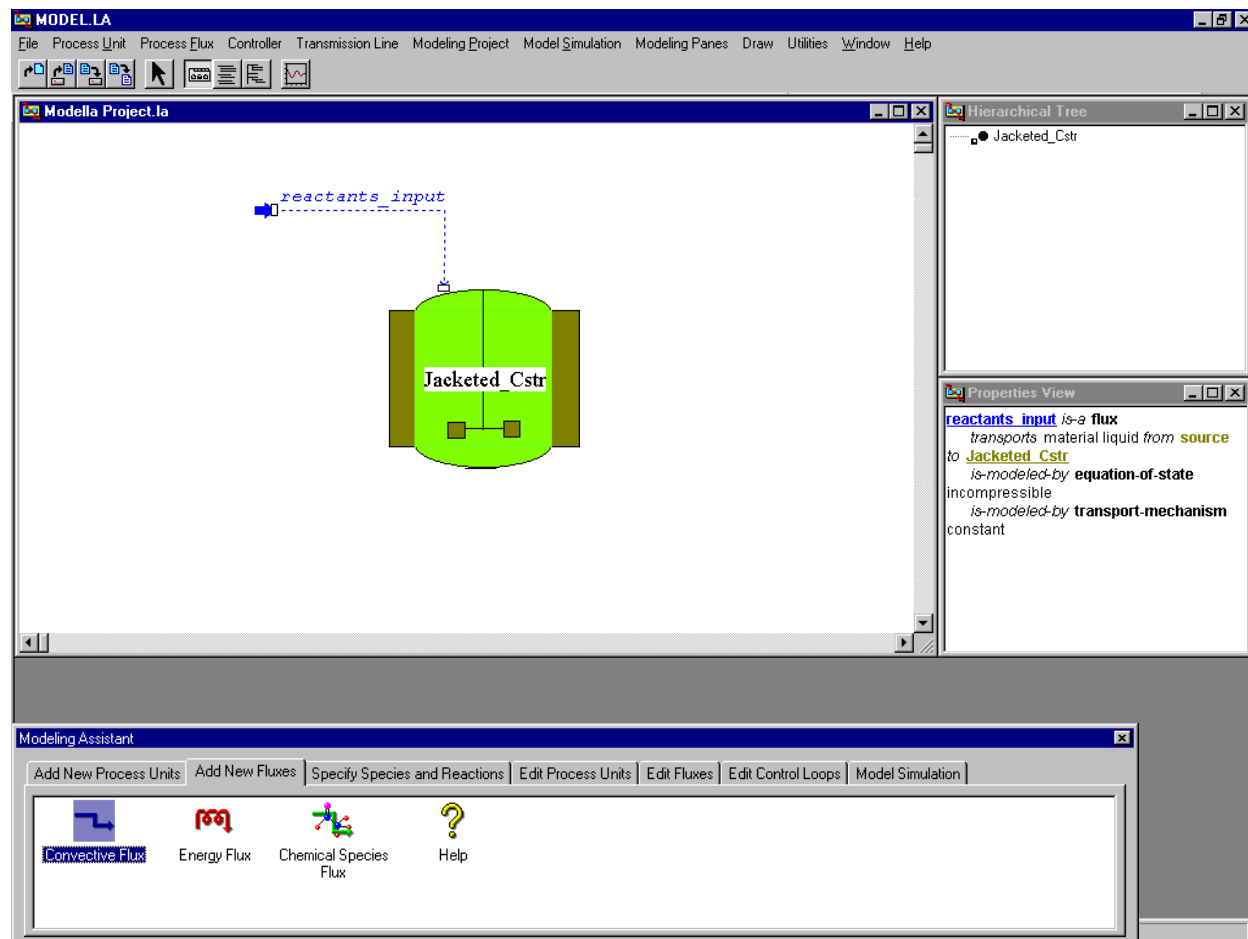


Figure 5-4: Declaration of a Convective Flux

Modeled-units are added to the flowsheet using the *Add New Process Units* tab of the *Modeling Assistant* (shown at the bottom of Figure 5-2). The modeler declares a modeled-unit by selecting the appropriate icon (e.g., *Blackbox Unit*) on the *Modeling Assistant* and dropping it on the flowsheet. The new modeled-unit, which is automatically given a default name, can be renamed by the modeler by selecting the *Rename Selected Unit* option on the *Edit Process Units* tab of the *Modeling Assistant* (shown at the bottom of Figure 5-3). For example, in Figure 5-3, a modeled-unit has been declared by the modeler and renamed *Jacketed_CSTR*. The *Hierarchical Tree* is automatically updated to reflect this addition, and the *Properties View* shows all assumptions made for the selected modeled-unit. If desired, the default icon for a modeled-unit

may also be replaced with more descriptive icons using a right-click menu option. Such icons do not change the phenomena-based description of a modeled-unit, but can be used to give graphical clues as to the purpose of the modeled-unit. For example, in Figure 5-4, the default icon for the *Jacketed_CSTR* is replaced with an icon depicting a jacketed well-stirred tank.

Fluxes are added to the flowsheet using the *Add New Fluxes* tab of the *Modeling Assistant* (shown at the bottom of Figure 5-4). The modeler declares a flux by selecting the appropriate icon (e.g., *Convective Flux*) on the *Modeling Assistant* and dragging on the flowsheet from the source modeled-unit to the sink modeled-unit. The flux can be renamed by selecting the *Rename Selected Flux* option on the *Edit Fluxes* tab of the *Modeling Assistant* (as shown in Figure 5-18). For example, in Figure 5-4, a convective flux has been declared from the surrounding environment to the *Jacketed_CSTR* and renamed *reactants_input*. As for modeled-units, the *Properties View* shows all assumptions made for the selected flux.

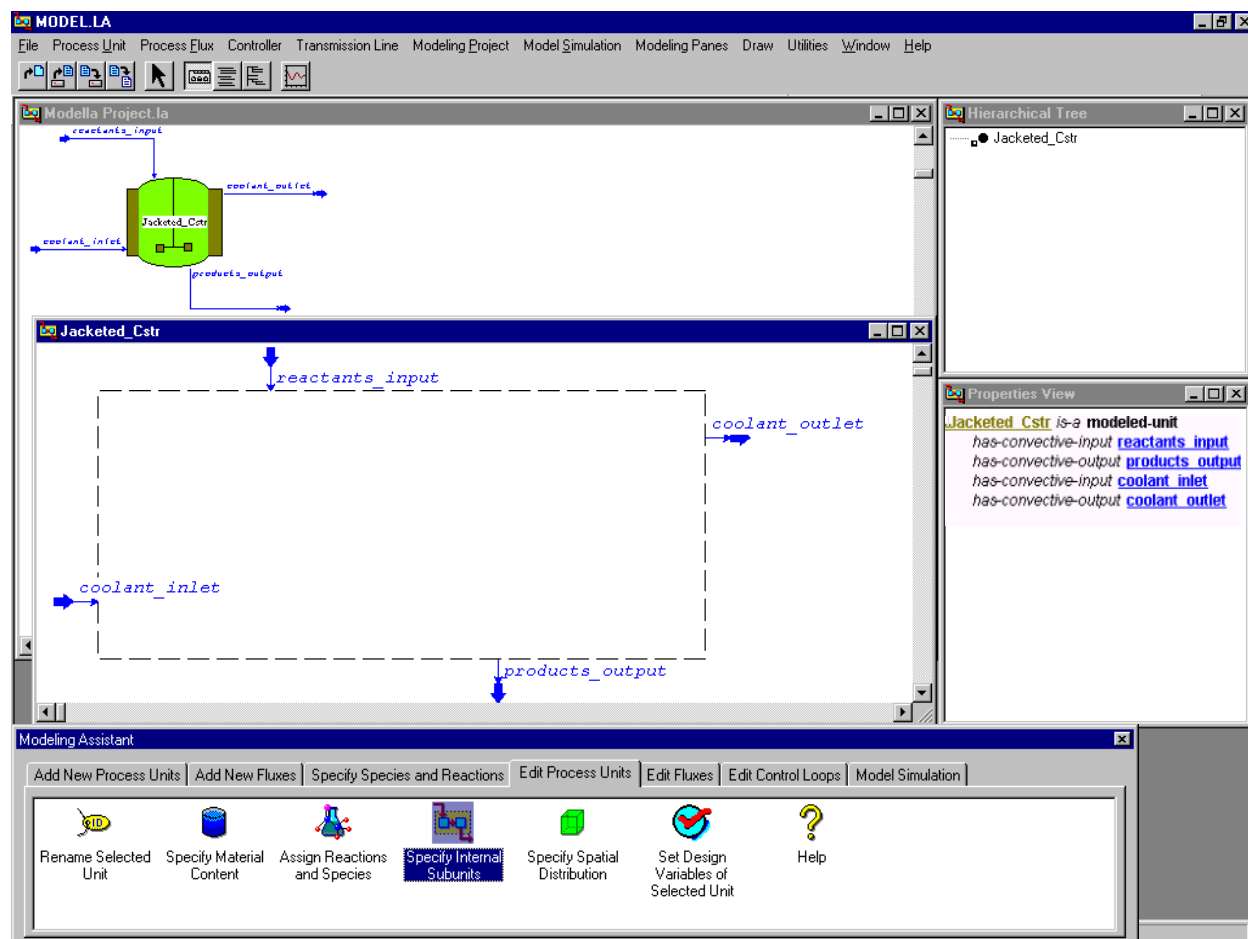


Figure 5-5: Decomposition Flowsheet

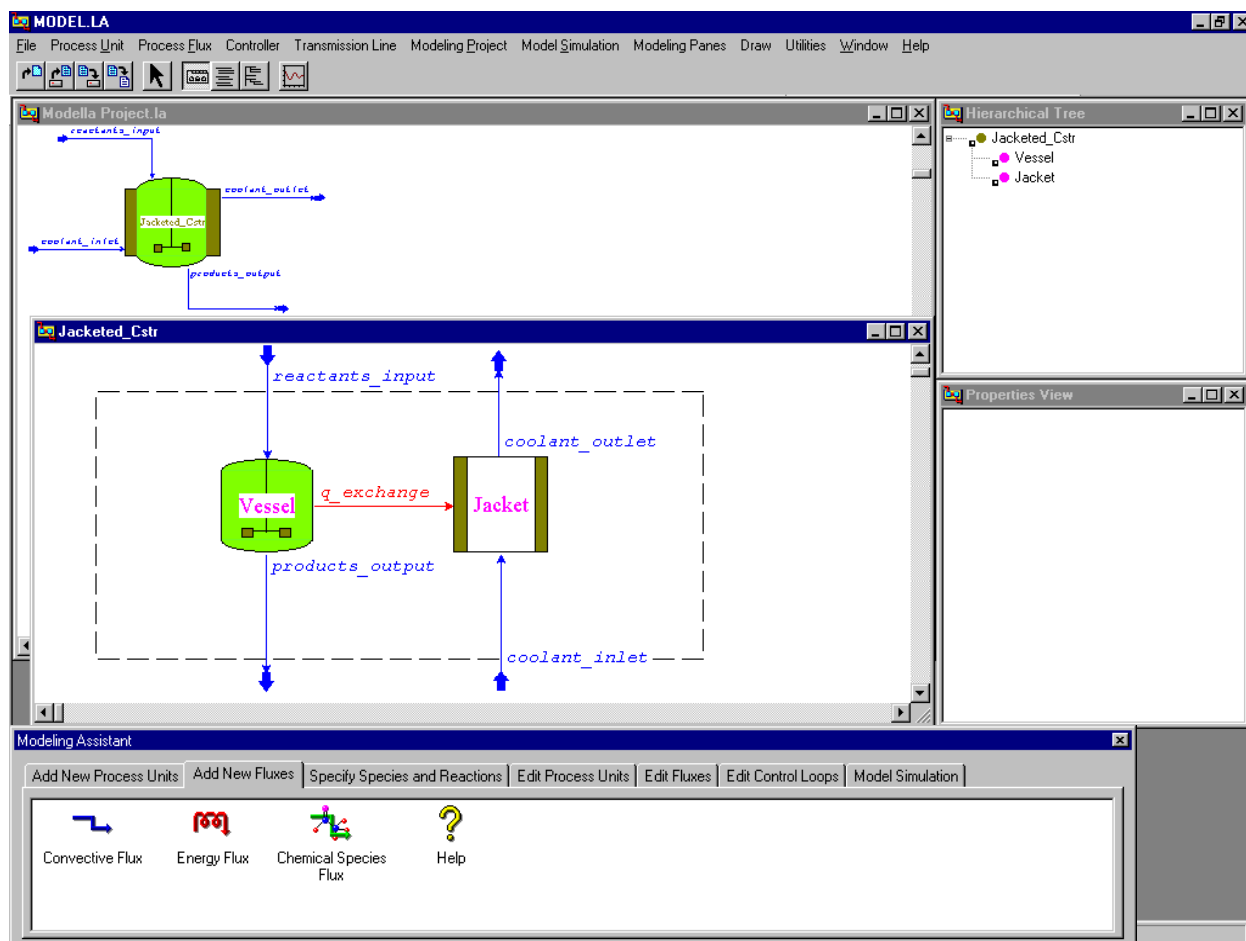


Figure 5-6: Jacketed_CSTR Decomposition

5.2.2 Hierarchical Structure

The hierarchical structure of a phenomena-based model is declared using a series of decomposition flowsheets where internal subunits may be declared within the boundary of a composite modeled-unit. A particular modeled-unit is decomposed by selecting the *Specify Internal Subunits* option on the *Edit Process Units* tab of the *Modeling Assistant* (shown at the bottom of Figure 5-5). For example, Figure 5-5 illustrates the initial decomposition flowsheet for the *Jacketed_CSTR* modeled-unit. In a decomposition flowsheet, the boundaries of the composite modeled-unit appear as a dashed outline. Subunits of the composite modeled-unit may be added within this boundary in the same manner as in the top-level flowsheet. Boundary fluxes to or from the composite modeled-unit initially appear terminating at the dashed boundary. Subsequently, these fluxes must be allocated to the subunits. For example, in Figure 5-5, the

Jacketed_CSTR has four boundary fluxes, each of which must be eventually allocated to a subunit of the *Jacketed_CSTR*. Internal fluxes may also be declared between any two subunits of a composite modeled-unit. Figure 5-6 shows the final hierarchical structure of the *Jacketed_CSTR*, which has been decomposed into two subunits, a *Vessel* and a *Jacket*. The *reactants_input* and *products_output* boundary fluxes have been allocated to the *Vessel*, and the *coolant_inlet* and *coolant_outlet* boundary fluxes have been allocated to the *Jacket*. Additionally, an internal energy flux, *q_exchange*, has been declared from the *Vessel* to the *Jacket*. If desired, these subunits may be likewise decomposed down to an arbitrary level of detail.

In addition to the disaggregation of an existing modeling unit into a set of new subunits, a set of existing modeled-units may also be aggregated into a new composite modeled-unit. This action is declared by the modeler using the *Modeled-Unit Aggregation Dialog* (Figure 5-7) where a set of modeled-units may be selected for aggregation into a new composite modeled-unit.

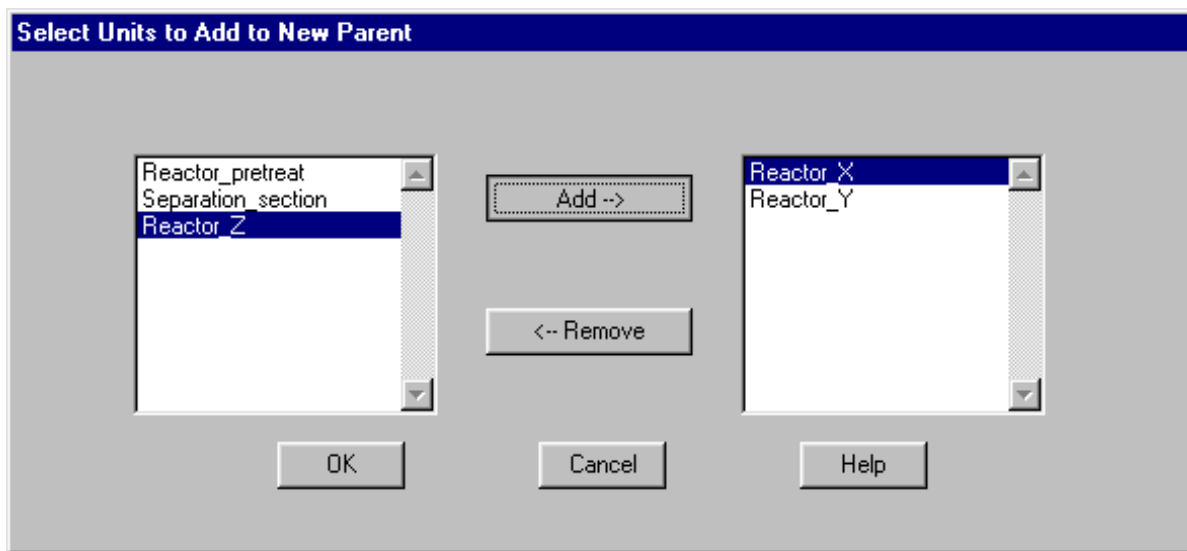


Figure 5-7: Modeled-Unit Aggregation Dialog

Two special cases of the composite modeled-unit are the staged and the distributed composite units. Staged modeling units may be declared by selecting the *Staged Unit* icon on the *Add Process Units* tab of the *Modeling Assistant* and dragging it to a flowsheet. In a staged modeled-unit, the unit is refined into a series of identical subunits. In the decomposition flowsheet of the staged modeled-unit, only three subunits appear. Assumptions made for the representative (second) subunit are automatically propagated to the other subunits in the staged system. Fluxes in the staged system are declared only between the first two subunits. An

identical flux is then automatically created between every two subsequent adjacent subunits in the staged system. For example, in Figure 5-8, the rectifying section of a distillation column is modeled as a staged system with 15 staged subunits. Fluxes in the staged system are declared between the first two subunits, *Rectifying_1* and *Rectifying_2*. For example, a convective flux identical to flux *Lr* is propagated between every two adjacent subunits in the stage. Thus, stage *Rectifying_10* has a convective input flux, *Lr_8*, and a convective output flux, *Lr_9*, resulting from declaration of *Lr*.

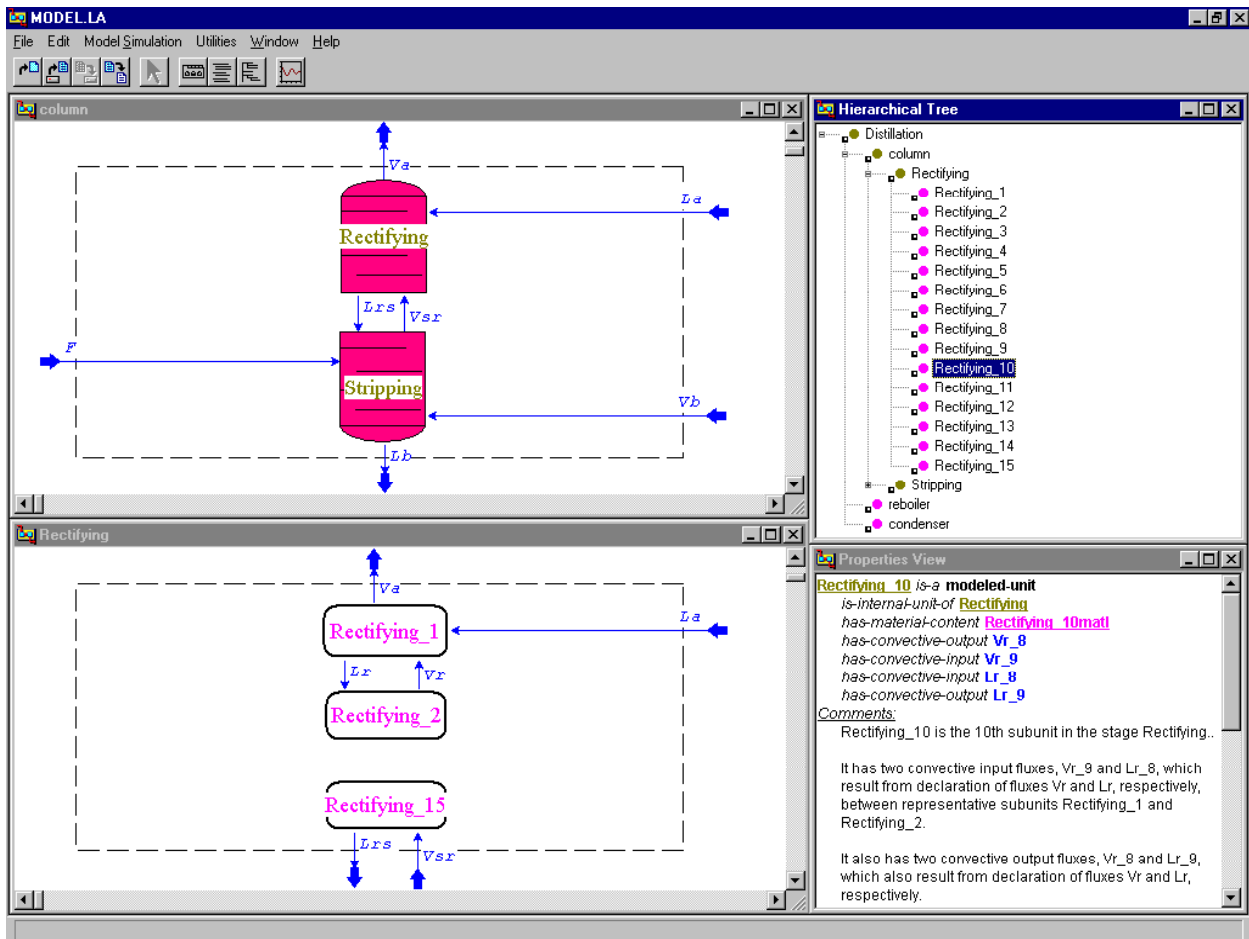


Figure 5-8: Example Staged Modeled-Unit

Distributed modeling units may be declared by selecting the *Distributed Unit* icon on the *Add Process Units* tab of the *Modeling Assistant* and dragging it to a flowsheet. A distributed unit represents a process unit whose state is characterized internally by spatially distributed properties. The assumed distribution of such a modeled-unit is declared using the *Spatial Distribution Dialog*, shown in Figure 5-9, where the coordinate system, distributed dimensions,

and desired solution methods are selected. Distributed modeled-units assumptions are declared using a differential element subunit, along with boundary subunits for each of the distributed dimensions. The balance equations for such a unit are in the form of partial differential equations (PDEs). In these systems, differential fluxes are added to or from the representative differential element subunit for each distributed dimension. Fluxes to or from the boundary elements determine the boundary conditions of the PDE balance equations for the distributed unit.

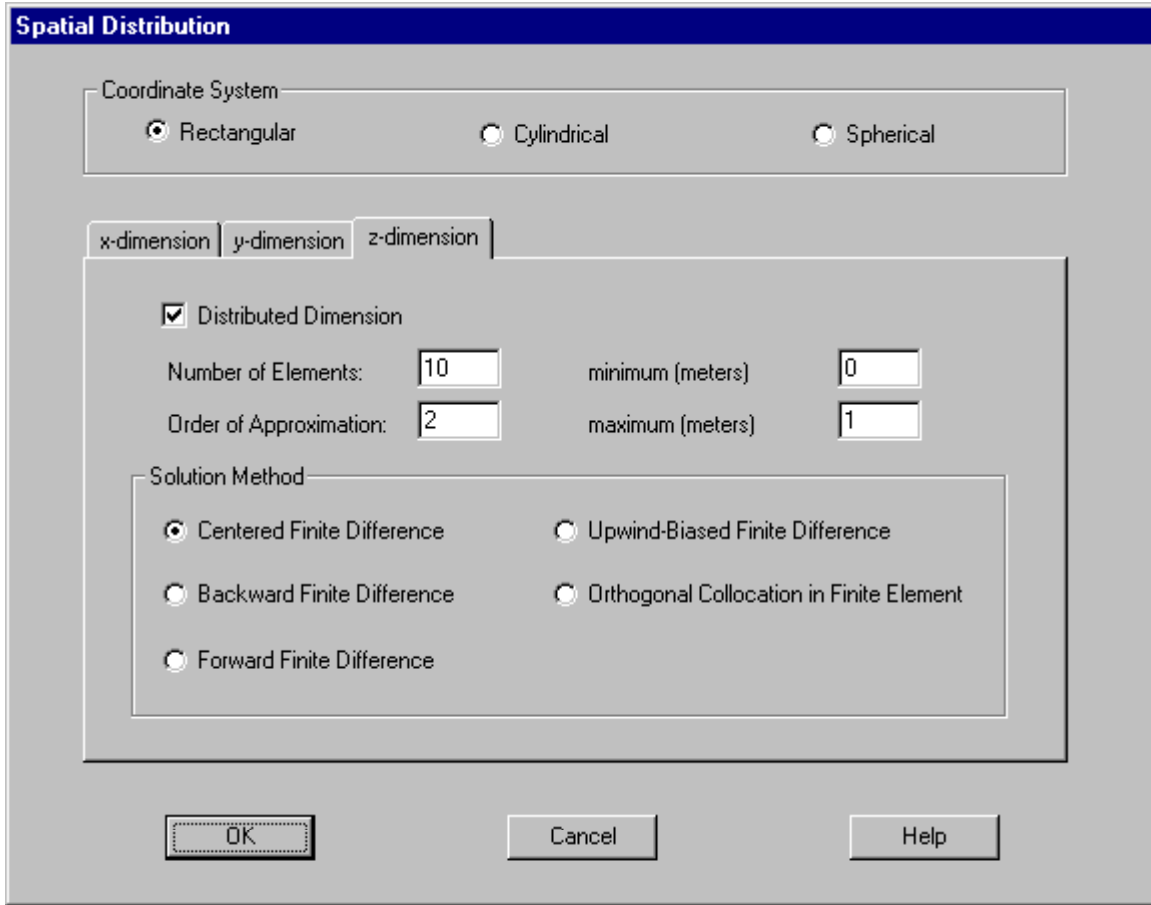


Figure 5-9: Spatial Distribution Dialog

For example, Figure 5-10, a model of the classic heated fin example (Bird et al, 1960) is illustrated. The fin is modeled with a rectangular coordinate system distributed along the z-dimension (as shown in Figure 5-9). An energy flux, q_z , is assumed along the z-axis. There is also an energy flux, T_w , at the initial boundary of the z-axis, and another, q_w , to the surroundings along the length of the z-dimension. The no flux boundary condition at the final z-axis boundary is established since there is no boundary flux declared at the final boundary element of the z-axis. The corresponding steady-state model equations derived are listed in Table 5-1.

Table 5-1: Heated Fin Model Equations

<p>1. Unit fin_z energy balance $\mathbf{z} := (\mathbf{z}_{\min, \text{fin}}, \mathbf{z}_{\max, \text{fin}})$ $(-\mathbf{e}_{\text{vol, qw, source}}(\mathbf{z})) - (\partial \mathbf{e}_{\text{area, qz, source}}(\mathbf{z}) / \partial \mathbf{z}_{\text{fin}}) = 0$</p> <p>2. Unit fin_z1 boundary energy balance $\mathbf{e}_{\text{area, qz}}(\mathbf{z}_{\text{coord_min_fin}}) = \mathbf{e}_{\text{Tw, source}} / ((\mathbf{x}_{\max, \text{fin}} - \mathbf{x}_{\min, \text{fin}}) * (\mathbf{y}_{\max, \text{fin}} - \mathbf{y}_{\min, \text{fin}}))$</p> <p>3. Unit fin_z2 boundary energy balance $\mathbf{e}_{\text{area, qz}}(\mathbf{z}_{\text{coord_max_fin}}) = 0$</p> <p>4. Flux qz flux integrated $\mathbf{z} := (\mathbf{z}_{\min, \text{fin}}, \mathbf{z}_{\max, \text{fin}})$ $\mathbf{e}_{\text{qz, source}}(\mathbf{z}) = (\mathbf{e}_{\text{area, qz, source}}(\mathbf{z}) * (\mathbf{x}_{\max, \text{fin}} - \mathbf{x}_{\min, \text{fin}})) * (\mathbf{y}_{\max, \text{fin}} - \mathbf{y}_{\min, \text{fin}})$</p>
--

Subsequent mechanistic characterizations of the energy fluxes assume a temperature fixed by T_w at the initial z-boundary, Fourier conduction by qz along the z-axis, and surface convection by qw at the outer boundary along the z-dimension, thus completing the heated fin example. The additional model equations derived from these assumptions are shown in Table 5-2.

Table 5-2: Heated Fin Model Equations with Mechanistic Characterizations

<p>1. Unit fin_z energy balance $\mathbf{z} := (\mathbf{z}_{\min, \text{fin}} +, \mathbf{z}_{\max, \text{fin}} -)$ $(-\mathbf{e}_{\text{vol, qw, source}}(\mathbf{z})) - (\partial \mathbf{e}_{\text{area, qz, source}}(\mathbf{z}) / \partial \mathbf{z}_{\text{fin}}) = 0$</p> <p>2. Unit fin_z1 boundary energy balance $\mathbf{e}_{\text{area, qz}}(\mathbf{z}_{\text{coord_min_fin}}) = \mathbf{e}_{\text{Tw, source}} / ((\mathbf{x}_{\max, \text{fin}} - \mathbf{x}_{\min, \text{fin}}) * (\mathbf{y}_{\max, \text{fin}} - \mathbf{y}_{\min, \text{fin}}))$</p> <p>3. Unit fin_z2 boundary energy balance $\mathbf{e}_{\text{area, qz}}(\mathbf{z}_{\text{coord_max_fin}}) = 0$</p> <p>4. Flux Tw thermal equilibrium $\mathbf{T}_{\text{fin, z1}}(\mathbf{z}_{\text{coord_min_fin}}) = \mathbf{T}_{\text{Tw, source}}$</p> <p>5. Flux qw energy flux $\mathbf{z} := (\mathbf{z}_{\min, \text{fin}}, \mathbf{z}_{\max, \text{fin}})$ $\mathbf{e}_{\text{vol, qw, source}}(\mathbf{z}) = (\mathbf{U}_{\text{o, qw}}(\mathbf{z}) * (\mathbf{T}_{\text{fin, z}}(\mathbf{z}) - \mathbf{T}_{\text{qw, sink}}(\mathbf{z}))) / \mathbf{A}_{\text{qw}}(\mathbf{z})$</p> <p>6. Flux qz energy flux $\mathbf{z} := (\mathbf{z}_{\min, \text{fin}}, \mathbf{z}_{\max, \text{fin}})$ $\mathbf{e}_{\text{area, qz, source}}(\mathbf{z}) = (-(\mathbf{k}_{\text{f, qz}}(\mathbf{z}) * (\partial \mathbf{T}_{\text{fin, z}}(\mathbf{z}) / \partial \mathbf{z}_{\text{fin}})))$</p> <p>7. Flux qz flux integrated $\mathbf{z} := (\mathbf{z}_{\min, \text{fin}}, \mathbf{z}_{\max, \text{fin}})$ $\mathbf{e}_{\text{qz, source}}(\mathbf{z}) = (\mathbf{e}_{\text{area, qz, source}}(\mathbf{z}) * (\mathbf{x}_{\max, \text{fin}} - \mathbf{x}_{\min, \text{fin}})) * (\mathbf{y}_{\max, \text{fin}} - \mathbf{y}_{\min, \text{fin}})$</p>

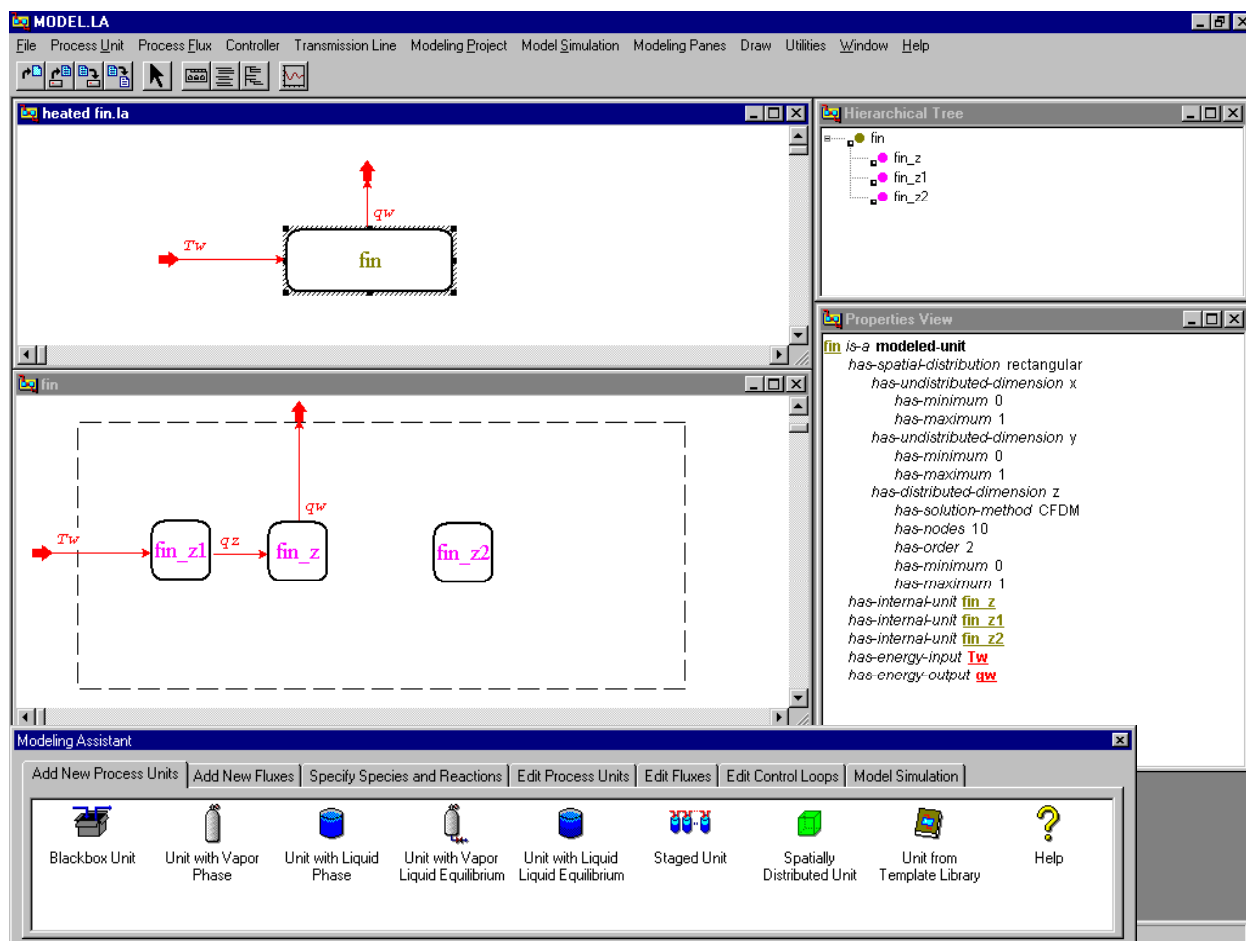


Figure 5-10: 1-D Distributed Heated Fin Example

5.2.3 Chemical Characterization

The chemical characterization of a phenomena-based model is characterized by declarations of chemical species, chemical reactions, material-contents, and phases. Chemical species and chemical reactions are declared by selecting the *Declare Chemical Species* and *Declare Chemical Reactions* options, respectively, on the *Specify Species and Reactions* tab of the *Modeling Assistant* (shown at the bottom of Figure 5-11).

- **Chemical Species:** In MODEL.LA, chemical species are selected from database using the *Project Species Selection Dialog* shown in Figure 5-11. Chemical species in the database are listed in the *DATABASE* group, and chemical species currently declared in the phenomena-based model are listed in the *PROJECT* group. In Figure 5-11, four chemical species have been declared for the model: *acetic acid*, *water*, *1-butanol*, and *n-butyl acetate*. In addition to the

chemical species listed in the database (which currently contains the set of over 1400 chemical compounds from the DIPPR database compiled by AIChE (1982)), new chemical species may be declared by the modeler for inclusion in a model. Once chemical species are added to a phenomena-based model, the modeler may assign these to the individual modeled-units, material-contents, and phases in the model, and use them to define chemical reactions.

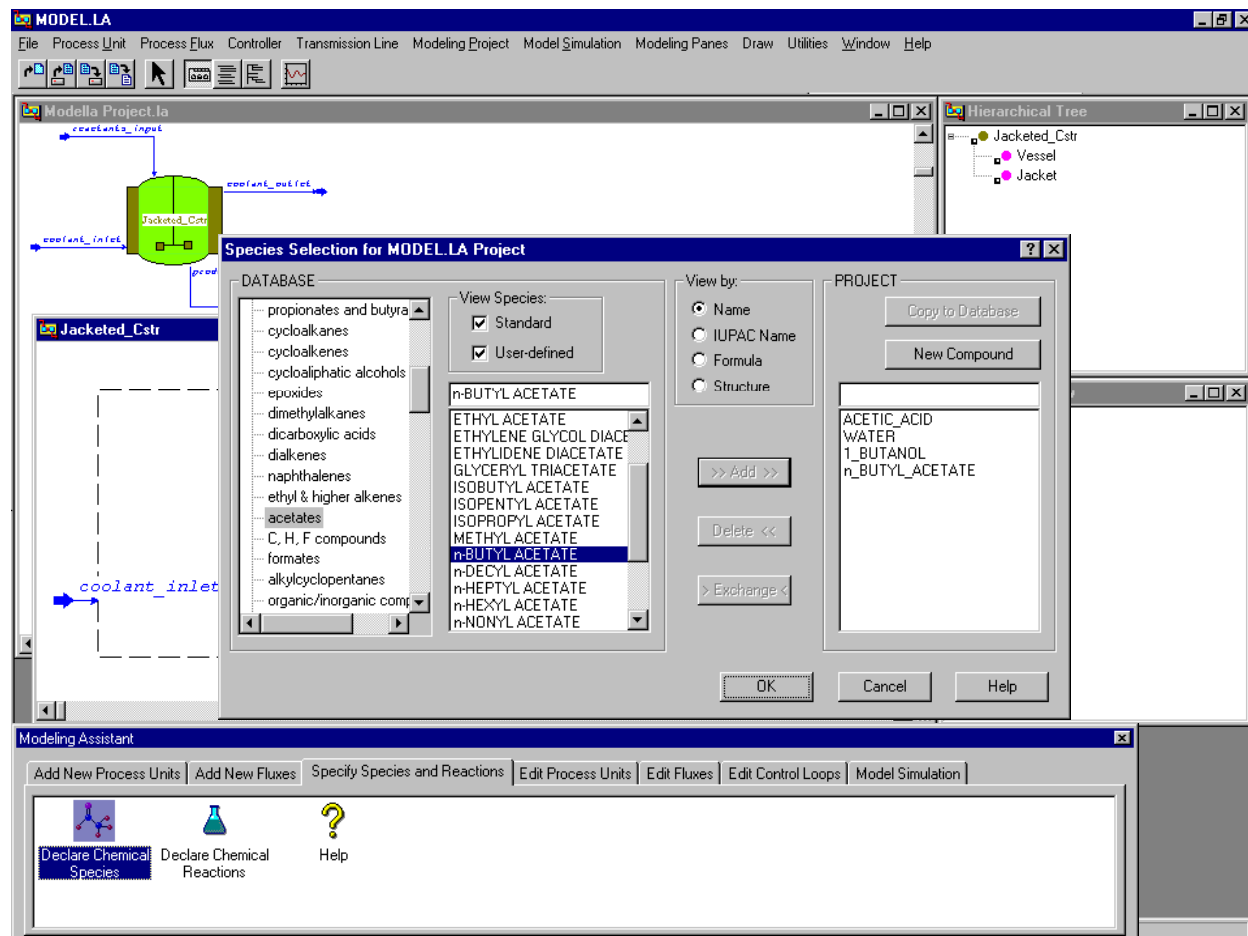


Figure 5-11: Project Species Selection Dialog

- Chemical Reactions:** Chemical reactions in a phenomena-based model are declared using the *Project Reaction Dialog*, shown in Figure 5-12, where participating species, their stoichiometry, and other characterizations are declared. In Figure 5-12, the reversible reaction of *acetic acid* and *1-butanol* to form *water* and *n-butyl acetate* has been declared. Once chemical reactions have been declared in a phenomena-based model, the modeler may assign these to the individual modeled-units and phases in the model.

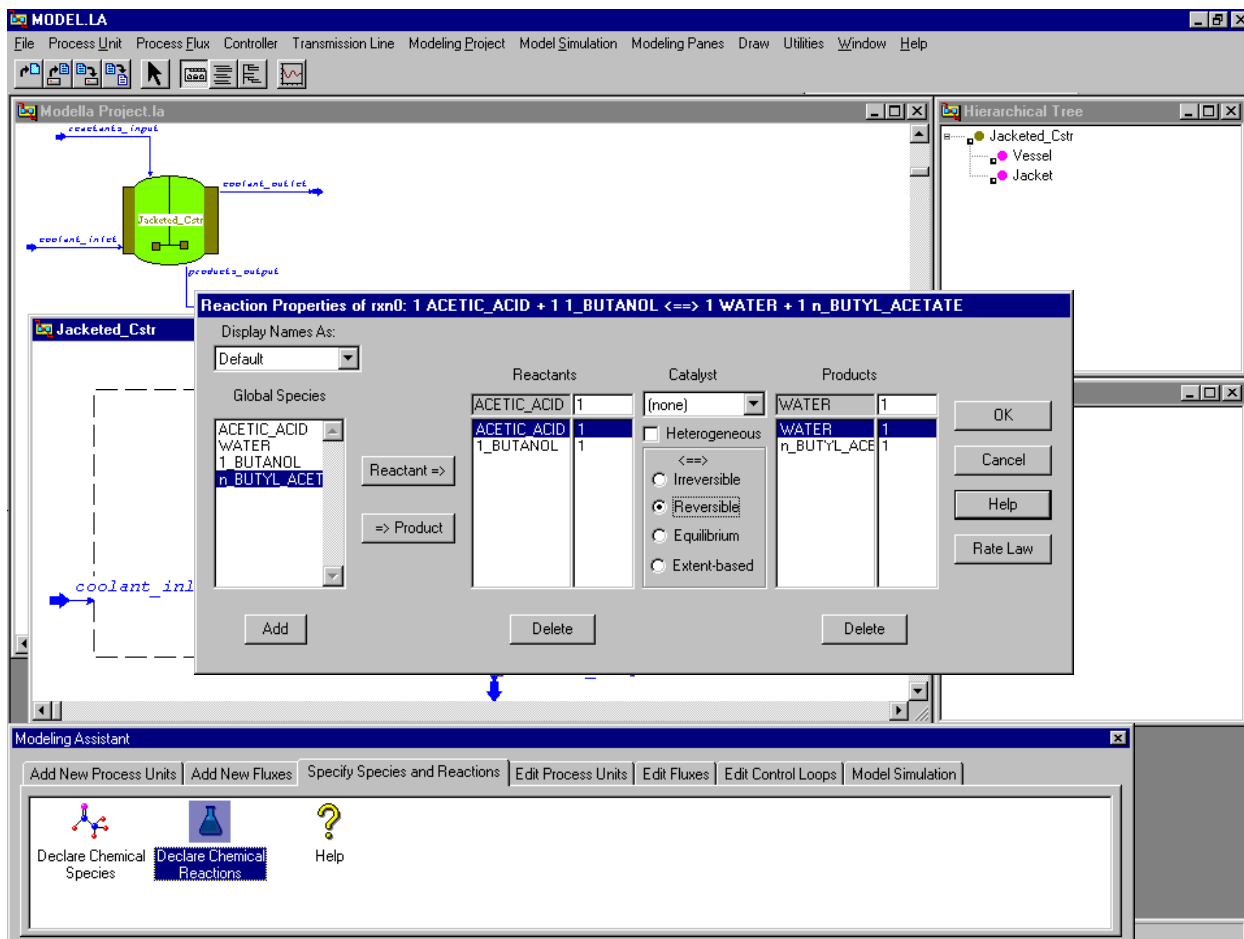


Figure 5-12: Project Reaction Dialog

Chemical species and reactions are assigned to a modeled-unit by selection of the *Assign Reactions and Species* option on the *Edit Process Units* of the *Modeling Assistant*. This activates the *Modeled-Unit Chemical Content Characterization Dialog*, shown in Figure 5-13. Here, chemical species and reactions declared for the phenomena-based model may be assigned to a particular modeled-unit.

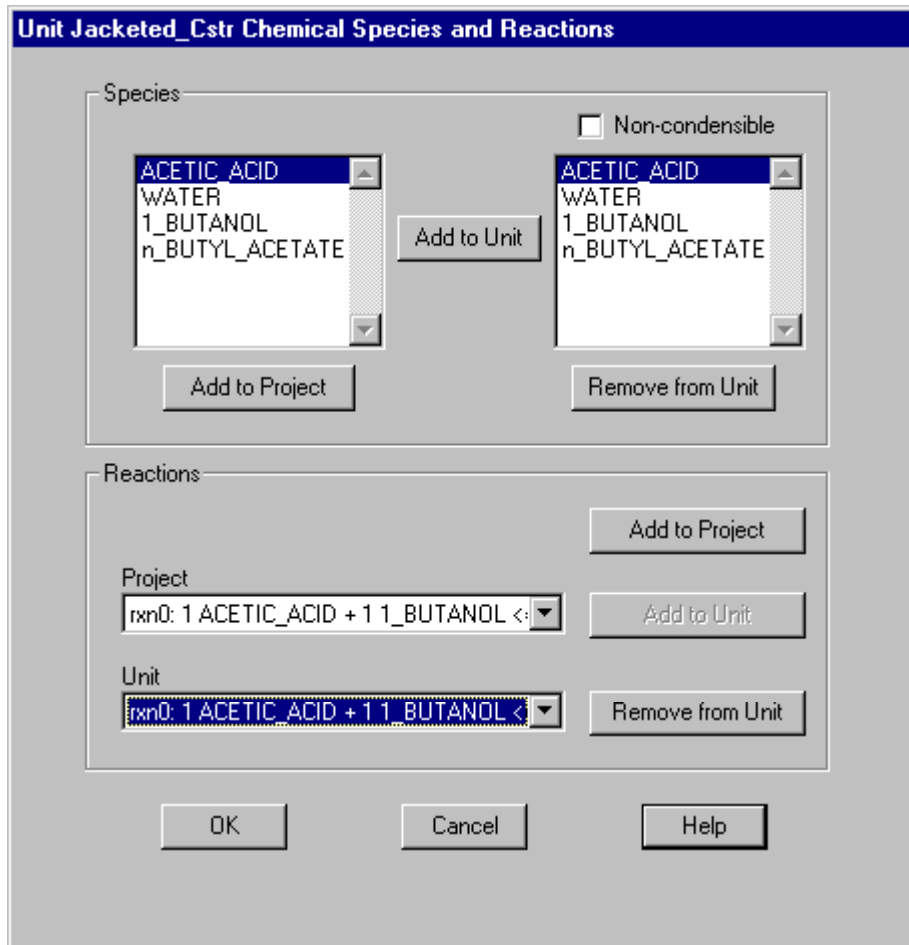


Figure 5-13: Modeled-Unit Chemical Content Characterization Dialog

- Material-Contents:** A material-content is declared for a modeled-unit by selection of the *Specify Material Content* on the *Edit Process Units* tab of the *Modeling Assistant*. This activates the *Material Content Declaration Dialog* (as shown in Figure 5-14). Here the chemical species, geometry (Figure 5-15), phases, and allocation of boundary fluxes to phases (Figure 5-16) for a material-content are declared.

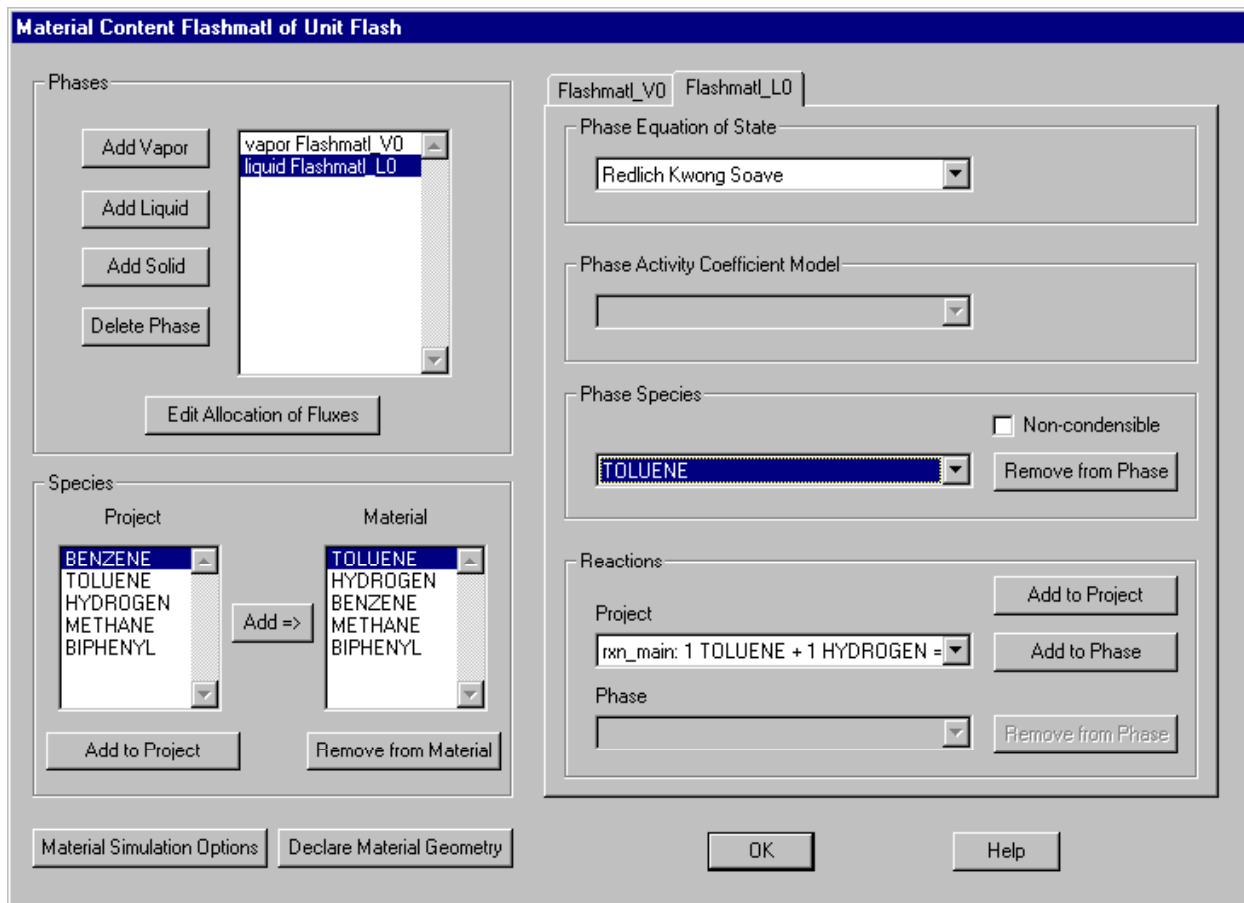


Figure 5-14: Material-Content Declaration Dialog

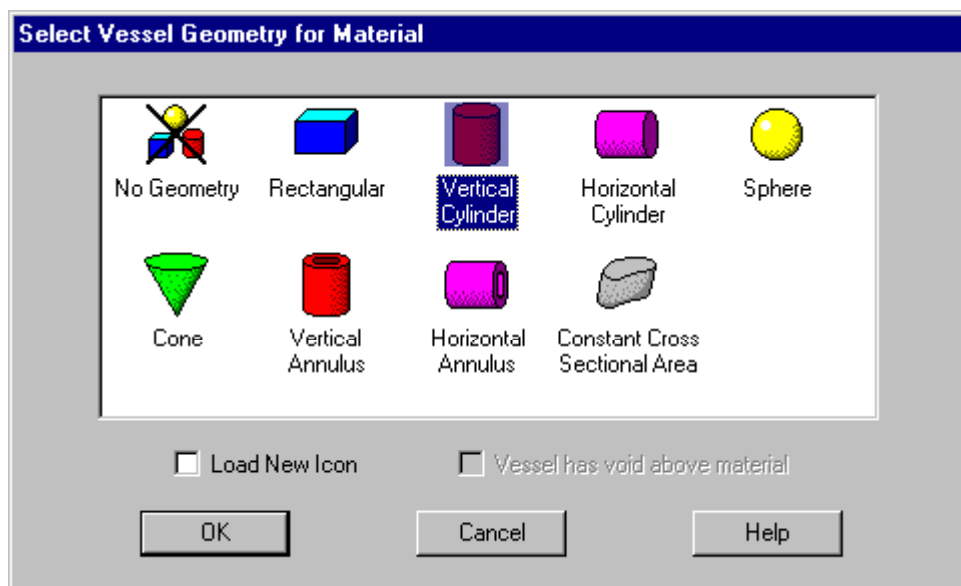


Figure 5-15: Material-Content Geometry Declaration Dialog

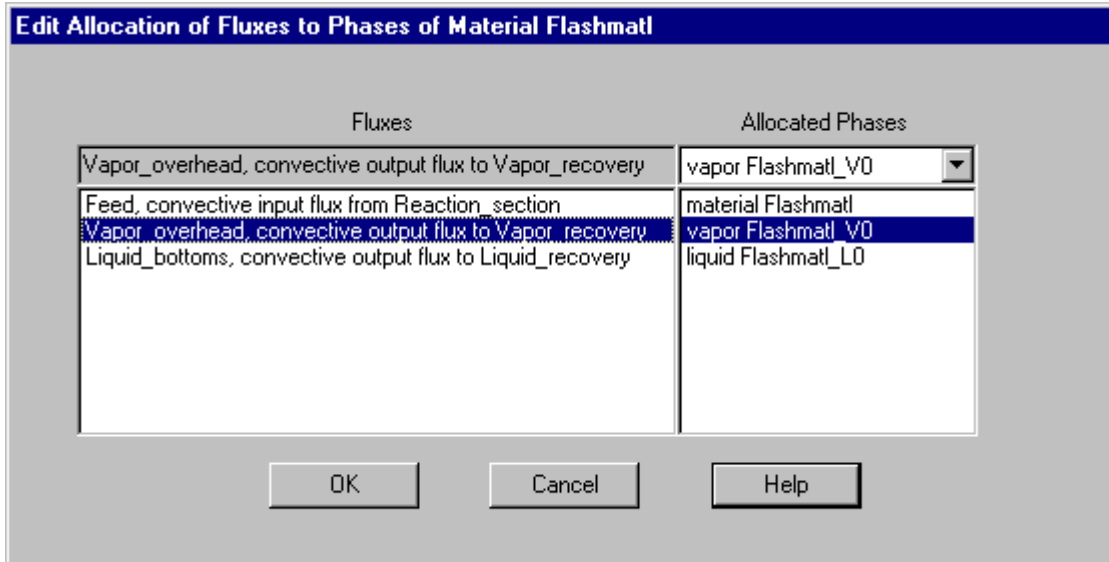


Figure 5-16: Material-Content Flux Allocation Dialog

The declaration of a geometry for a material-content introduces relationships into the mathematical model that express the height of the phases that compose the material-content as a function of material volume. The available geometrical configurations are summarized in Table 5-3. Declaration of a geometry allows the allocation of boundary fluxes (e.g., convective outputs streams) to be determined by geometry during model simulation. In such cases, a *port height* for the boundary flux is designated. The properties of the material transported by the flux is determined by the phase whose height overlaps that of the port height. For example, in Figure 5-17 a material-content is shown with two phases, *phase-1* and *phase-2*, and three convective output boundary fluxes, *flux-1*, *flux-2*, and *flux-3*, allocated to the vessel geometry. The material transported by *flux-1* is characterized by the properties of *phase-1* since the port height of *flux-1* overlaps that of *phase-1*. Similarly, the material transported by *flux-2* is characterized by the properties of *phase-2*. However, no material is transported by *flux-3* since the port height of the flux exceeds that of the entire material-content. Note that this allocation is not necessarily known *a priori* and must be determined during model simulation. Thus, the allocation of boundary fluxes to vessel geometry introduces conditional discontinuities into the mathematical model.

- **Phases:** Phases declared for a material-content are also assigned chemical reactions and species using appropriate tab on the right side of the *Material-Content Declaration Dialog* (shown in Figure 5-14).

Table 5-3: Geometric Vessel Configurations for a Material-Content

<u>Geometry</u>	<u>Geometric Relationships</u> V =vessel volume, $H_{material}$ =material-content height $V_{material}$ = material-content volume	<u>Vessel Parameters</u>
Rectangular	$V = LWH, \quad H_{material} = H$	L =length, W =width, H =height
Vertical cylinder	$V = \pi R^2 H, \quad H_{material} = H$	R =radius, H =height
Horizontal cylinder	$V = \pi R^2 L, \quad H_{material} = 2R$	R =radius, L =length
Horizontal cylinder (with vessel void)	$V = \pi R^2 L,$ $V_{material} = L * \text{circlesegment}(H_{material}, R),$ $\text{circlesegment}(h, r) \equiv (h - r)\sqrt{2rh - h^2}$ $+ r^2 \arcsin\left(\frac{h}{r} - 1\right) + \frac{\pi r^2}{2}$	R =radius, L =length
Sphere	$V = \frac{4\pi}{3} R^3, \quad H_{material} = 2R$	R =radius
Sphere (with vessel void)	$V = \frac{4\pi}{3} R^3, \quad V_{material} = \frac{\pi}{3} H_{material}^2 (3R - H_{material})$	R =radius
Cone	$V = \frac{\pi}{3} R^2 H, \quad H_{material} = H$	R =radius, H =height
Vertical annulus	$V = \pi(R_2^2 - R_1^2)H, \quad H_{material} = H$	R_1 =inner radius, R_2 =outer radius, H =height
Horizontal annulus	$V = \pi(R_2^2 - R_1^2)L, \quad H_{material} = 2R$	R_1 =inner radius, R_2 =outer radius, L =length
Horizontal annulus (with vessel void)	$V = \pi(R_2^2 - R_1^2)L,$ $V_{material} = L * [\text{circlesegment}(H_{material}, R_2) - A_{inner}],$ $\text{circlesegment}(h, r) \equiv (h - r)\sqrt{2rh - h^2}$ $+ r^2 \arcsin\left(\frac{h}{r} - 1\right) + \frac{\pi r^2}{2},$ $A_{inner} = \begin{cases} 0, & 0 \leq H_{material} < R_2 - R_1; \\ \text{circlesegment}(H_{material} - (R_2 - R_1), R_1), & R_2 - R_1 \leq H_{material} < R_2 + R_1; \\ \pi R_1^2, & R_2 + R_1 \leq H_{material} \leq 2R_2; \end{cases}$	R_1 =inner radius, R_2 =outer radius, L =length
Slanted cylinder with constant cross sectional area	$V = \frac{A \cdot H}{\sin \theta}, \quad H_{material} = H$	A =cross sectional area, H =height, θ = slant angle

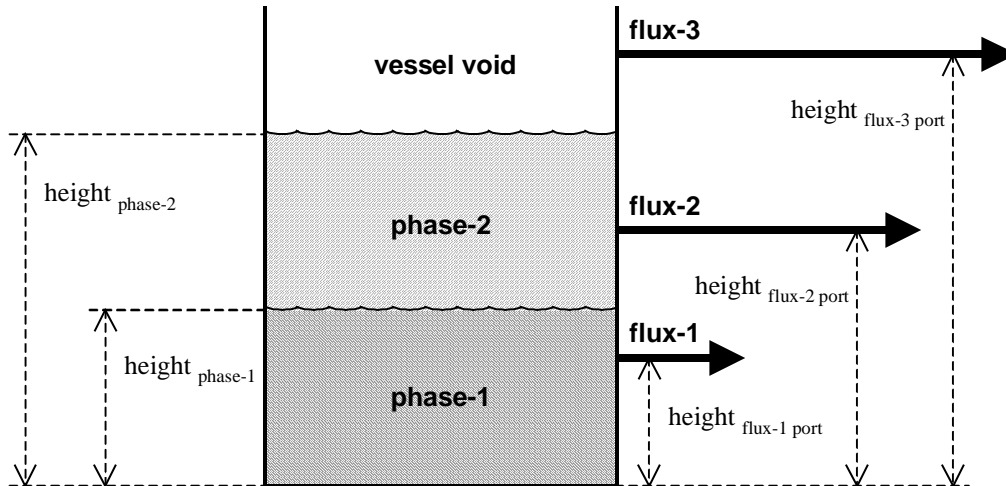


Figure 5-17: Example Vessel Geometry and Flux Allocation

5.2.4 Phenomena-Based Mechanistic Characterization

- **Phases:** The physical state of a phase is selected when the phase is declared. Thermodynamic characterizations of phases are also declared in the *Material-Content Declaration Dialog* (shown in Figure 5-14). Either a P-V-T equation of state or, for incompressible phases, an activity coefficient model may be selected to characterize the thermodynamic behavior of a phase.
- **Fluxes:** Fluxes are characterized by selecting the *Edit Flux Properties* option in the *Edit Fluxes* tab on the *Modeling Assistant* (as shown in Figure 5-18).

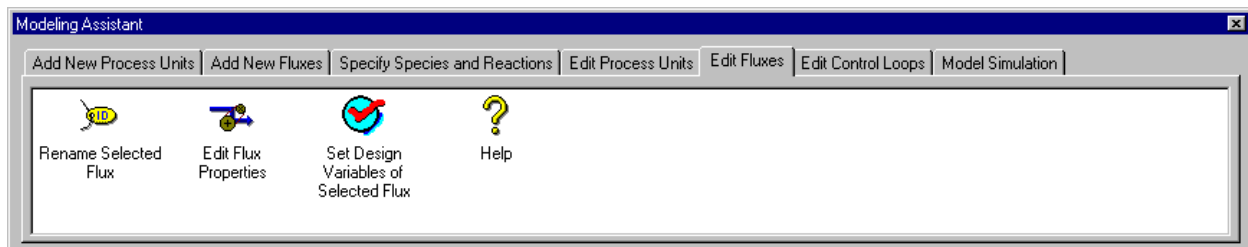


Figure 5-18: Modeling Assistant: Edit Fluxes Tab

This activates the appropriate flux characterization dialog for the selected convective, energy, or selected chemical species flux, as shown in Figure 5-19, Figure 5-20, and Figure 5-21, respectively.

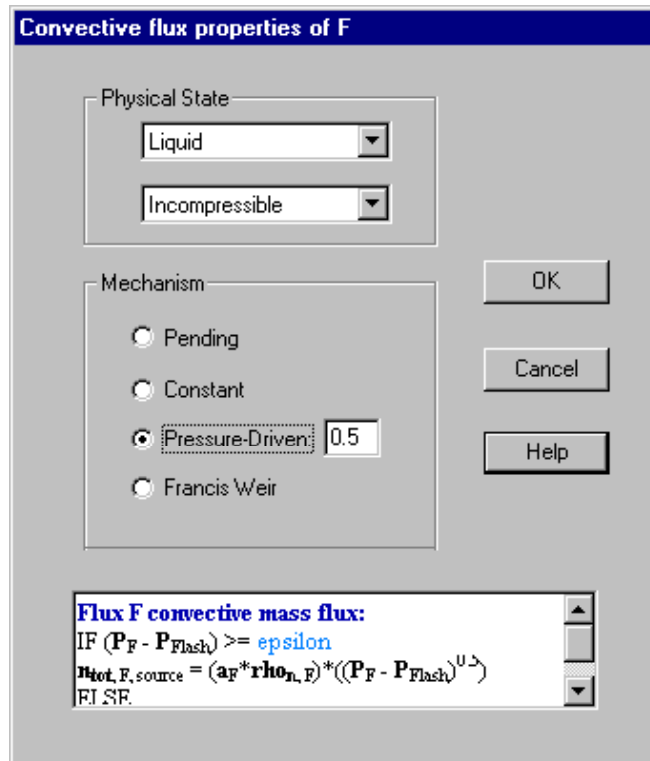


Figure 5-19: Convective Flux Characterization Dialog

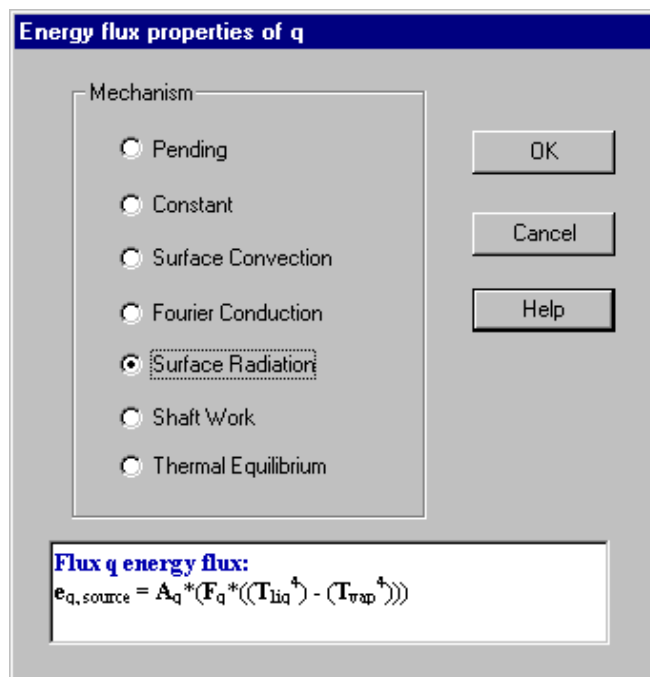


Figure 5-20: Energy Flux Characterization Dialog

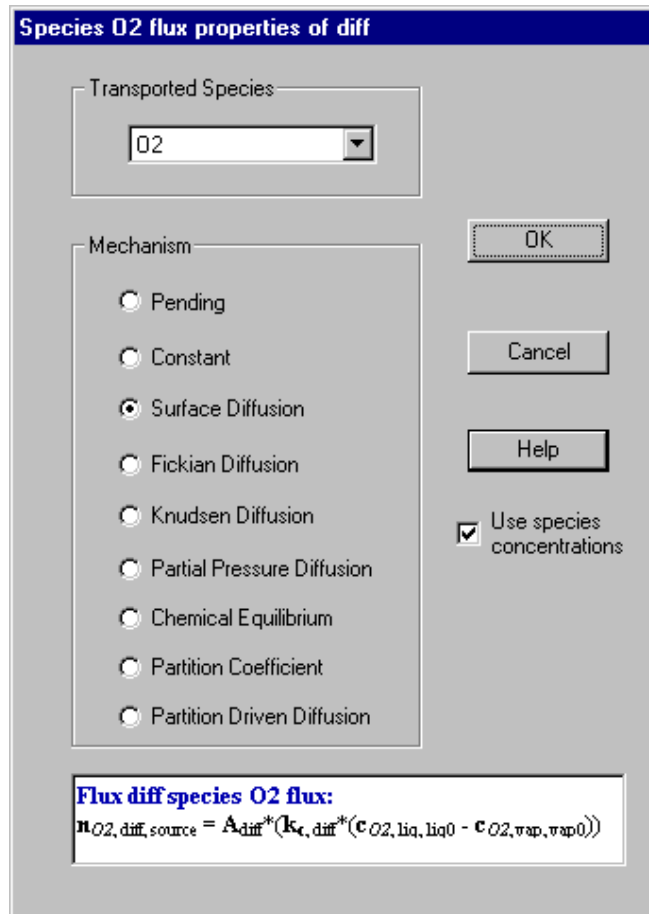


Figure 5-21: Species Flux Characterization Dialog

- **Reactions:** Kinetic rate laws are declared for rate-based (i.e., reversible or irreversible) reactions by selecting the *Rate Law* button on the *Project Reaction Dialog* (shown in Figure 5-12). This activates the *Reaction Rate Law Dialog* shown in Figure 5-22. Rate laws are declared separately for forward and reverse rates of reaction. Template forms available for reaction rate laws are listed in Table 5-4. If another form of a kinetic rate law is required, it may be declared as a user-defined equation in the *Operations Manager*.

Reaction Rate of rxn0: 1 ACETIC_ACID + 1 1_BUTANOL <=> 1 WATER + 1 n_BUTYL_ACETATE

Forward Reaction Rate | Reverse Reaction Rate

Rate Law Denominator Terms

none two three four

Numerator

Rate Constant

Pending

Constant k

Arrhenius

Modified Arrhenius: n =

Participants	
ACETIC_ACID	2
ACETIC ACID	2
1_BUTANOL	1
WATER	0
n_BUTYL_ACETA	0

OK

Cancel

Help

Rate Law: Use partial pressures

Phase Unitmatl_VD reaction rxn0 forward rate:

$$r_{rxn0_forw_{Unit,vap0}} = (k_{constforw_{rxn0}} * (C_{ACETIC_ACID, Unit,vap0}^2)) * (C_{1_BUTANOL, Unit,vap0})$$

Phase Unitmatl_VD reaction rxn0 reverse rate:

$$r_{rxn0_back_{Unit,vap0}} = (k_{constback_{rxn0}} * (C_{WATER, Unit,vap0})) * (C_{n_BUTYL_ACETATE, Unit,vap0})$$

Figure 5-22: Project Reaction Rate Law Dialog

Table 5-4: MODEL.LA Reaction Rate Law Templates

<p>For reaction:</p> <p>Define:</p> <p>where:</p>	$a_1 R_1 + a_2 R_2 + \dots + a_r R_r \leftrightarrow b_1 P_1 + b_2 P_2 + \dots + b_p P_p$ $F_i(\text{reaction}) = c_{R_1}^{n_1} * c_{R_2}^{n_2} * \dots * c_{R_r}^{n_r} * c_{P_1}^{m_1} * c_{P_2}^{m_2} * \dots * c_{P_p}^{m_p}$ <p>c_{R_i} represents the molar concentration (or partial pressure) of reactant R_i, c_{P_i} represents the molar concentration (or partial pressure) of product P_i, and n represents an exponent specified by the modeler.</p>
<p>For both forward and reverse reaction, rate law may be specified as one of:</p> <ul style="list-style-type: none"> • no denominator terms: $rate = k_1 * F_1(\text{reaction})$ • two denominator terms: $rate = \frac{k_1 * F_1(\text{reaction})}{1 + k_2 * F_2(\text{reaction})}$ • three denominator terms: $rate = \frac{k_1 * F_1(\text{reaction})}{1 + k_2 * F_2(\text{reaction}) + k_3 * F_3(\text{reaction})}$ • four denominator terms: $rate = \frac{k_1 * F_1(\text{reaction})}{1 + k_2 * F_2(\text{reaction}) + k_3 * F_3(\text{reaction}) + k_4 * F_4(\text{reaction})}$ 	
<p>Rate constant models may be specified for each k_i:</p> <ul style="list-style-type: none"> • Constant: $k_i = const$ • Arrhenius: $k_i = A_i * \exp\left(\frac{-E_i}{RT}\right)$ • Modified Arrhenius: $k_i = A_i * T^{n_i} * \exp\left(\frac{-E_i}{RT}\right)$ 	

5.2.5 Phenomena-Based Model Summary

At any point during the modeling activity, a complete summary of all assumptions and declarations made for a phenomena-based model is available for display in the *Project Data Summary Dialog* (as illustrated in Figure 5-23).

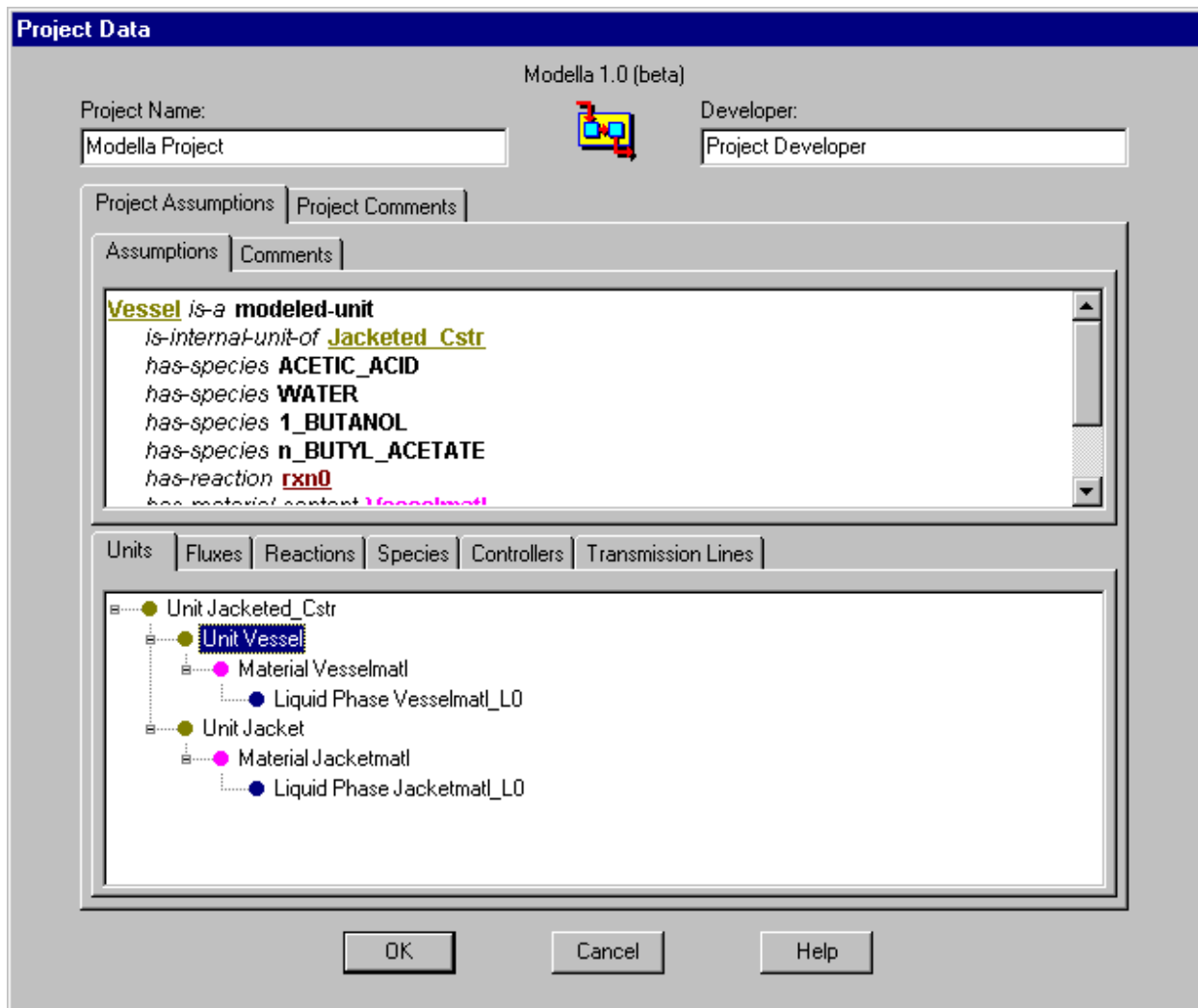


Figure 5-23: Project Data Summary Dialog

This dialog groups all modeling assumptions by element type. Here, comments for any modeling elements may be documented and edited. Hypertext is used to facilitate navigation of the phenomena-based assumptions.

In addition, the assumptions of any modeled-unit in a phenomena-based model may be saved, along with its topological and hierarchical structure, as a template in a model library for

reuse in the same or any other modeling project. A model template is added to a model by selecting the *Unit from Template Library* icon on the *Add Process Units* tab of the *Modeling Assistant* and dragging it to a flowsheet. This activates the *Modeled-Unit Template Selection Dialog* (Figure 5-24), where the appropriate template library file is selected.

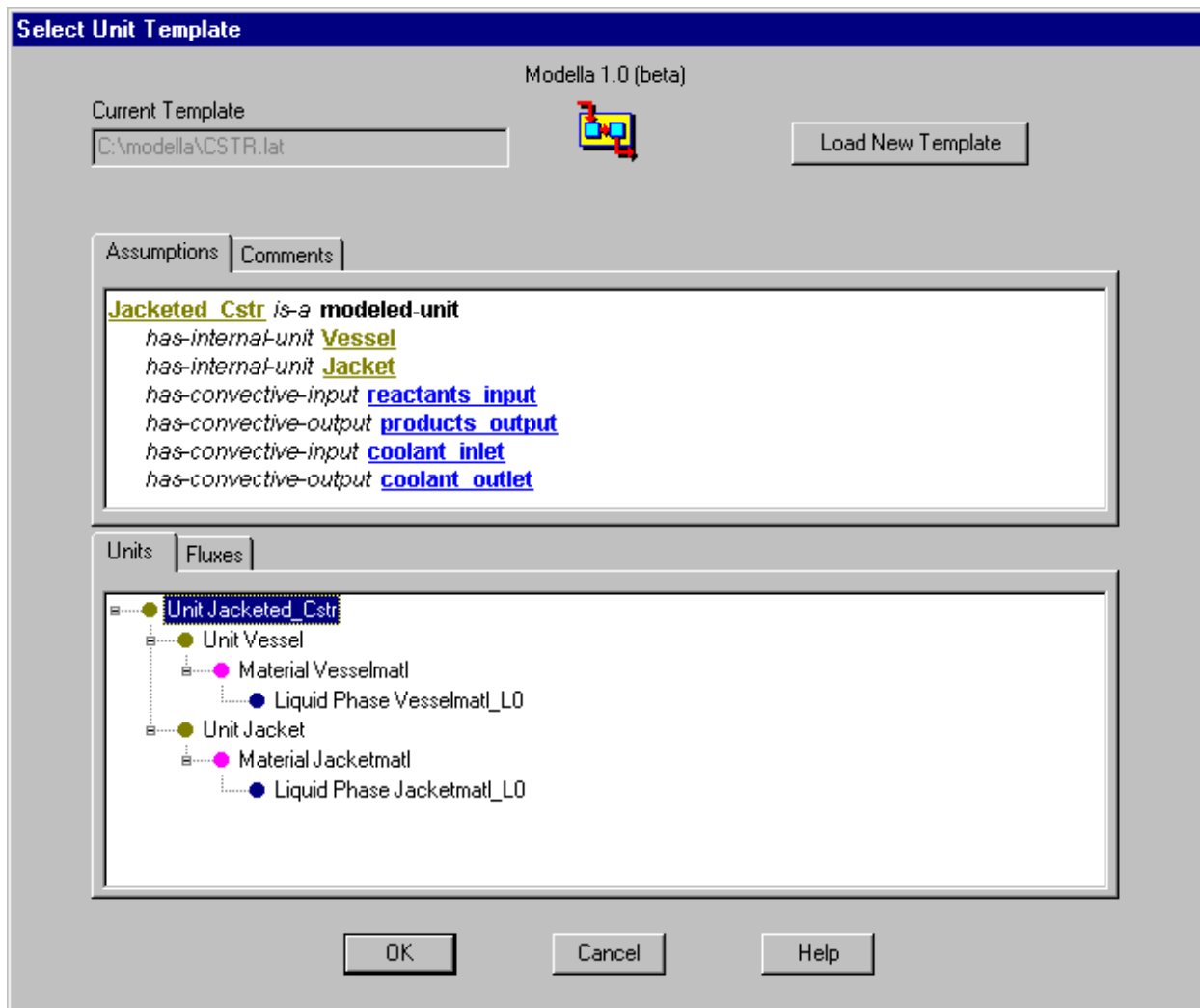


Figure 5-24: Modeled-Unit Template Selection Dialog

5.2.6 Mathematical Model Derivation

When the modeler completes the phenomena-based model description, MODEL.LA analyzes model for any inconsistencies or incompleteness (e.g., missing assumptions, inconsistent species allocation, and unallocated fluxes). If any are detected, they are described to the modeler in the *Model Inconsistency Dialog* (illustrated in Figure 5-25). Such inconsistencies must be remedied

before a mathematical model can be derived.

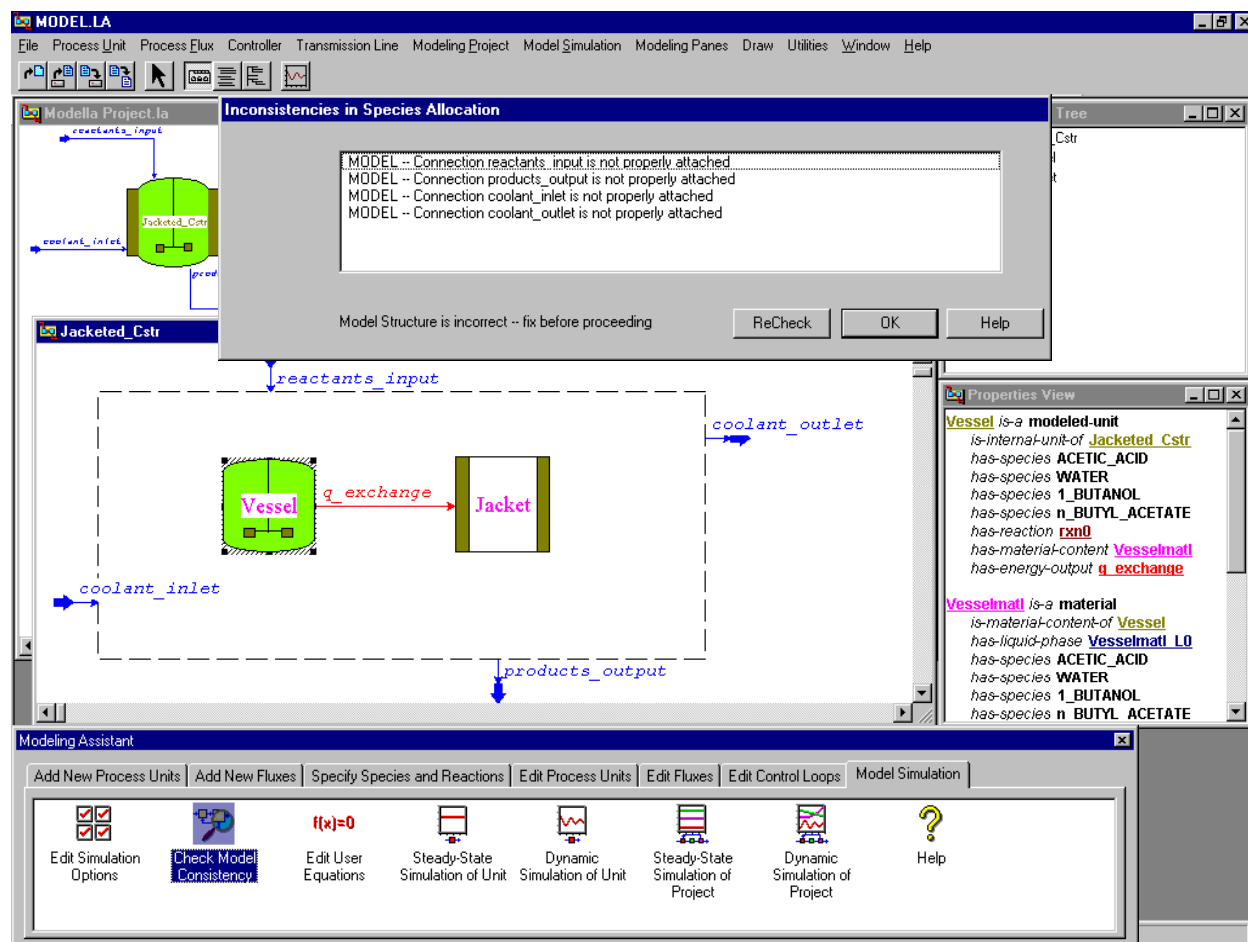


Figure 5-25: Model Inconsistency Dialog

When the phenomena-based model is completed, a derivation context for the mathematical model is specified using the *Simulation Options Dialog* shown in Figure 5-26. The level of detail for a hierarchical model may also be specified by selecting the desired resolution in the *Hierarchical Tree*. The model equations are then generated from first principles based on the modeling logic operators described in the previous chapter. The model equations which are automatically derived include conservation equations, equilibrium relationships, transport mechanisms, rate laws, geometry constraints, extensive property decompositions, etc. These equations are supplemented with thermodynamic and physical property correlations constructed by the *MODEL.LA Properties Manager*, and additional constraints from the *Operations Manager*. The model equations derived for the *Jacketed_CSTR* model in this section are given in Appendix D.

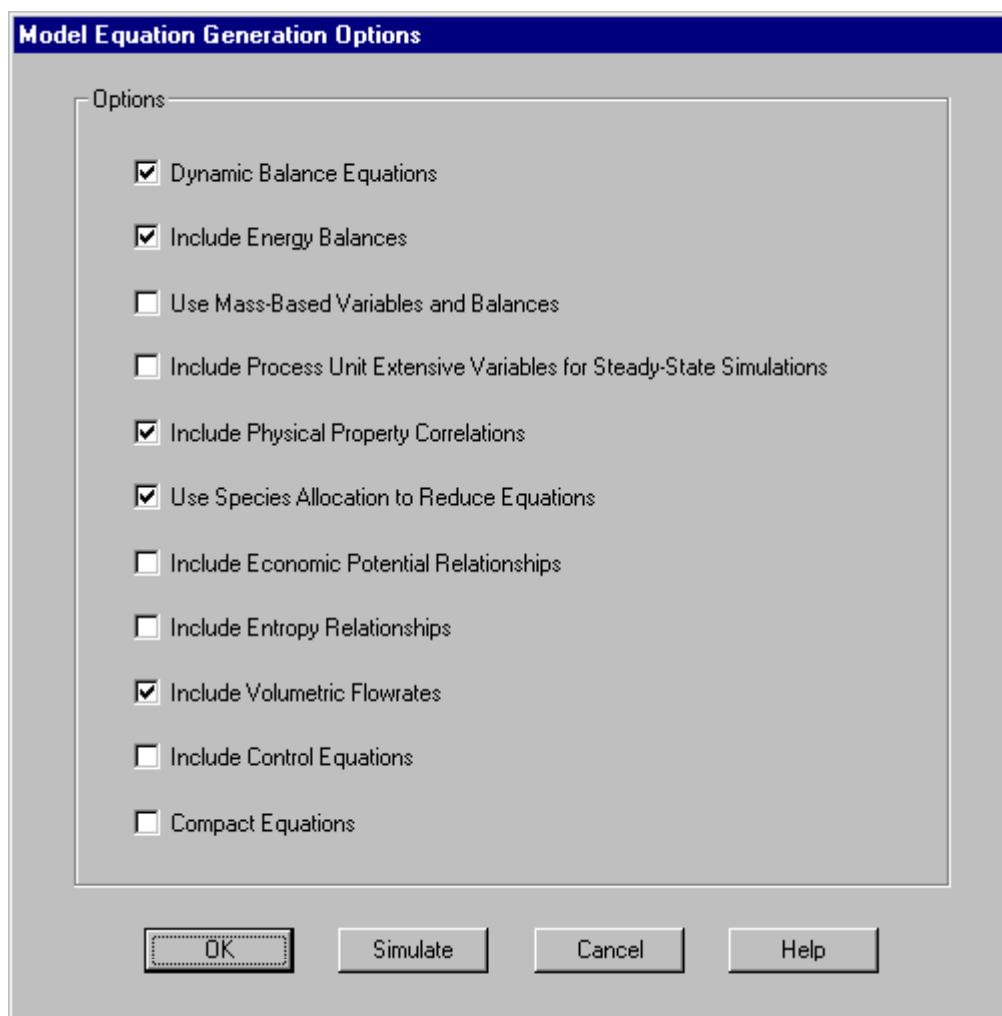


Figure 5-26: Simulation Options Dialog

5.3 Properties Manager

The MODEL.LA *Properties Manager* constructs correlations that describe thermodynamic and physical properties of mixtures (i.e., phases) based on the selected state, equation of state or activity coefficient model, and selected chemical species for each phase. These properties are expressed as equation-based functions of pressure, temperature, and composition, and are appended to the model equations derived by the *Model Generator*. These correlations are dependent on the constant and temperature-dependent pure species properties and binary interaction parameters declared in the database. Supplementary details regarding the MODEL.LA *Properties Manager* are given in Appendix B.

5.3.1 Pure Species Properties

The MODEL.LA *Properties Manager* currently accesses the DIPPR database, which contains data on 36 constant and temperature-dependent properties of over 1400 chemical species. A modeler may add additional species to the database as necessary. Each species is characterized by a set of *identification properties* (Figure 5-27), *constant properties* (Figure 5-28), *temperature-dependent properties* (Figure 5-29), and *UNIFAC groups* (Figure 5-30) that are displayed and may be edited using dialogs.

5.3.2 Binary Interaction Parameters

In addition to pure species property data, the MODEL.LA *Properties Manager* stores data on binary interaction parameters for equations of state (Figure 5-31) and activity coefficient models (Figure 5-32). These data are also displayed and edited using dialogs.

5.3.3 Material Behavior Analysis

The MODEL.LA *Properties Manager* provides several features to facilitate the study of the behavior of phases that are declared by the modeler, independently of model equations derived by the *Model Generator*. The thermodynamic and physical properties of these phases are based on assigned species and assumed equation of state or activity coefficient models in a phenomena-based model, along with pure component data and binary interaction parameters from the database. The *Phase Equilibrium Calculations Dialog* (Figure 5-33) of the *Properties Manager* is used to perform flash calculations for multiphase systems. The *Phase Properties Dialog* (Figure 5-34) displays and plots correlations for physical and thermodynamic properties of single phases. Finally, the *Phase Diagram Dialog* (Figure 5-35) is used to generate phase diagrams for multiphase systems.

Project Species Properties [?] [X]

Compound:

Identification | **Properties** | Correlations | UNIFAC

Index:

Name:

IUPAC Name:

CAS #:

Formula:

Structure:

Chemical Group:

Figure 5-27: Species Database: Identification Properties

Project Species Properties [?] [X]

Compound:

Identification | **Properties** | Correlations | UNIFAC

Molecular Weight	<input type="text" value="116.16"/>	kg/kmol
Critical Temperature	<input type="text" value="579.15"/>	K
Critical Pressure	<input type="text" value="3110000"/>	Pa
Critical Volume	<input type="text" value="0.389"/>	m3/kmol
Critical Compressibility Factor	<input type="text" value="0.251"/>	
Melting Point (1 atm)	<input type="text" value="199.65"/>	K
Triple Point Temperature	<input type="text" value="199.65"/>	K
Triple Point Pressure	<input type="text" value="0.143467"/>	Pa
Normal Boiling Point (1 atm)	<input type="text" value="399.15"/>	K
Liquid Molar Volume (at 298.15K)	<input type="text" value="0.132593"/>	m3/kmol
Enthalpy of Formation (ideal gas)	<input type="text" value="-485600000"/>	J/kmol
Gibbs Energy of Formation (ideal gas)	<input type="text" value="-312600000"/>	J/kmol
Absolute Entropy (ideal gas)	<input type="text" value="442500"/>	J/kmol-K
Enthalpy of Fusion	<input type="text" value="14400000"/>	J/kmol
Net Enthalpy of Combustion	<input type="text" value="-3280000000"/>	J/kmol
Acentric Factor	<input type="text" value="0.410061"/>	
Solubility Parameter (at 298.15K)	<input type="text" value="17590"/>	(J/m3) ^{1/2}
Dipole Moment	<input type="text" value="6.14e-30"/>	c-m
van der Waals Reduced Volume	<input type="text" value="0.07323"/>	m3/kmol
van der Waals Area	<input type="text" value="1049000000"/>	m2/kmol
Lower Flammability Limit	<input type="text" value="1.7"/>	vol% in air
Upper Flammability Limit	<input type="text" value="7.6"/>	vol% in air
Auto-ignition Temperature	<input type="text" value="694"/>	K

Figure 5-28: Species Database: Constant Properties

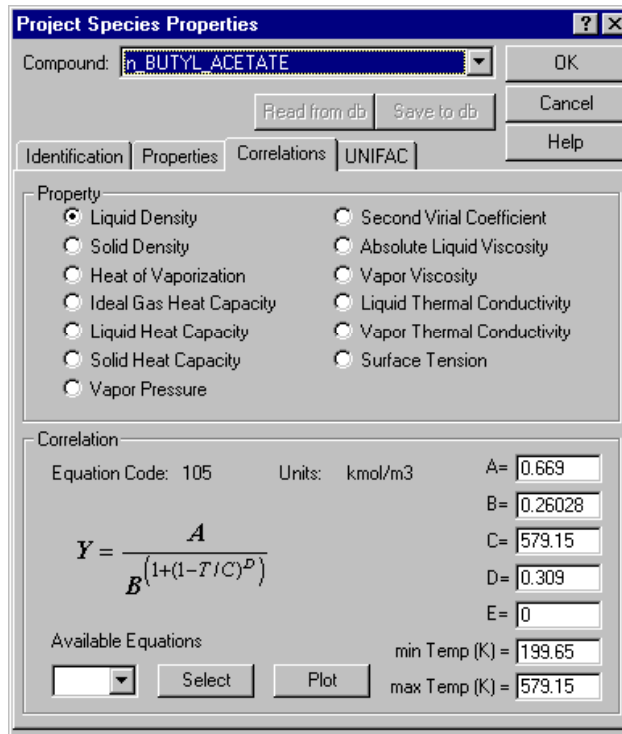


Figure 5-29: Species Database: Temperature Dependant Properties

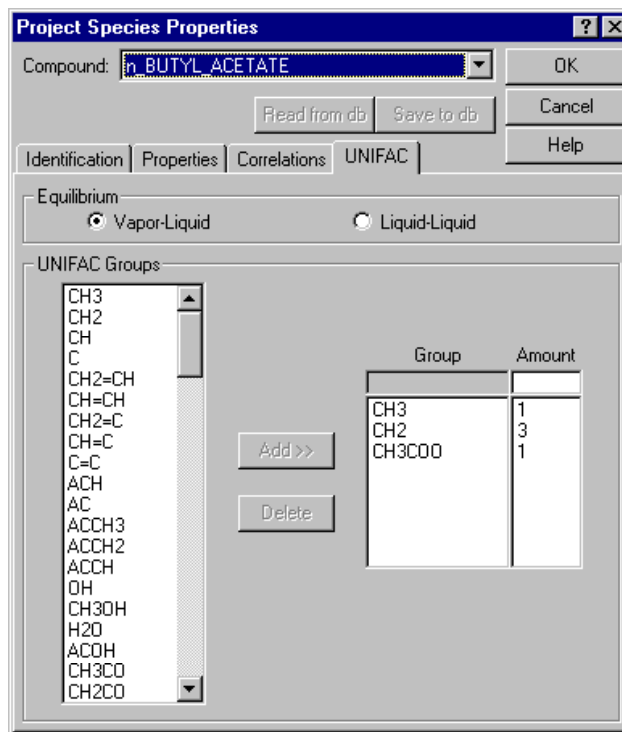


Figure 5-30: Species Database: UNIFAC Groups Properties

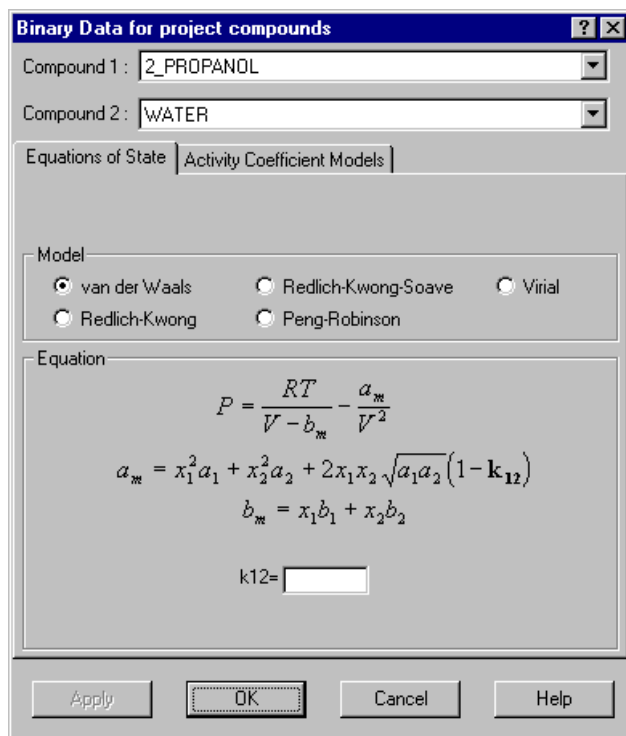


Figure 5-31: Species Database: Binary Interaction Parameters for Equations of State

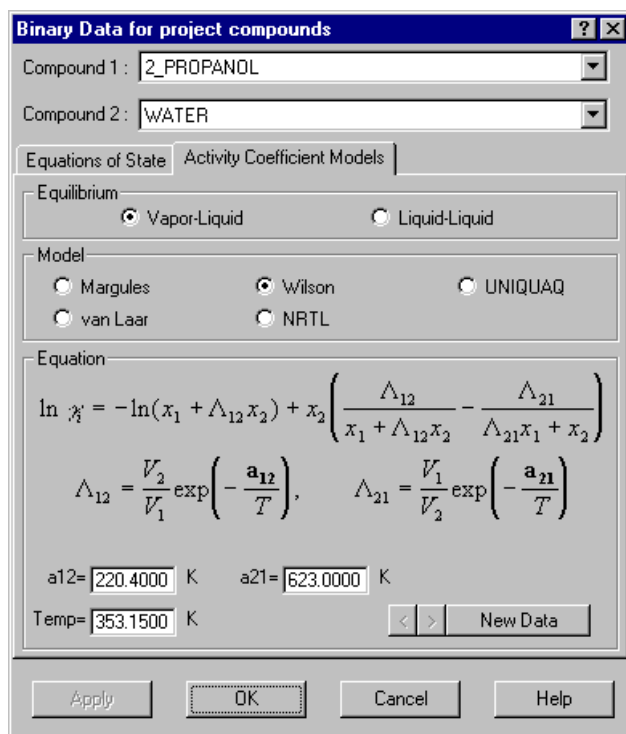


Figure 5-32: Species Database: Binary Interaction Parameters for Activity Coefficient Models

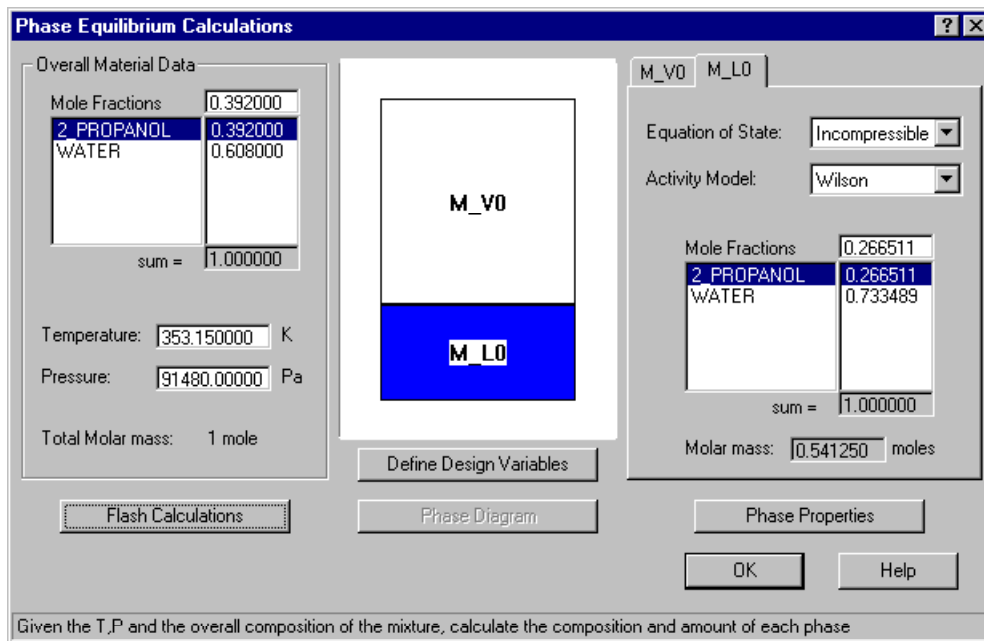


Figure 5-33: Phase Equilibrium Calculations Dialog

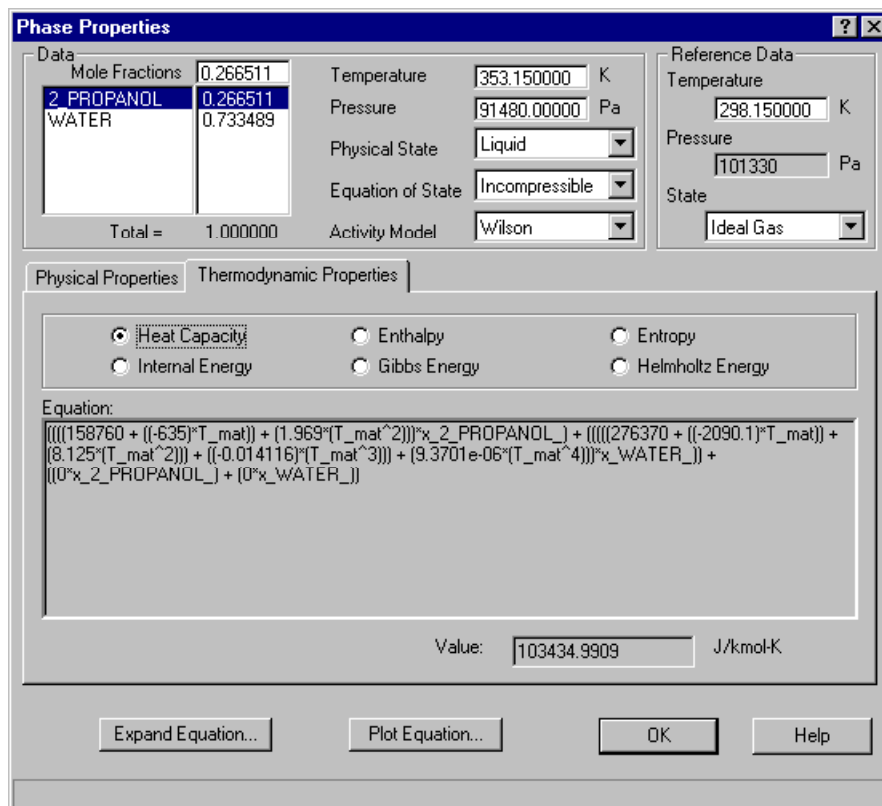


Figure 5-34: Phase Properties Calculations Dialog

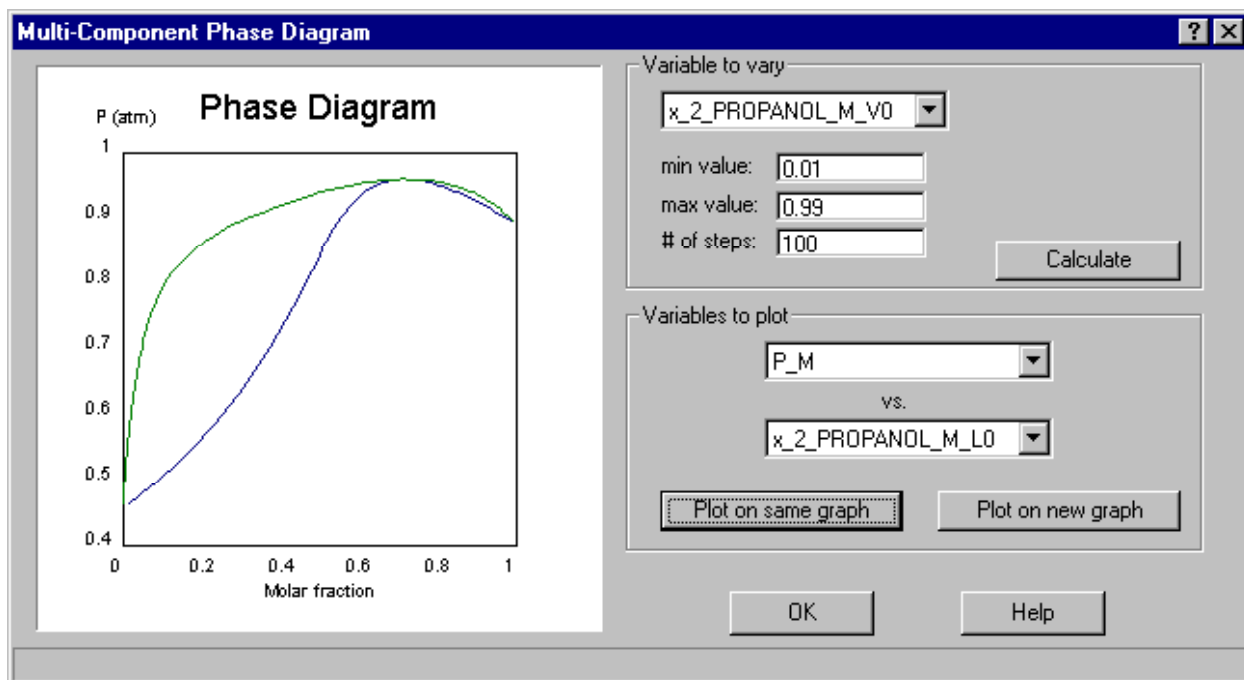


Figure 5-35: Phase Diagram Dialog

5.4 Operations Manager

The MODEL.LA modeling language is designed to represent chemical processes in terms of interacting physicochemical phenomena. From this phenomena-based description, the corresponding model equations are derived from first principles whenever a mathematical model is required. There are several situations, however, where additional relationships are needed to describe certain modeling objectives and complete the mathematical model. The MODEL.LA *Operations Manager* provides facilities for the modeler to introduce these additional relationships integrated within the context of the phenomena-based model description. Supplementary details regarding the MODEL.LA *Operations Manager* are given in Appendix C.

In MODEL.LA, the modeler may introduce four different types of relationships in the *Operations Manager*, including:

1. *User Equations*,
2. *Process Controllers*,
3. *External Models*, and
4. *Operational Schedules*.

The declaration of each of these types of relationships will be discussed in the remainder of this

section.

5.4.1 User Equations

User Equations represent generic equations added to the mathematical model by the modeler. Such equations are declared using the *User-Entered Equation Dialog* illustrated in Figure 5-36.

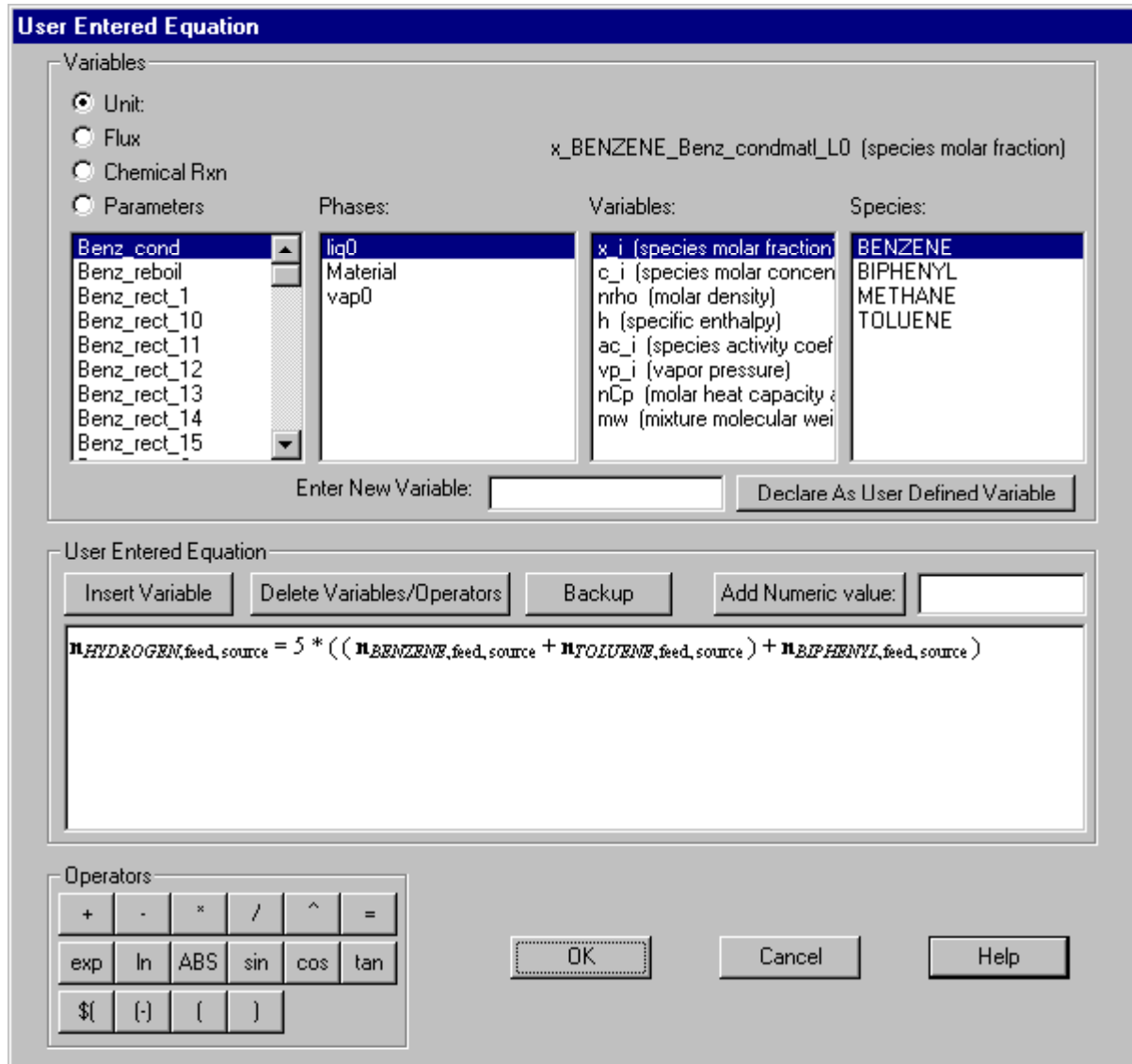


Figure 5-36: User-Entered Equation Dialog

Here the modeler may construct arbitrary mathematical equations that interrelate variables associated with the phenomena-based model and also additional variables, parameters, and numerical constants that the modeler defines. These equations are stored symbolically and

subsequently appended to the model equations derived from the phenomena-based representation.

5.4.2 Process Controllers

While equations that introduce continuous process control laws may be defined as *User Equations*, the MODEL.LA *Operations Manager* allows control structures to be introduced explicitly as elements in the process modeling flowsheet. This methodology enforces an intuitive distinction between the model behavior that is the result of physicochemical phenomena (exemplified by the conservation equations) and the behavior that stems from external intervention (due to manual or automatic manipulations of process parameters).

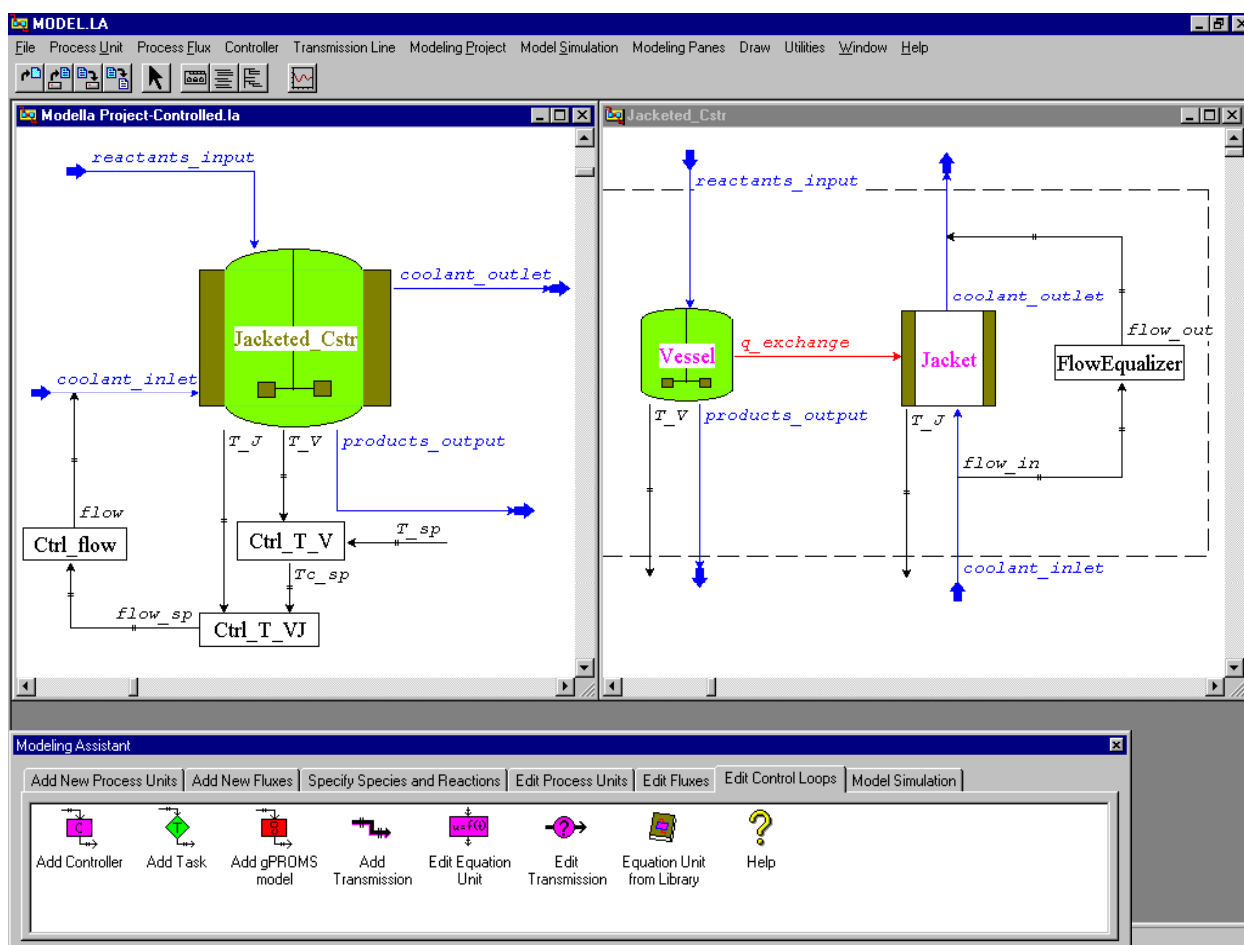


Figure 5-37: Declaration of Control Structures

Controllers are introduced by the modeler using the *Edit Control Loops* tab of the *Modeling Assistant* (illustrated at the bottom of Figure 5-37). Controllers may represent process controllers, valves, sensors, actuators, etc. Conservation equations are not derived for a

controller. Rather, each controller in the model relates a set of one or more input variables to a set of one or more output variables through a set of *control laws* (i.e., equations). *Transmission lines* establish links between the input and output variables of the controller and process variables associated with the phenomena-based model that are being measured or manipulated by the controller. The associated variable of a transmission line is selected using the *Transmission Variable Selection Dialog* illustrated in Figure 5-38.

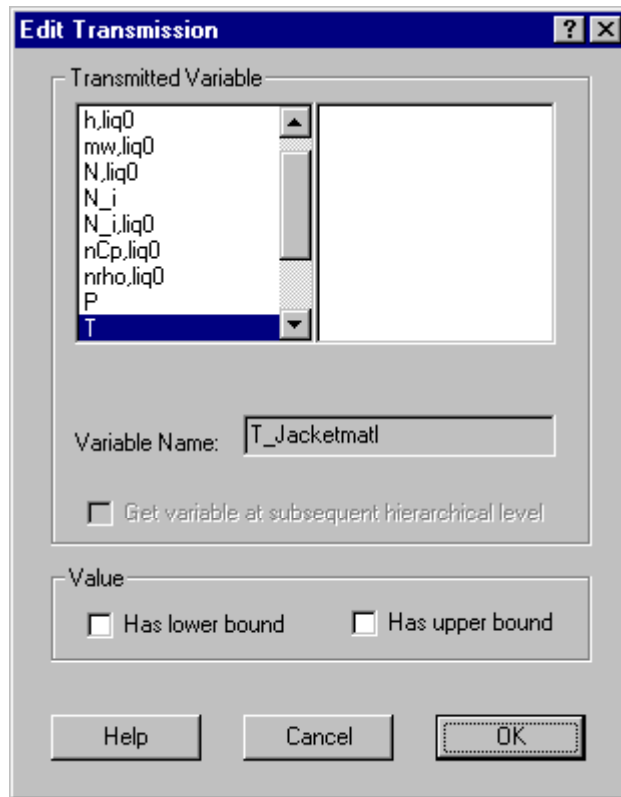


Figure 5-38: Transmission Variable Selection Dialog

In defining process control relationships, the modeler may select from a set of predefined control law templates (e.g., PID control) or may define an arbitrary set of equations to serve as the control laws. Control laws are specified using the *Control Law Specification Dialog*, illustrated in Figure 5-39. A restriction is imposed that each control structure is structurally self-consistent (i.e., given the set of controller inputs, the controller laws provide a well-defined set of mathematical equations that may be used to uniquely calculate the set of controller outputs. At solution time, the equations represented by the process controller are appended and solved simultaneously with the conservation equations generated by the *Model Generator*.

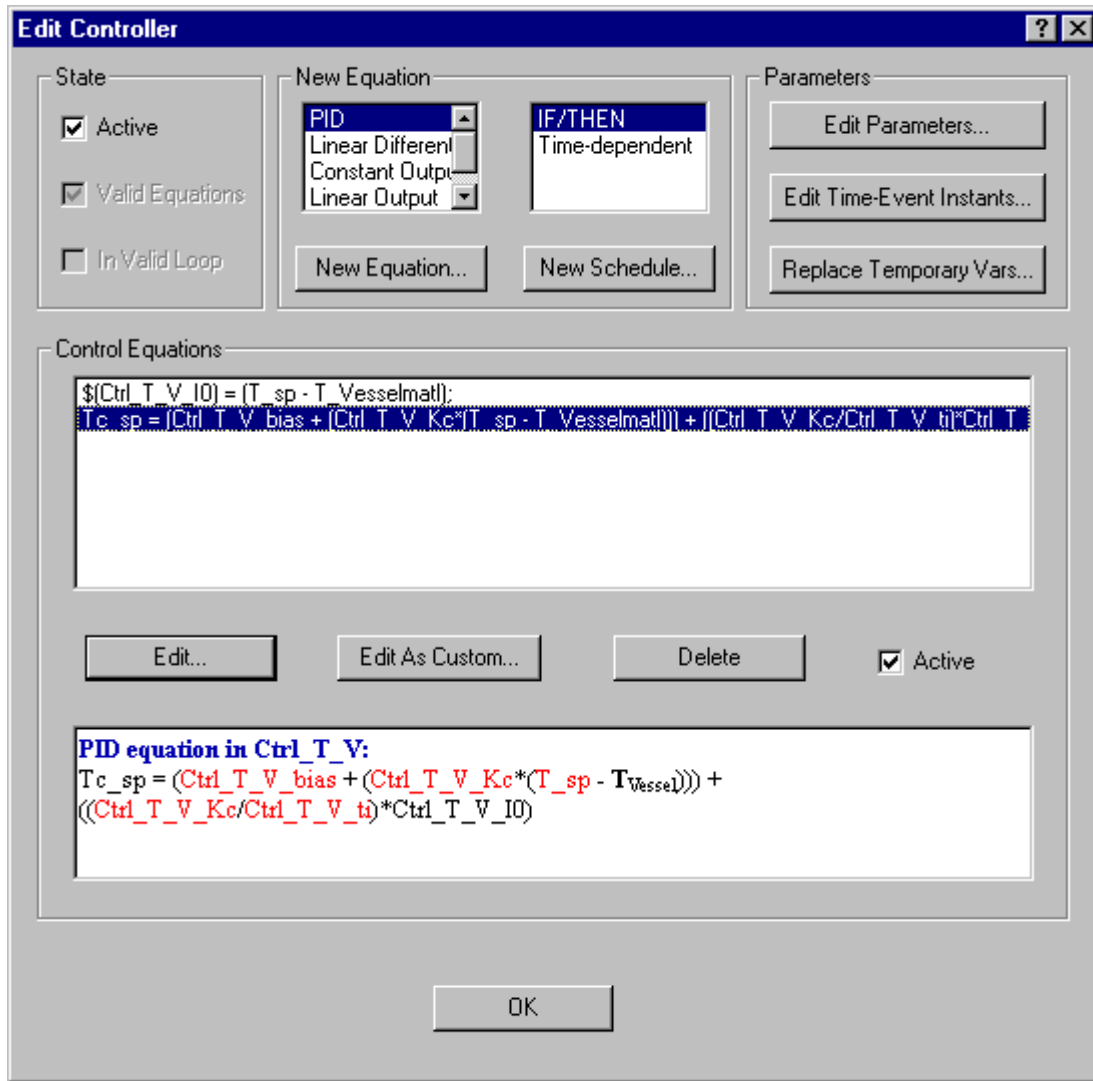


Figure 5-39: Control Law Specification Dialog

5.4.3 External Models

The concept of a structurally self-consistent process controller modeling element has been extended to also incorporate elements modeled outside of MODEL.LA. These external components, modeled using the gPROMS equation-based modeling language (Barton, 1992), may be incorporated as *external models* on the MODEL.LA process modeling flowsheet. Each such external model is associated with a gPROMS model definition file. Transmission lines are again used to establish links between the input and output variables of the external model and process variables associated with the phenomena-based model. The *gPROMS External Model Definition Dialog* (illustrated in Figure 5-40) is used to relate process variables to variables

appearing in the external model definition file. In a similar manner, the open-architecture features of the gPROMS language may be used to send, retrieve, and incorporate runtime data from external applications during model simulation.

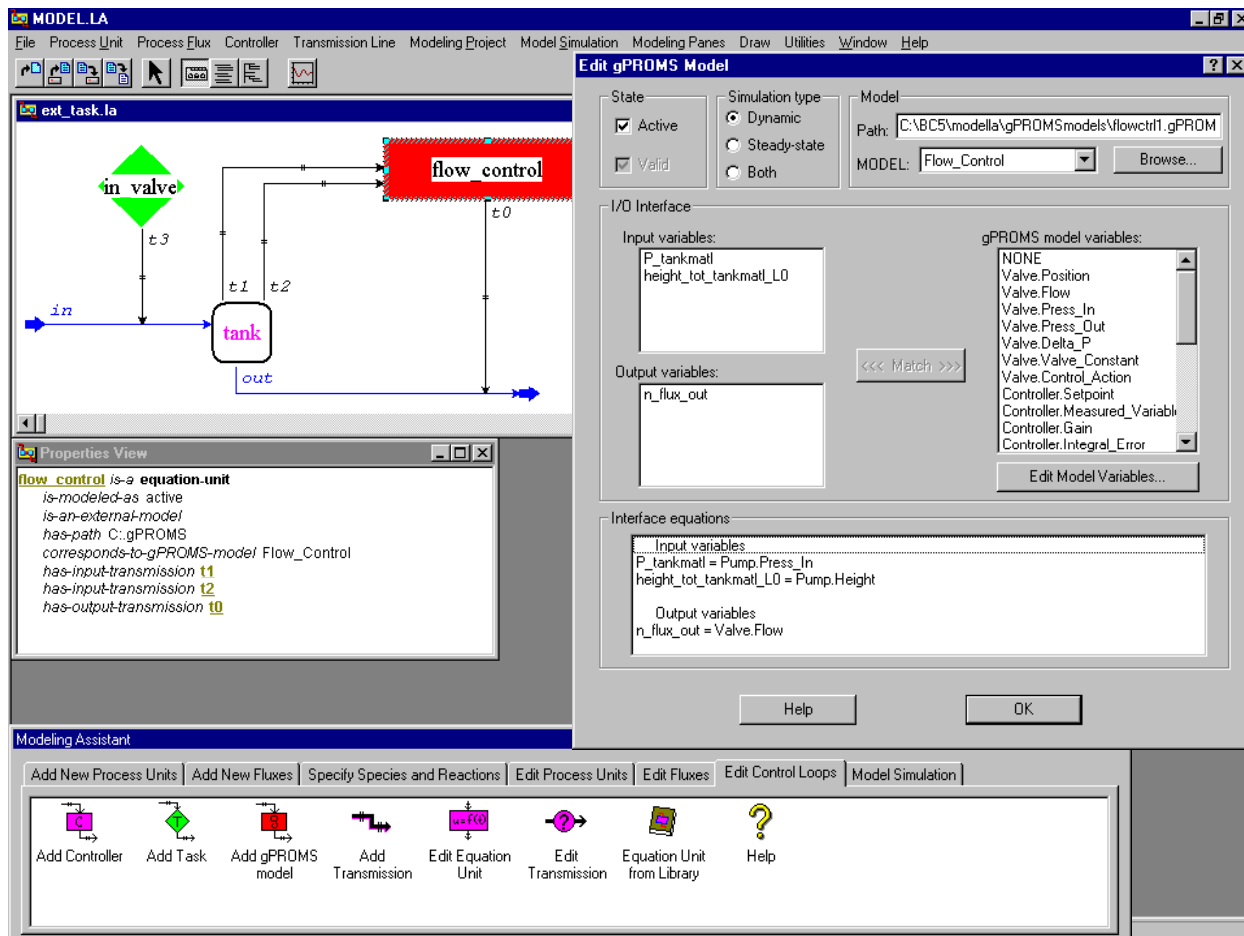


Figure 5-40: gPROMS External Model Definition Dialog

5.4.4 Operational Schedules

In the definition of a combined discrete/continuous model in gPROMS, *processes* are formed by the application of *tasks* to instances of *models*. A gPROMS model encompasses a set of continuous mathematical equations meant to describe the behavior of a modeled system. Tasks, which represent discrete procedures such as control actions or disturbances, may be imposed on the modeled system. Both models and tasks may be defined hierarchically through inheritance from other models or tasks, respectively. Tasks are used to compose *schedules*, which specify the sequential or concurrent (i.e., parallel) execution of selected tasks. Furthermore, alternative tasks

may be executed conditionally, based on run-time process conditions, or iteratively, where a task is executed repeatedly until some conditions are met.

In a similar manner, the definition of tasks and schedules may be incorporated into the definition of a phenomena-based model using the *Operations Manager* of MODEL.LA. The *task* element, declared on the MODEL.LA process model flowsheet, represents the discrete manipulation of a process variable (e.g., volumetric flow rate). This allows discrete events, such as opening a valve or charging a reactor, to be modeled. Such tasks are used to compose *schedules*, which are comprised of a set of actions and events executed sequentially or in parallel. Different types of actions and events allow the declaration of complex schedules involving sequential and parallel branches with actions triggered by a multitude of conditions. Separation of schedules from the phenomena-based model allows different schedules to be implemented for the same model.

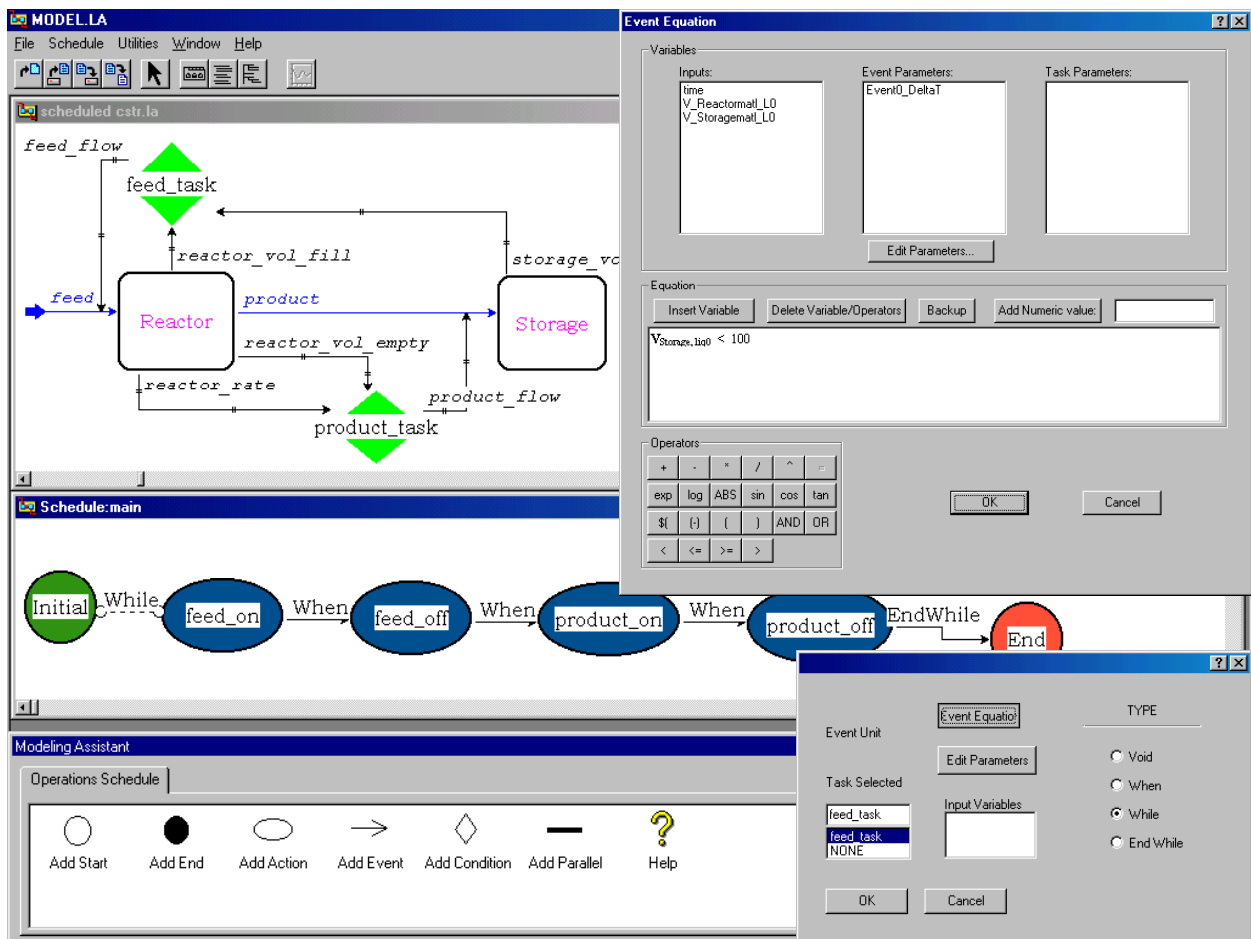


Figure 5-41: Specification of Operational Schedule

In MODEL.LA, there are two primary steps in declaration of a schedule. The tasks are first declared on the process model flowsheet and then these tasks are used to compose the schedule. Similar to a process controller, each task on the process model flowsheet is associated with a set of one or more measured variables, represented by transmission lines incident to the task icon, and a set of one or more manipulated variables, represented by transmission lines incident from the task icon.

Once the necessary tasks have been declared on the process model flowsheet, a schedule can be defined. In MODEL.LA, schedules are defined graphically using flowcharts (as illustrated in Figure 5-41). A schedule is composed of a structured sequence of tasks and/or other schedules (allowing a hierarchical description of composite schedules). Every task in a schedule is composed of an event followed by an action. When control passes to the event and its condition is satisfied, the action is triggered.

Events in a schedule flowchart are depicted as arrows that interconnect icons which represent actions. Events may be of type *void*, *when*, *while* or *end while*. When control is passed to a *void* event, the associated action is immediately triggered. When control is passed to a *when* event, the associated action is triggered once the condition defined for the event is satisfied. At the conclusion of an action associated with a *void* or *when* event, control is passed to the subsequent event. When control is passed to a *while* event, if the condition defined for the *while* event is not satisfied, control immediately passes to the subsequent *end while* event in the schedule and the action following the *end while* event is then triggered. If the condition defined for the *while* event is satisfied, the associated action is triggered and execution of the schedule continues from that point until an *end while* event is triggered. At that point, control returns to the previous *while* event. In this manner, the *while* event allows the definition of iterative task execution. A condition defined for a *when* or *while* event may be a function of a time event (e.g., $\text{time} < 1 \text{ hour}$), a state event (e.g., $\text{temperature} \geq 300 \text{ K}$), or both. In the case of a state event, the event must be associated with a task that measures the required input variable.

Each action in a schedule is associated with a task that results in the manipulation of some process variable when the action is triggered. Each schedule must consist of an initial event (representing the start of the schedule), one or more intermediate actions, and at least one end action (which represent termination of the schedule). Conditional branches in a schedule may also

be defined to specify alternative paths in a schedule. Here, the branch taken in the schedule is determined during the simulation by the validity of the conditional defined by the modeler. Parallel branches may also be defined to introduce the concurrent execution of two or more branches in a schedule. Finally, composite actions may be defined which embed an entire sub-schedule or operating procedure, allowing a hierarchical abstraction of complex schedules.

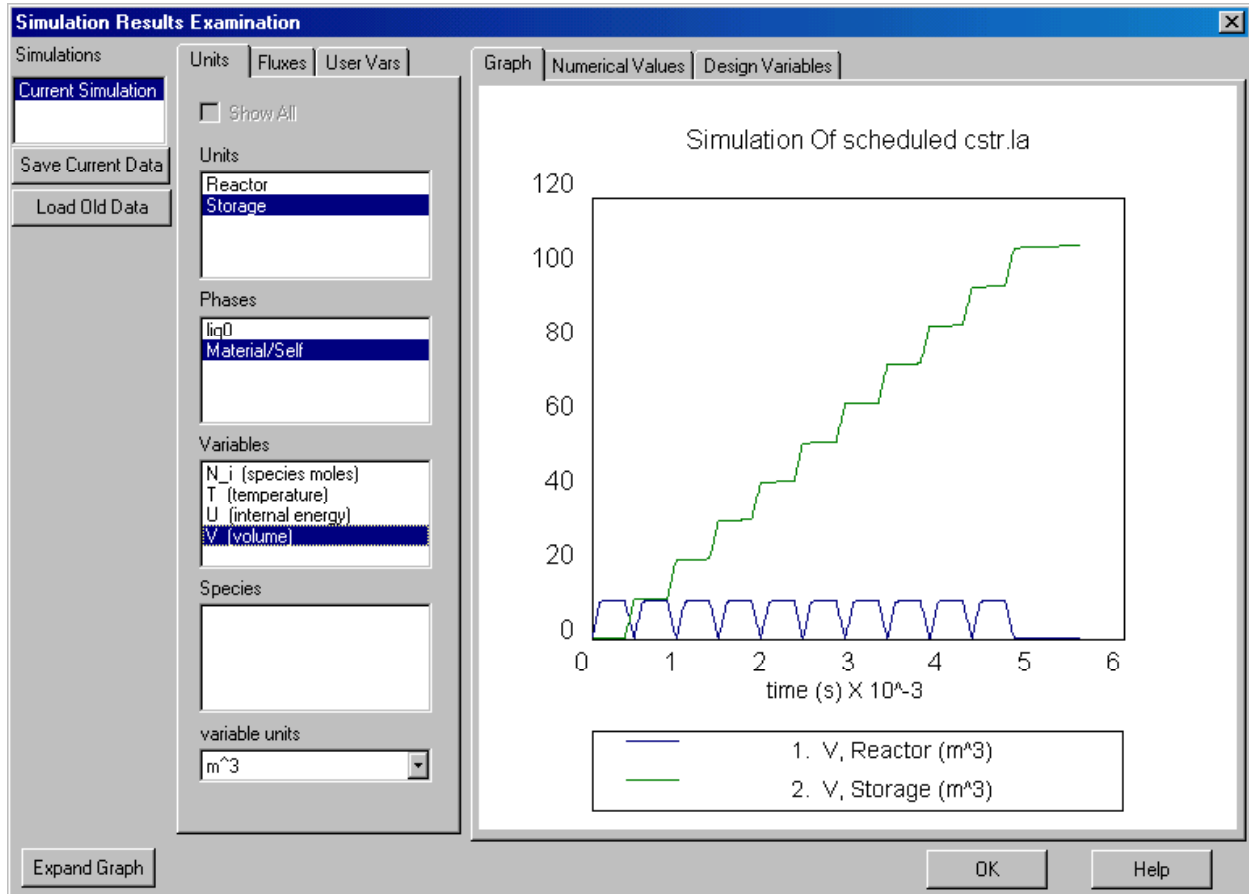


Figure 5-42: Discrete/Continuous Behavior of Scheduled Process Model

The introduction of *Operational Schedules* introduces a hybrid discrete and continuous behavior into a phenomena-based model simulation. Figure 5-42 illustrates the such behavior for the model shown in Figure 5-41. In this example, the *Reactor* is charged to a volume of 10 m^3 , then drained into a *Storage* vessel once the rate of reaction in the *Reactor* falls below a certain level. This is repeated (using a *while* task in the schedule) until the volume of material in the *Storage* vessel exceeds 100 m^3 .

5.5 Numerical Engine

Once the model equations are derived by the *Model Generator* and supplemented with additional correlations and relationships from the *Properties Manager* and *Operations Manager*, the complete mathematical model is passed to the *Numerical Engine* of MODEL.LA. Here, the modeler is guided through specification of design variables, initial guesses, an index analysis and initial condition for dynamic models, solution of the equations, and display of results. The entry point into the *Numerical Engine* for the modeler is the *Numerical Engine Toolbar* (Figure 5-43).

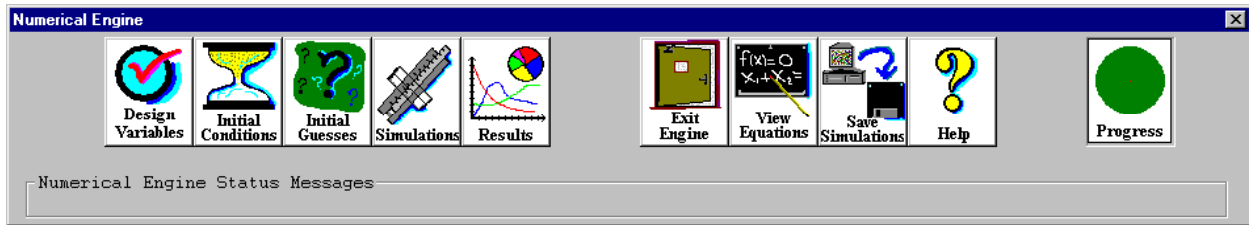


Figure 5-43: Numerical Engine Toolbar

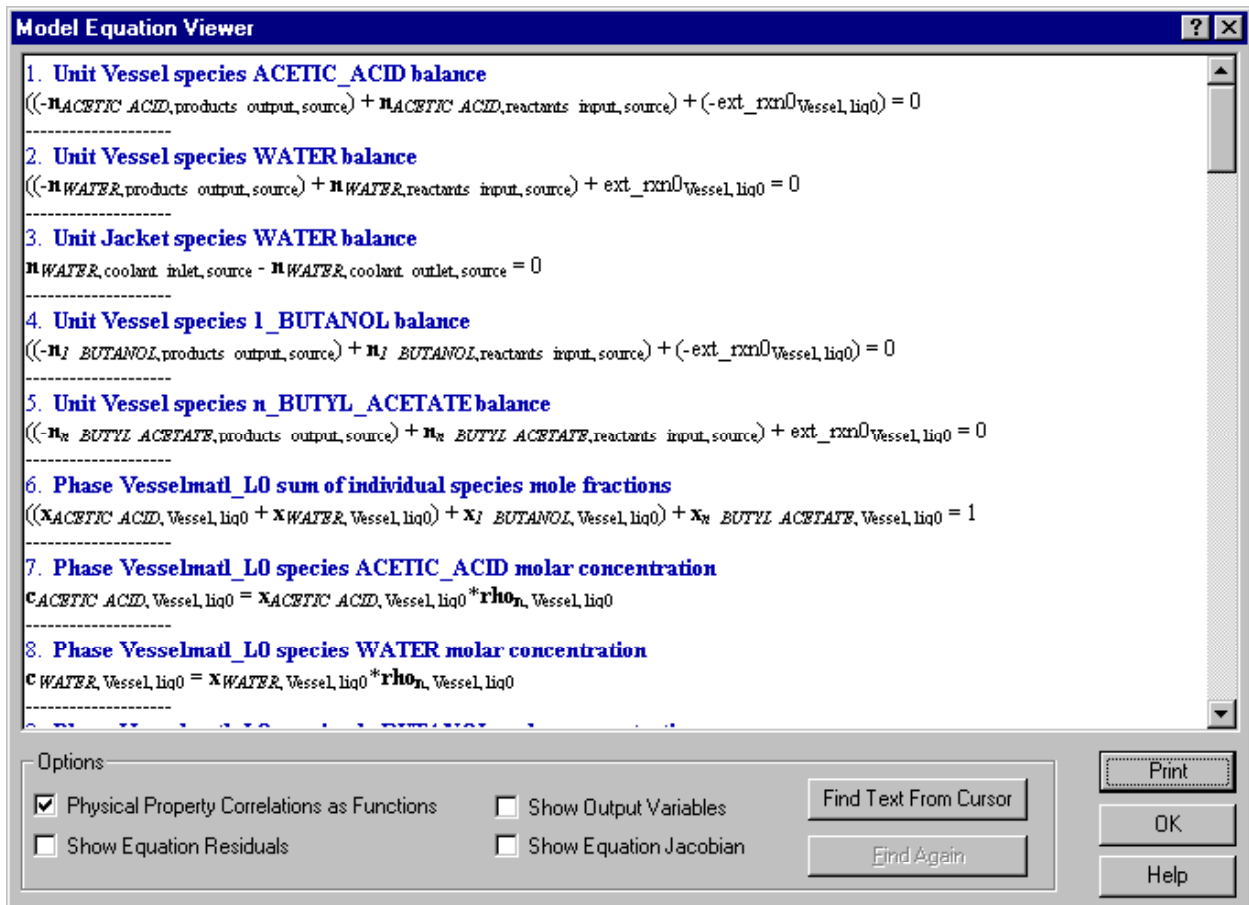


Figure 5-44: Model Equations Dialog

5.5.1 Display of Model Equations

The modeler may view the complete set of model equations at any time by selecting the option *View Equations* on the *Numerical Engine Toolbar*. This activates the *Model Equations Dialog* (Figure 5-44). Equations, terms, and variables, are all related to the modeling elements that produced them through the use of subscripts and headings. Also, the basis of each equation (e.g., *Acetic Acid balance*) is displayed.

5.5.2 Design Variable Specification

The first task in the specification of the mathematical model is the selection of the design (or known) variables. The modeler initiates this action by selecting the *Design Variables* option on the *Numerical Engine Toolbar*. This activates the *Design Variable Specification Dialog* (Figure 5-45).

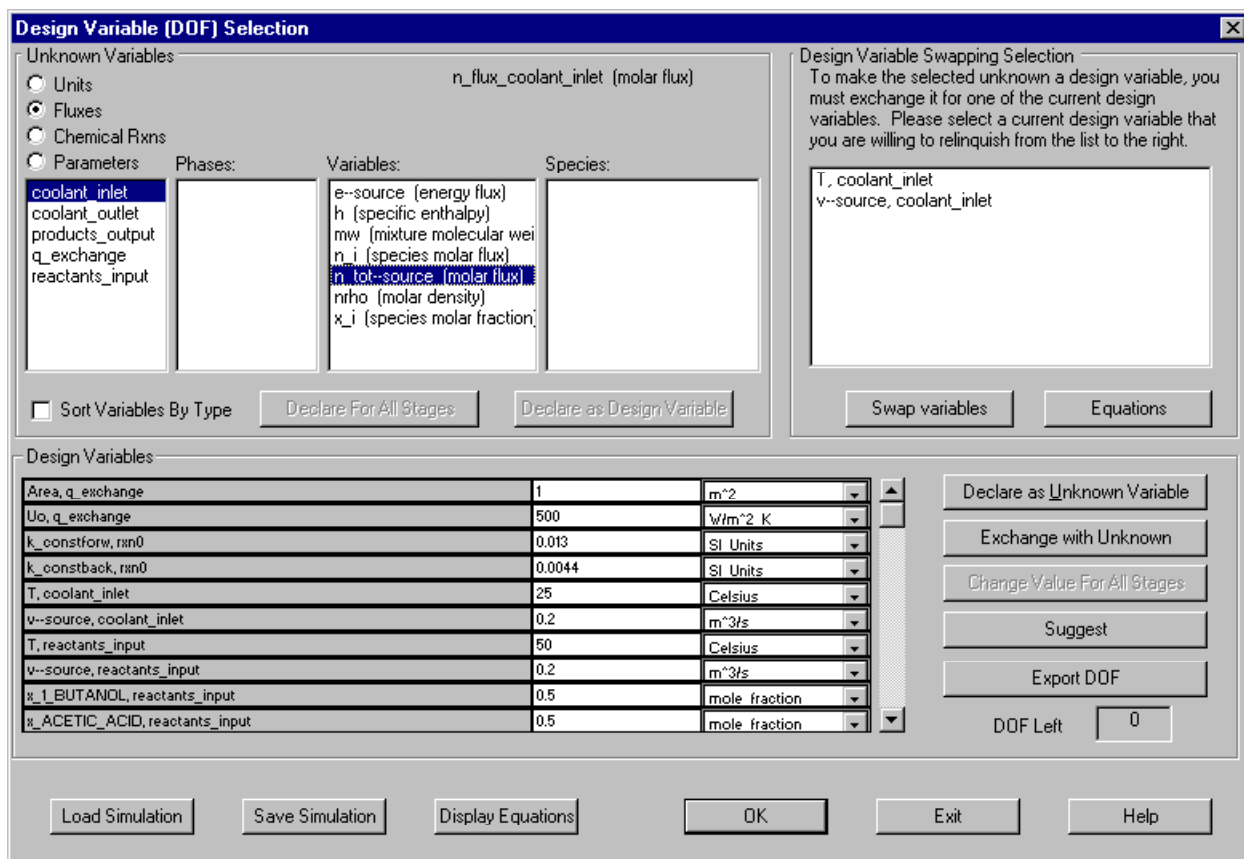


Figure 5-45: Design Variable Specification Dialog

The number of variables that must be specified (also referred to as the number of degrees of freedom) is equal to the number of variables minus the number of equations in the mathematical

model. The *Design Variable Specification Dialog* displays all variables in the mathematical model, grouped according to modeling element (e.g., *Vessel*) or variable type (e.g., *temperature*). As the modeler selects each desired design variable, the structural consistency of the selection is immediately verified using an incidence matrix (Steward, 1962). If the selection conflicts with any preselected design variables, a list of the conflicting variables is presented for exchange with the most recent selection by generation of all feasible Steward paths (Steward, 1965). At any point in the selection the modeler may request that a structurally consistent set of design variables be proposed to fill the remaining degrees of freedom, which are determined using the path augmentation algorithm described by Duff (1981a, 1981b). The modeler may review these suggested variables and for each that is not a desirable design variable, the alternative variables to replace it are presented for exchange.

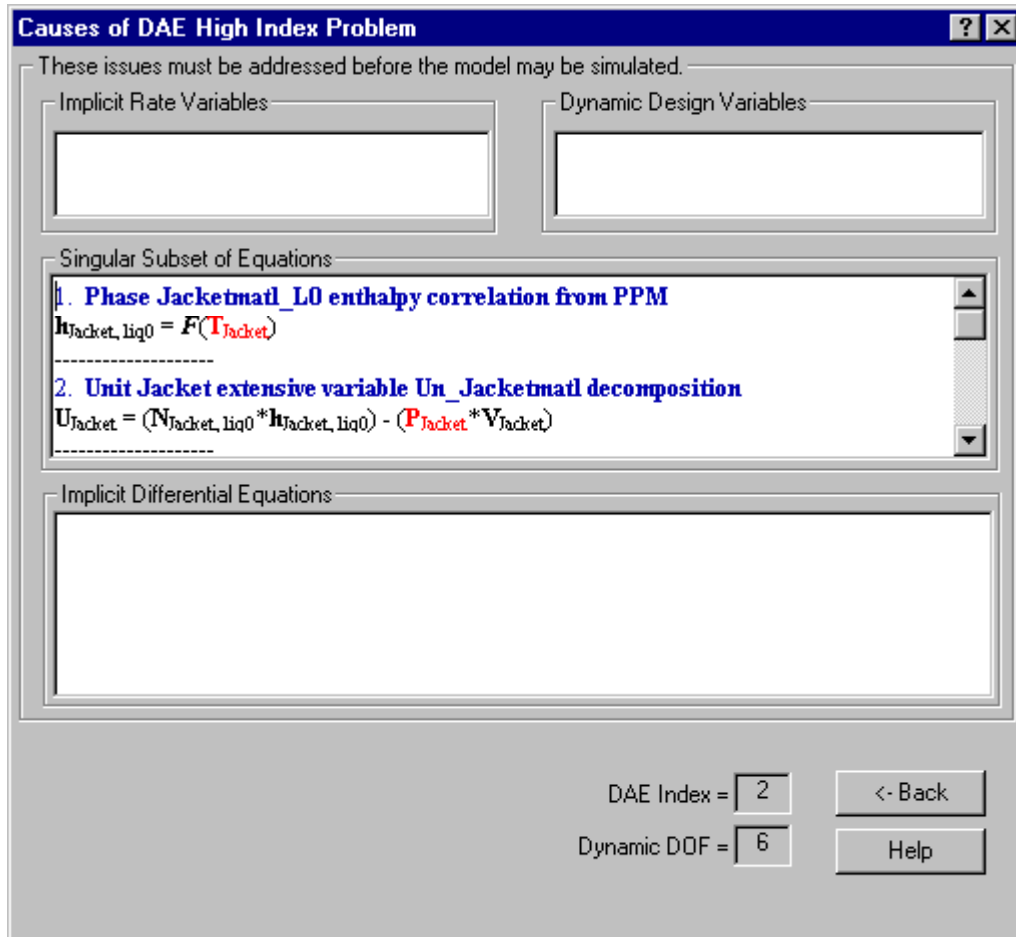


Figure 5-46: High Index Diagnosis Dialog

5.5.3 Index Analysis

For dynamic models, a structural index analysis (Pantelides, 1988) is performed immediately after the complete specification of the degrees of freedom design variables. If a structurally high index mathematical model is detected, the singular subset of equations or other cause of the index problem is presented in the *High Index Diagnosis Dialog* (illustrated in Figure 5-46). In such a case, the numerical solution is not allowed to proceed until the problem is reformulated. Techniques for the reduction of high index models have been discussed by Moe (1995).

5.5.4 Initial Conditions

For dynamic models, a set of initial conditions must also be specified. The modeler initiates this action by selecting the *Initial Conditions* option on the *Numerical Engine Toolbar*. This activates the *Initial Conditions Specification Dialog* (Figure 5-47).

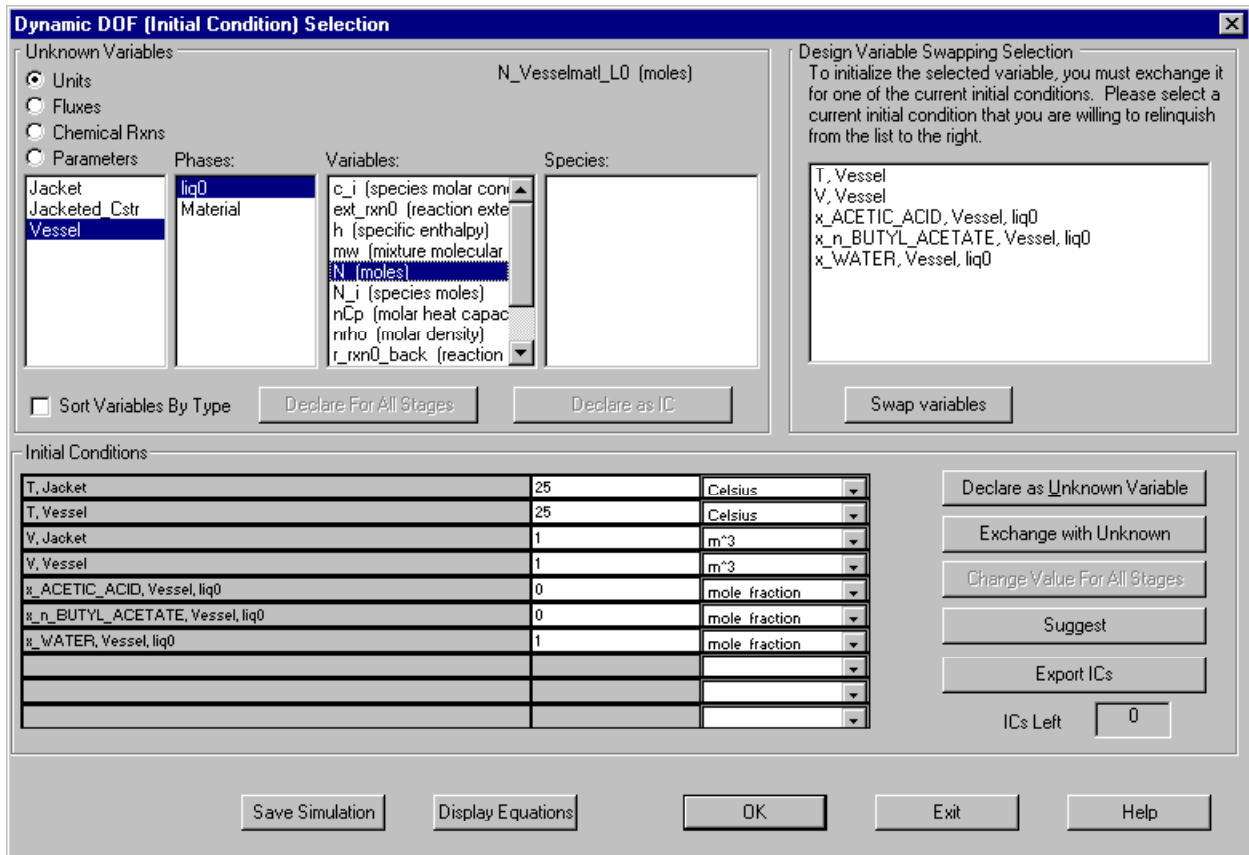


Figure 5-47: Initial Condition Specification Dialog

Since initial conditions are specified in a manner analogous to the specification of degrees of freedom, the functionality of this dialog is essentially the same as the *Design Variable*

Specification Dialog.

5.5.5 Initial Guess Specification

Once the proper number of design variables are specified to satisfy the degrees of freedom, the modeler may specify initial guesses for any or all unknown variables. The modeler initiates this action by selecting the *Initial Guesses* option on the *Numerical Engine Toolbar*. This activates the *Initial Guesses Variable Specification Dialog* (Figure 5-48).

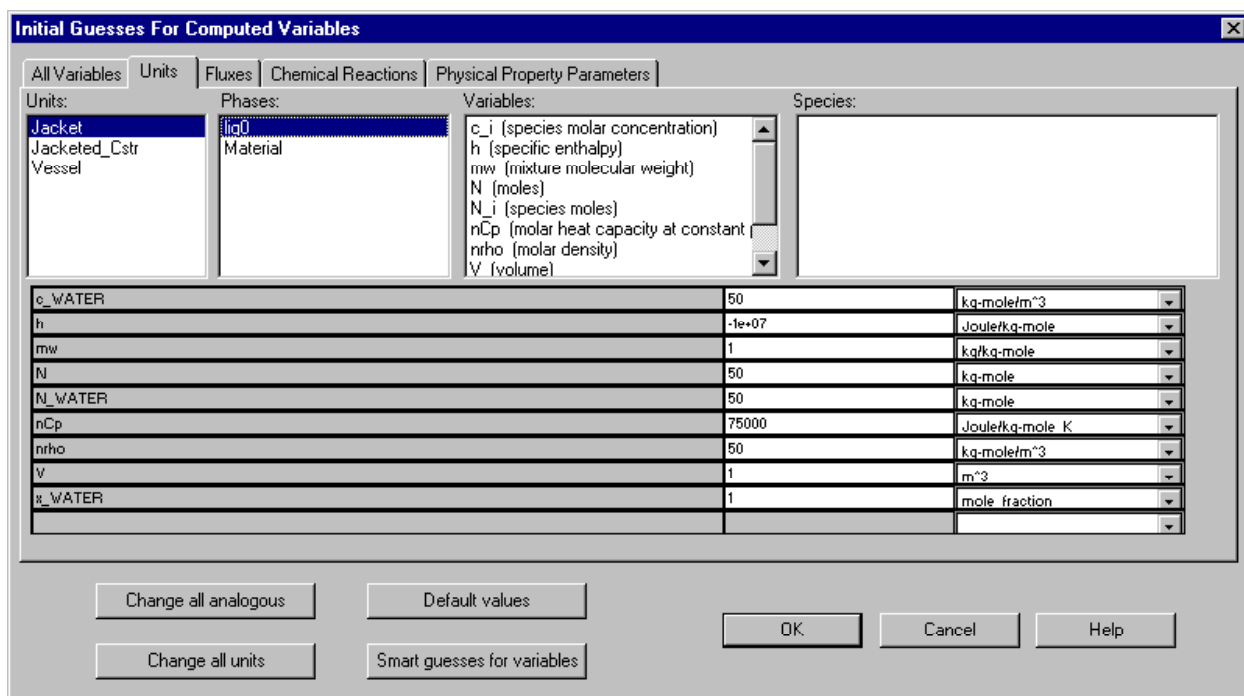


Figure 5-48: Initial Guesses Specification Dialog

Rough order-of-magnitude estimates for all variables are provided by default based on their type (e.g., temperature variables are set to 298 K). The modeler may specify a different estimate for any variable or set of variables of a particular type. The *Numerical Engine* will also automatically calculate initial guesses for all unknown thermodynamic and physical properties based on guessed values of temperature, pressure and composition.

5.5.6 Solution of Model Equations

Once the mathematical model is structurally well-posed through specification of design variables and initial conditions (if necessary), the model is prepared for numerical solution. The modeler

initiates this action by selecting the *Simulation* option on the *Numerical Engine Toolbar*. This activates the *Numerical Solver Specification Dialog* (Figure 5-49).

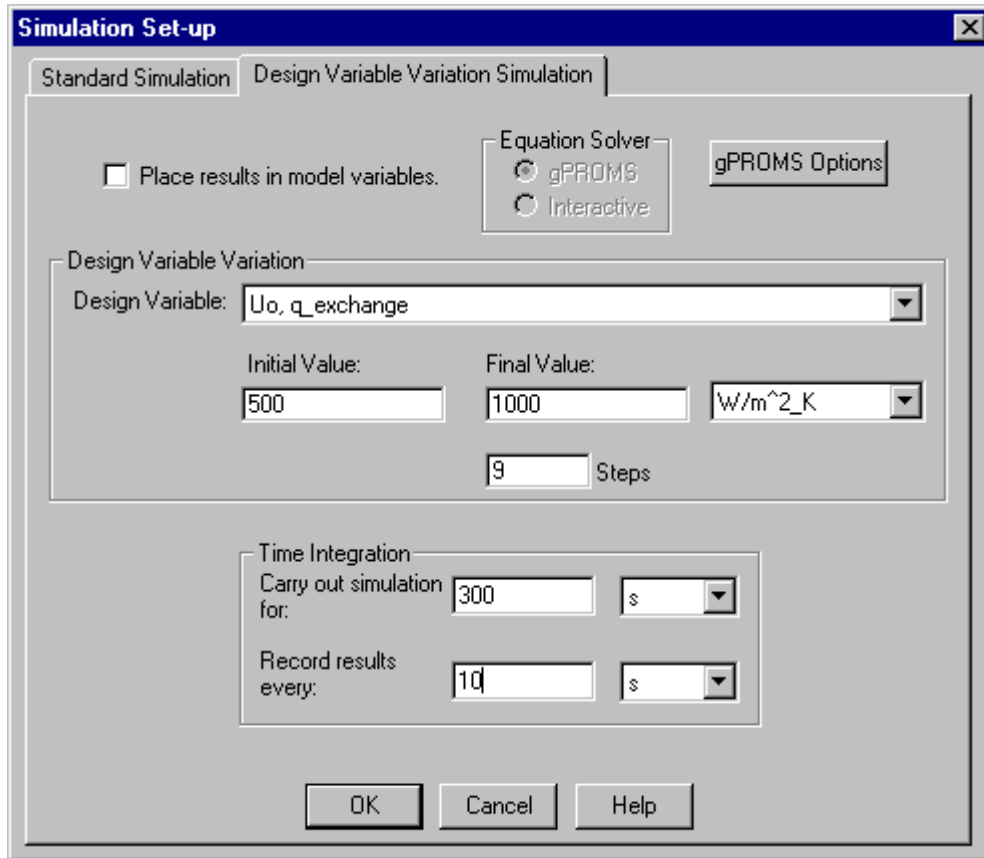


Figure 5-49: Numerical Solver Specification Dialog

For steady-state models, the model is ready for solution. For dynamic models, the modeler must additionally specify the length of simulation time and the time interval at which results are to be recorded. The *Numerical Engine* then automatically generates a gPROMS input file, launches the solver for solution of the model, and reads the simulation results back into the MODEL.LA process variables (thus retaining robust initial guesses for subsequent simulations). Alternatively, for steady-state (algebraic) models, the modeler may select to use the built-in interactive solver of the *Numerical Engine* (illustrated in Figure 5-50). This solver presents the modeler with a block-by-block decomposition of the model equations, and uses a Newton-Raphson method to solve each block of equations individually. If a block of equations does not converge, the user may view the value of the unknown variables at the last iteration, update their guesses, and restart the solution procedure.

For steady-state or dynamic models, the modeler may also choose a *Design Variable Variation Simulation*. Here, a design variable of interest is selected for variation over a specified range. The mathematical model is then solved repeatedly where for each solution the variable of interest is incremented through the specified range over a designated number of intervals.

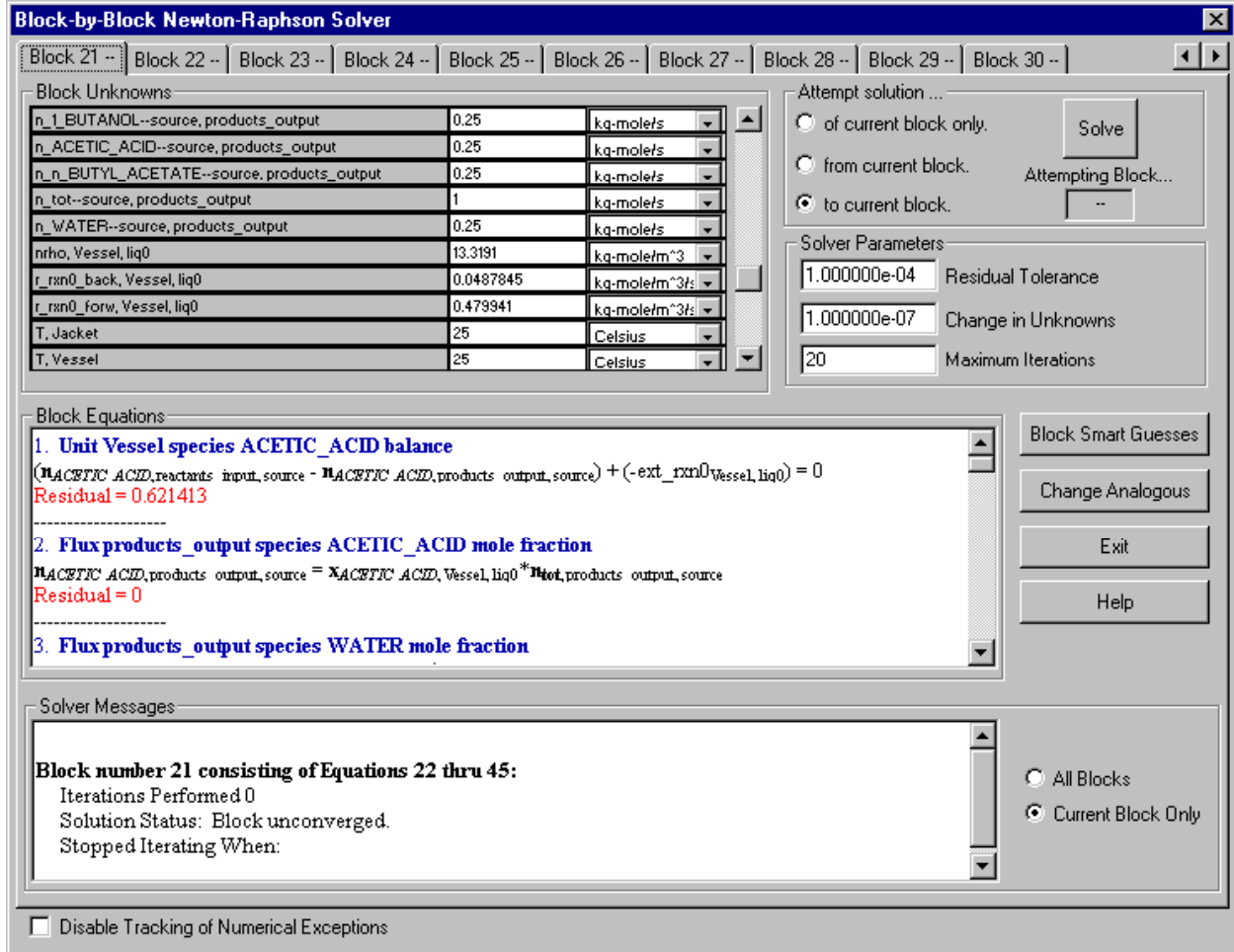


Figure 5-50: MODEL.LA Block Solver Dialog

5.5.7 DAE Systems Numerical Solution Methods

The original implementation of gPROMS (Barton, 1992) utilized the numerical package DASOVL (Jarvis, 1993) for the solution of DAE systems. A DAE system may be expressed as:

$$f(\dot{x}, x, y, t) = 0$$

where x is the n -dimensional vector of unknown variables, y is m -dimensional vector of known variables, t is the independent variable, $\dot{x} \equiv \frac{dx}{dt}$, and f is the set of equations represented as an $n+m$

dimensional vector-valued function. Many DAE solvers such as DASOLV and DASSL (Petzold, 1982) are based on a method first proposed by Gear (1971). This method substitutes a difference approximation for \dot{x} which is based on the backwards differentiation formula family of methods. By fitting a k^{th} order polynomial to $k+1$ values of x , the time derivatives at a given integration step are approximated as:

$$\dot{x}_n \approx \frac{1}{h_n} \sum_{i=0}^k \alpha_n x_{n-i}$$

where subscripts such as n refer variable values at the n^{th} integration step, α is a scalar multiplier, and h is the current step size. This expression may be rearranged to give:

$$\dot{x}_n \approx \frac{\alpha_0 x_n}{h_n} + \frac{1}{h_n} \sum_{i=1}^k \alpha_n x_{n-i} = \frac{\alpha_0 x_n}{h_n} + \gamma_n$$

where γ_n at each iteration is a constant computed from terms from previous iterations. Substituting this expression into the DAE system for each integration step yields :

$$f\left(\frac{\alpha_0}{h_n} x_n + \gamma_n, x_n, y_n, t_n\right) = 0$$

Thus, the resulting set of *algebraic* equations can be solved at each integration step using a Newton-based method. DAE solvers such as DASOLV and DASSL also automatically adjust the integration step size and order of integration of BDF methods so that error estimates satisfy a *user-specified* tolerance.

The BDF solution methods described above are limited to the solution DAE systems whose index does not exceed unity. Furthermore, these methods are appropriate for the solution of continuous DAE systems. Introduction of discrete changes into the mathematical model requires reinitialization of the mathematical model (i.e., a new initial value problem must be solved) at the point of discontinuity. This of course requires the localization of these discrete events by the solver that implements DAE solution routines. The introduction of time events, whose exact time of occurrence is known in advance, simply requires integration to the specified time points. However, the introduction of state events, whose exact time of occurrence is not known in advance but rather determined by state conditions that come true during the integration, require identification and localization during the solution procedure. The implementation of routines for the identification and localization of state events in gPROMS is discussed in Barton

(1992). An efficient algorithm that “guarantees the location of all state event in strict time order” for initial value problems in DAE systems with discontinuities has described by Park and Barton (1996).

5.5.8 IPDAE Systems Numerical Solution Methods

The extension of gPROMS to encompass the numerical solution of IPDAE systems (Oh, 1995) involved the integration of solution methods classified as belonging to the family of *methods of lines*. In these methods, a two-step approach is used. First the distributed spatial dimensions are discretized into finite dimensional representations, yielding a DAE approximation of the IPDAE system. The DAEs are then integrated over the desired solution time using appropriate numerical techniques. The discretized results from solution of the DAEs are interpreted as approximations to the continuous behavior of the IPDAE system.

The solution methods for IPDAE systems implemented in gPROMS and accessible through MODEL.LA for the modeling of spatially distributed systems are listed in Table 5-5.

Table 5-5: Summary of gPROMS IPDAE Solution Methods

<i>Numerical Method</i>	<i>Orders of Approximation</i>
Centered Finite Difference Method (CFDM)	2, 4, 6
Backward Finite Difference Method (BFDM)	1, 2
Forward Finite Difference Method (FFDM)	1, 2
Upwind-Biased Finite Difference Method (UFDM)	2
Orthogonal Collocation on Finite Elements Method (OCFEM)	2, 3, 4

Common features of these discretization methods (Oh, 1995) include:

- the spatial variation of each distributed variable $\phi(z)$, $z \in [Z^L, Z^U]$ is approximated in terms of the values of variable as $\phi(z_i)$ at a finite and *fixed* set of positions $z_i \in [Z^L, Z^U]$,
- equations that are distributed over the domain $[Z^L, Z^U]$ are enforced at some of the points $\{z_i\}$ while other desirable properties of the solution (e.g., continuity) are enforced at others, and
- the partial derivatives of $\phi(z)$ at the points $\{z_i\}$ and integrals over the domain

$[Z^L, Z^U]$ are approximated in terms of the values $\phi(z_i)$.

In gPROMS, finite difference methods are based on polynomial approximations of the distributed variables about the grid points $\{z_i\}$ which are *uniformly spaced* at a distance h apart. A n^{th} order polynomial approximation can be constructed in terms of the values of the variable at $n+1$ consecutive points on the grid (i.e., z_i, \dots, z_{i+n}). This approximation may be expressed as:

$$\phi(z) \approx \sum_{j=0}^n \phi(z_{i+j}) L_j^{[n]}(z)$$

where $L_j^{[n]}(z)$ is a n^{th} degree Lagrange polynomial defined as

$$L_j^{[n]}(z) \equiv \prod_{k=0, k \neq j}^n \frac{z - z_{i+k}}{z_{i+j} - z_{i+k}}$$

The first spatial derivative of $\phi(z)$ at each grid point z_{i+q} , $q=0..n$, may then be approximated as

$$\frac{d\phi}{dz}(z_{i+q}) \approx \sum_{j=0}^n \phi(z_{i+j}) \frac{dL_j^{[n]}}{dz}(z_{i+q}), \quad q = 0..n$$

For the first order forward and backward finite difference methods, respectively, the resulting expressions are:

$$\begin{aligned} \frac{d\phi}{dz}(z_i) &\approx \frac{-\phi(z_i) + \phi(z_{i+1})}{h} \\ \frac{d\phi}{dz}(z_i) &\approx \frac{-\phi(z_{i-1}) + \phi(z_i)}{h} \end{aligned}$$

Likewise, for second order backward, centered, and forward finite difference approximations, respectively, the resulting expressions are:

$$\begin{aligned} \frac{d\phi}{dz}(z_i) &\approx \frac{-3\phi(z_i) + 4\phi(z_{i+1}) - \phi(z_{i+2})}{h} \\ \frac{d\phi}{dz}(z_i) &\approx \frac{-\phi(z_{i-1}) + \phi(z_{i+1})}{2h} \\ \frac{d\phi}{dz}(z_i) &\approx \frac{\phi(z_{i-2}) - 4\phi(z_{i-1}) + 3\phi(z_i)}{2h} \end{aligned}$$

Finally, an example of a biased upwind approximation is given as

$$\frac{d\phi}{dz}(z_i) \approx \frac{-6\phi(z_{i-1}) - 20\phi(z_i) + 36\phi(z_{i+1}) - 12\phi(z_{i+2}) + 2\phi(z_{i+3})}{24h}$$

An orthogonal collocation method approximates the solution of IPDAE systems by weighted combinations of orthogonal polynomials of degree n , and demands that the describing

equations be satisfied exactly at a finite set of points, called collocation points. In gPROMS, this method has been implemented in conjunction with a finite element approach, where the domain is divided into elements and an orthogonal collocation method is applied in each element. This solution method is termed *orthogonal collocation method on finite elements*. The function $\phi(z)$ in element I is approximated as

$$\phi(z^I) \approx \sum_{j=0}^n \phi(z_j^I) L_j^{[n]}(z^I) \quad I = 1..m$$

where the position of the j^{th} point in element I is denoted by z_j^I . Thus, the first-order derivative of the approximated solution $\phi(z)$ at position q in element I becomes:

$$\frac{d\phi(z_q^I)}{dz} \approx \frac{1}{E^I} \sum_{j=0}^n A_{jq}^{[n]}(z) \phi(z_j^I) \quad I = 1..m, q = 0..n$$

where E^I is the length of element I and A is a constant matrix defined by:

$$A_{jq}^{[n]} \equiv \frac{dL_j^{[n]}}{d\hat{z}}(\hat{z}_q) \quad j, q = 0..n$$

where $\hat{z} \equiv (z - z_i) / E^I$. As an example, second-order approximations of first-order spatial derivatives in element I are given by the expressions

$$\begin{aligned} \frac{d\phi}{dz}(z_0^I) &\approx \frac{-3\phi(z_0^I) - \phi(z_1^I) + \phi(z_2^I)}{E^I} \\ \frac{d\phi}{dz}(z_1^I) &\approx \frac{4\phi(z_0^I) - 4\phi(z_2^I)}{E^I} \\ \frac{d\phi}{dz}(z_2^I) &\approx \frac{-\phi(z_0^I) + \phi(z_1^I) + 3\phi(z_2^I)}{E^I} \end{aligned}$$

with normalized collocation points at 0, 0.5, and 1.

In gPROMS, selection of an appropriate solution method, order of approximation, and discretization for the solution of IPDAE systems is solely the responsibility of the modeler. Obviously, the development of robust and reliable generic solvers for these systems remains an important continuing area of research.

5.5.9 Display of Numerical Results

After a successful (i.e., converged) simulation, the Numerical Engine displays the numerical results in tabular and graphical form, with variables organized by modeling element (as illustrated

in Figure 5-51). The results may also be exported in spreadsheet format. For distributed systems, OLE automation is used to automatically create surface plots (which are animated for dynamic simulations) in Microsoft Excel.

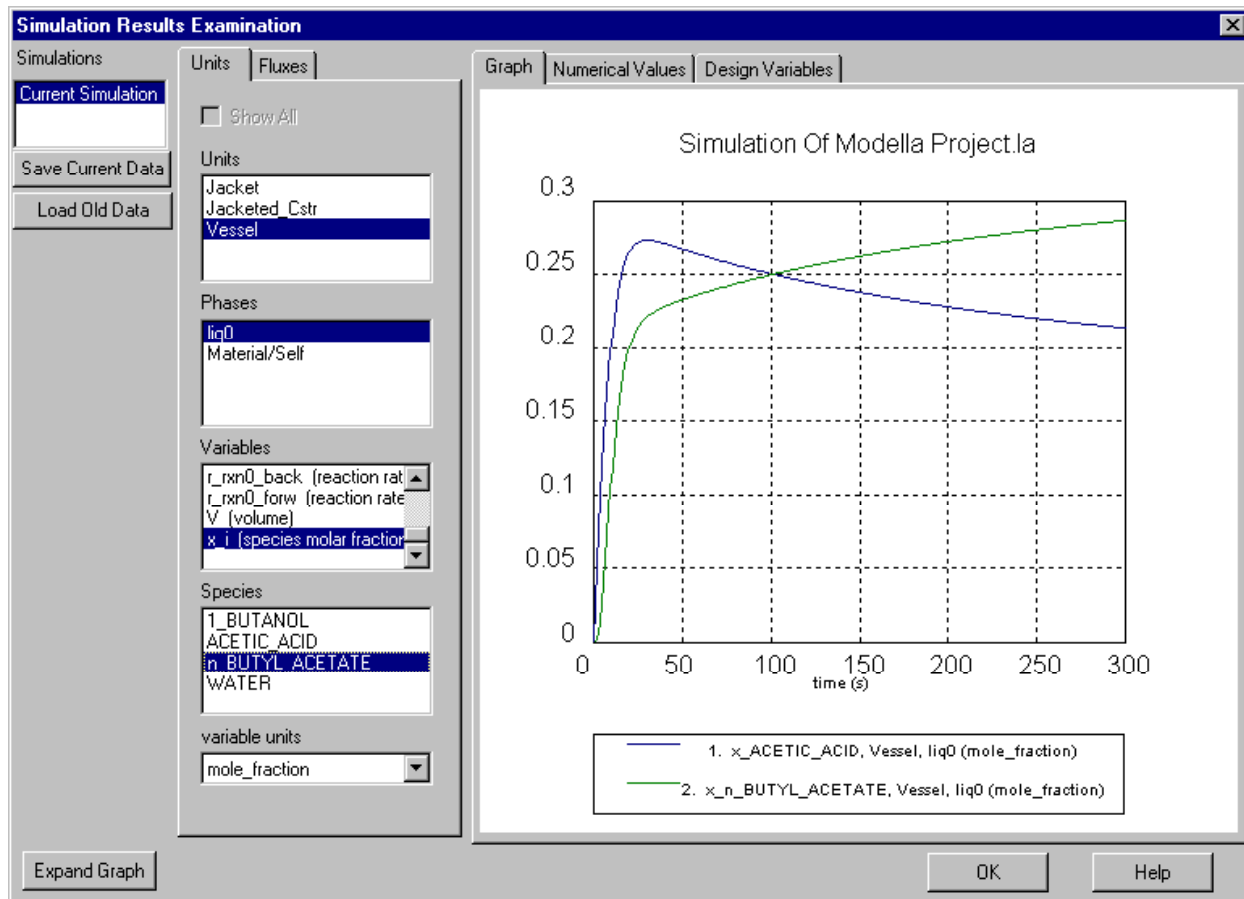


Figure 5-51: Numerical Results Display Dialog

5.6 Summary of MODEL.LA Modeling Environment

The MODEL.LA Modeling Environment provides an experimental framework for testing the concepts of phenomena-based modeling language and logic described earlier in this thesis. It also integrates state-of-the-art computer-aided modeling features, including an interactive graphical interface, incorporation of thermodynamic and physical property database information, description of process control and operational schedules, assistance for consistent specification of degrees of freedom and initial conditions for mathematical models, solution of the model equations using an equation-based modeling tool, and graphical display of results. In this manner, this high-level

modeling tool extends modeling assistance to all aspects of the process modeling activity, from declaration of the phenomena-based assumptions, generation of mathematical model equations, specification of the mathematical model, numerical solution, and display of results. At any point during these activities, the modeler is free to revisit the modeling assumptions, make any desired additions or modifications, and get immediate feedback on the impact of these assumptions on the resulting mathematical model and observed process behavior. In order to provide further details on the implementation of MODEL.LA, the following chapter describes the underlying software design of the modeling environment.

Chapter 6

Software Design of the MODEL.LA Modeling Environment

The previous chapter provided an overview of the functionality, graphical user interface, and overall structure of the phenomena-based MODEL.LA Modeling Environment. This description concentrated on the use of the environment from a modeler’s perspective. In this chapter, details regarding the software design of the MODEL.LA environment are presented. This description offers insight into and provides documentation for the construction of the underlying software system.

6.1 The Object Modeling Technique

In order to present the software design of the MODEL.LA Modeling Environment, a widely-used graphical notation known as the *Object Modeling Technique* (Rumbaugh et al, 1991), or OMT, is utilized. OMT is a methodology that captures multiple views of a system. The two primary views of the OMT methodology are the *object model* and the *functional model*. The object model represents the static, structural, “data” aspects of a systems. It describes the structure of objects in a system—their identity, their relationships to other objects, their attributes, and their operations. Objects are the units into which aspects of the real-world environment are divided. The *functional model* represents the transformational, “function” aspects of a system. The functional model specifies the meaning of the operations in the object model and the actions in the dynamic model. It specifies the results of a computation without specifying how or when they are computed. If desired, control information may be embedded into the functional model, or may be

represented as a separate *dynamic model*.

OMT utilizes a graphical notation to describe these software models. Depiction of the object model, which uses rectangles to represent classes of objects and lines to represent their interrelations, is similar to a semantic network. The functional model is depicted using data flow diagrams. Information included in these graphical models is usually presented selectively, abstracting certain details in order to highlight particular aspects of the system.

OMT is designed for the modeling and design of object-oriented systems. As a result, there is a natural mapping into an object-oriented programming language representation. In this work, the MODEL.LA Modeling Environment has been implemented in C++, an object-oriented programming language derived from the procedural language C. However, OMT is a generic representation that does not depend on the programming language used for implementation. In fact, a system such as MODEL.LA designed using the object modeling technique may be subsequently implemented in any object-oriented, procedural, or database programming language.

6.2 MODEL.LA Modeling Element Object Models

The object model depicts *classes* of objects in a system as rectangles. Each class is characterized by a name and, optionally, attributes and operations. In this work, object classes and their attributes are depicted using the graphical OMT notation illustrated in Figure 6-1.

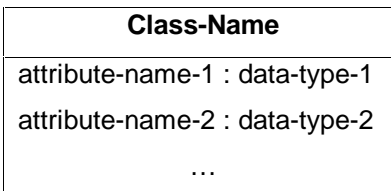


Figure 6-1: Object Modeling Notation for Classes

Interrelations between these classes are represented using lines (or *links*), which are referred to as *associations*. Associations are bi-directional, and may be labeled to characterize their purpose. Associations that are optional are indicated by a hollow circle at the end of the link. Associations that may occur an arbitrary number of times are indicated by a filled circle at the end of the link. Other constraints on the number of associations between two classes are indicated by numbers that label the end of the link (e.g., 2+ indicates a link that occurs 2 or more times). A class hierarchy is established by structuring object classes as a tree, where a superclass is connected by a line to the apex of a triangle and its subclasses are connected by lines to a horizontal bar

attached to the base of the triangle. Subclasses inherit all attributes and associations from their superclass. Object classes that are aggregates of a set of other object classes are also depicted using a tree structure, where a link originating from a hollow diamond attached to the base of the aggregate class branches to a set of one or more constituent classes.

The modeling elements introduced in Chapter 3 are represented as classes that provide the basis for the design of the MODEL.LA Modeling Environment. These classes are introduced in Figure 6-2 as subclasses of a *Modeling Element* class. This class has an attribute *name*, that uniquely identifies an instance of the class in a phenomena-based model, an attribute *comments*, that records any textual information specified by the modeler regarding an instance of the modeling element, and an association to multiple instances of object class *Variable*, which represent mathematical variables (e.g., temperature) associated with an instance of the modeling element.

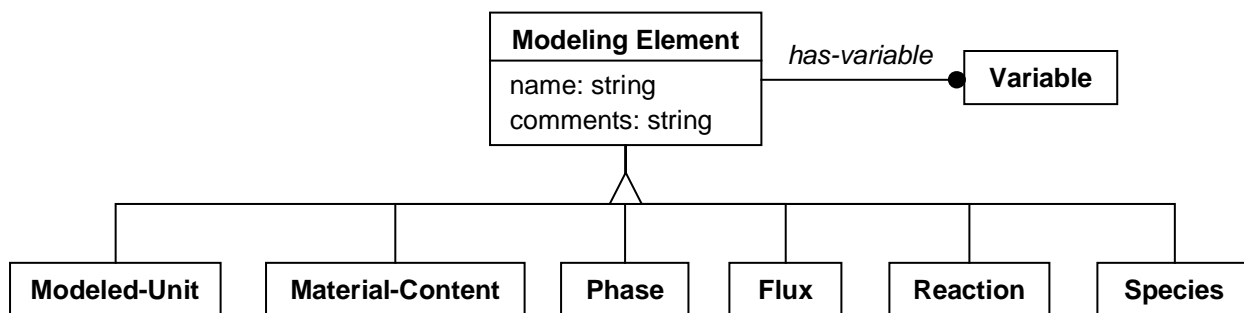


Figure 6-2: Modeling Element Class Object Model

For simplicity, Figure 6-2 does not include any attributes or associations for the modeling element subclasses. These features are introduced incrementally in the remainder of this section. However, before introducing the object models of the modeling element classes, two additional superclasses are first presented. Figure 6-3 illustrates the object model for the class *Species Container*. An instance of a *Species Container* is associated with a set of instances of class *Species*. In a phenomena-based model, these correspond to modeled-units, material-contents, and phases. Therefore, in the object model these modeling element classes appear as subclasses of the *Species Container* class.

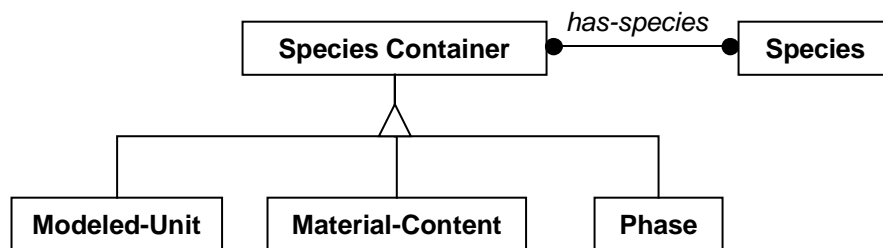


Figure 6-3: Species Container Class Object Model

Similarly, Figure 6-4 illustrates the object model for the class *Reaction Container*. An instance of a *Reaction Container* is associated with a set of instances of class *Reaction*. In a phenomena-based model, these correspond to modeled-units and phases, which consequently appear as subclasses of the *Reaction Container* class.

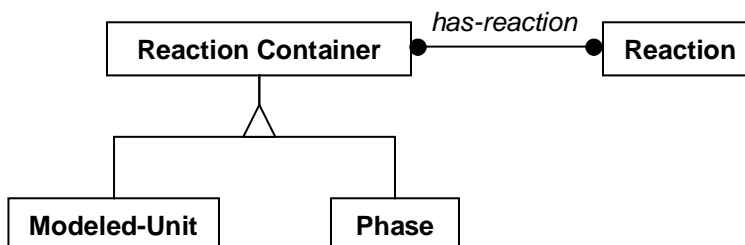


Figure 6-4: Reaction Container Class Object Model

All attributes and associations of the superclasses presented above are inherited by the *Modeled-Unit* class, whose object model is illustrated in Figure 6-5. Additionally, instances of the *Modeled-Unit* class possess five additional types of associations. Hierarchical structure is captured by associations with instances of other modeled-units. Abstraction is indicated by an association with another modeled-unit representing its *parent*, while decomposition is indicated by an association to a set of other modeled-unit representing *subunits*. Alternatively, the material-content of a modeled-unit without subunits is captured by an association to an instance of a *Material-Content*. Topological structure is captured by links to instances of the *Port* class. The *Port* class has a directionality attribute indicating whether it is an input or output to the system. Finally, spatially distributed modeled-units are associated with instances of the *Spatial Distribution* class. This class is decomposed into a *Coordinate System* whose *Rectangular*, *Cylindrical*, or *Spherical* subclasses have boolean attributes indicating which dimensions are distributed, and instances of the *Discretization* class for each distributed dimension, which has attributes indicating the number of discretization *nodes*, *order* of approximation, *minimum* and

maximum domain, and an association to a numerical *Solution Method* selected for the partial differential equations that characterize the system.

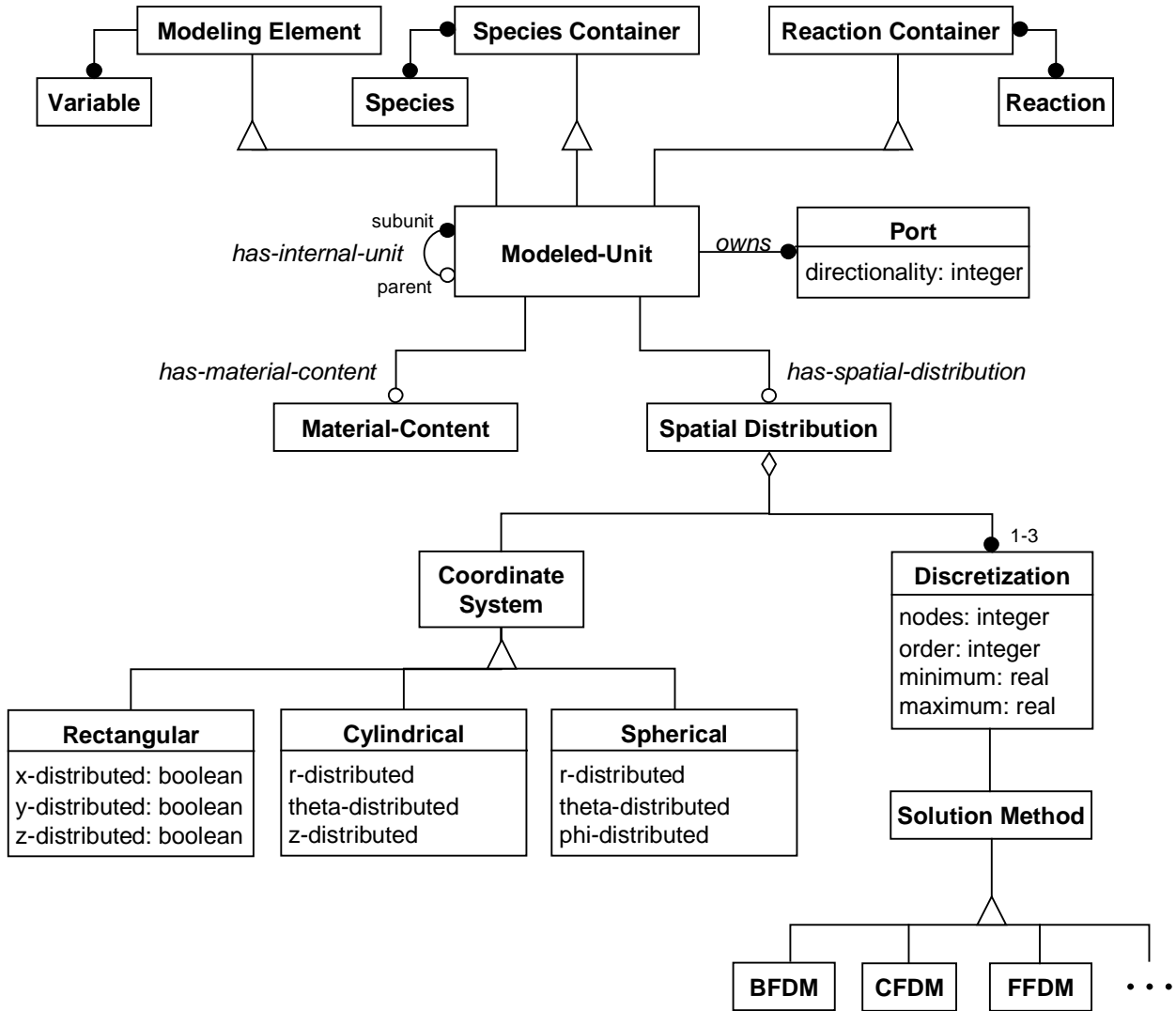


Figure 6-5: Modeled-Unit Class Object Model

The object model for the *Flux* class is illustrated in Figure 6-6. All types of fluxes share three types of associations that represent the topology of a phenomena-based model. The connectivity of a flux is indicated by associations with instances of the *Port* class. In addition, for fluxes attached to modeled-units with a material-content, a flux may be associated with a *Phase* or *Geometry* to which it is allocated. An instance of a flux is characterized as either a *Convective Flux*, an *Energy Flux*, or a *Species Flux*, indicating the type of transport. Transport mechanisms for these types of fluxes are indicated by respective associations with a *Convective Mechanism*, an *Energy Mechanism*, or a *Species Mechanism*. Additionally, *Convective Fluxes* have a state attribute that indicates the physical *state* of the material transported, and an association with an *Equation of State* that characterizes the physical behavior of the material. A *Species Flux* is also associated with a *Species* indicating the selected chemical species being transported.

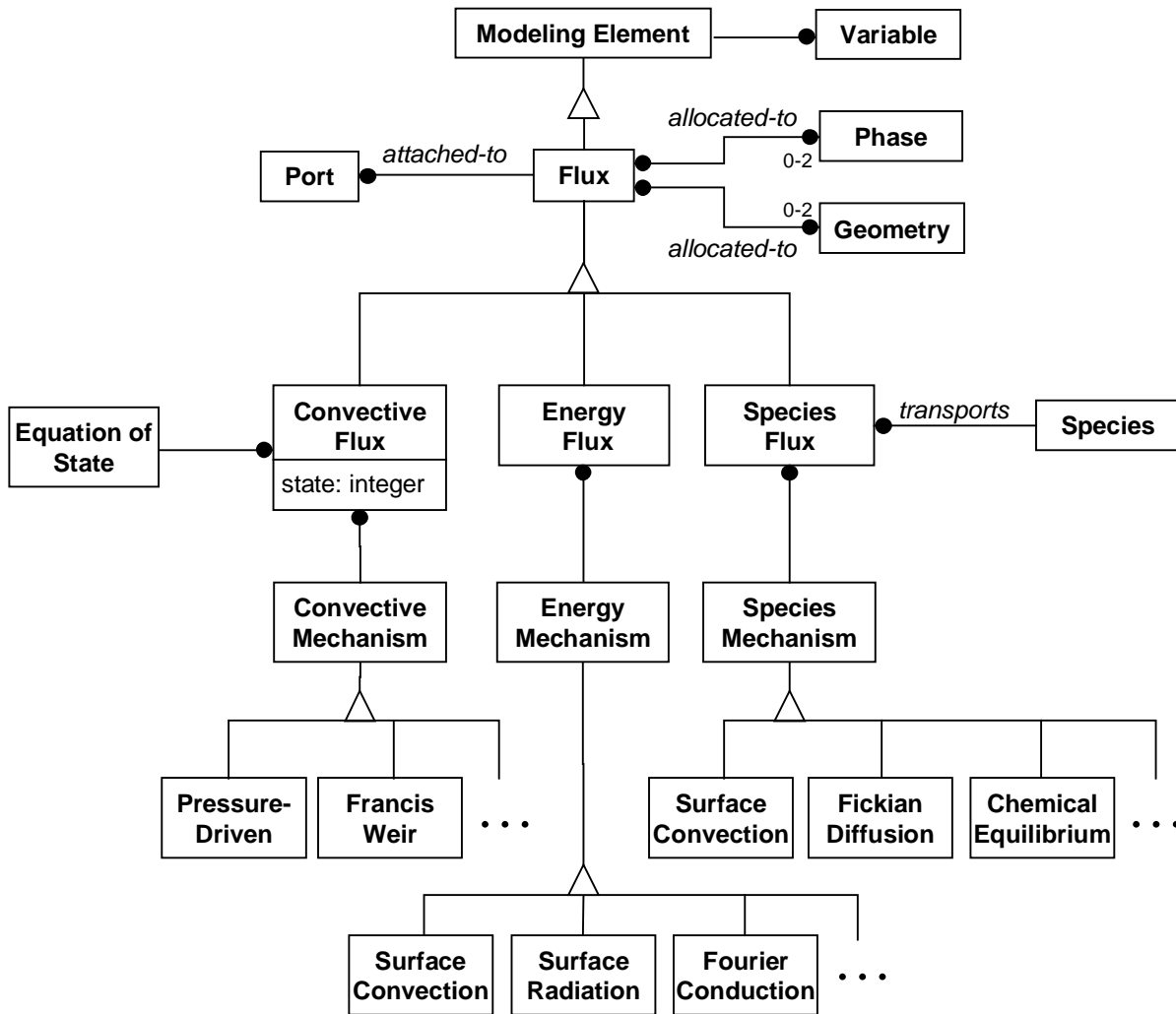


Figure 6-6: Flux Class Object Model

The object models for the *Material-Content* class and the associated *Phase* class are illustrated in Figure 6-7. In addition to inherited *Species* associations, the *Material-Content* class is associated with a *Geometry* class, which represents a geometric characterization of the vessel that contains it. In addition to inherited *Species* and *Reaction* associations, the *Phase* class is associated with an *Equation of State* class or an *Activity Coefficient Model* class that represent the thermodynamic characterization of a phase in a phenomena-based model. As described in the object model for the *Flux* class, both the *Geometry* and the *Phase* classes may be associated with an instance of a *Flux*.

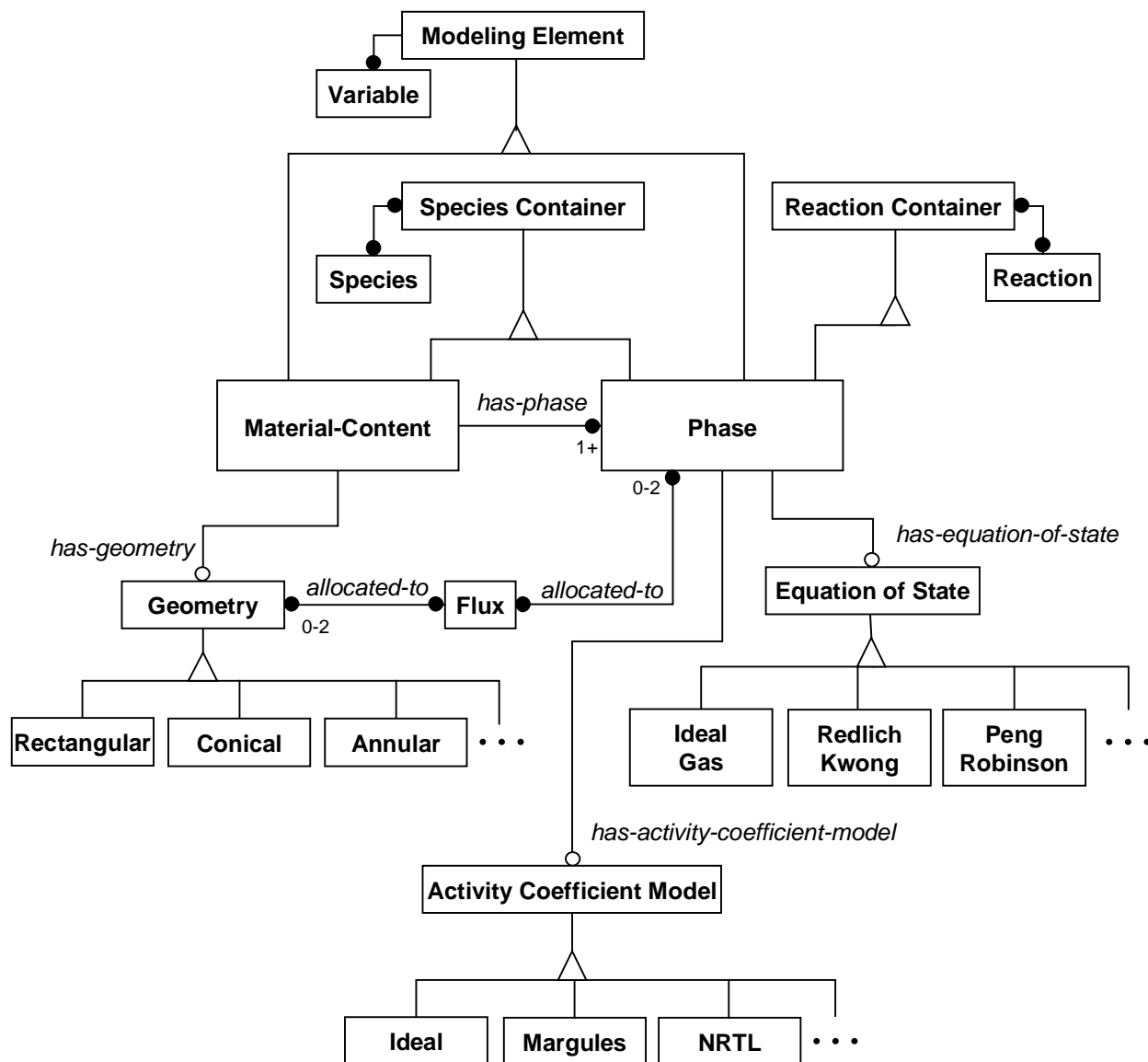


Figure 6-7: Material-Content Class and Phase Class Object Models

The object model for the *Reaction* class is illustrated in Figure 6-8. Each reaction is composed of instances of the *Participant* class, the *Catalyst* class, and the *Rate Law* class. These classes are used to characterize the stoichiometry, participating species, and kinetic rate law for a reaction in a phenomena-based model.

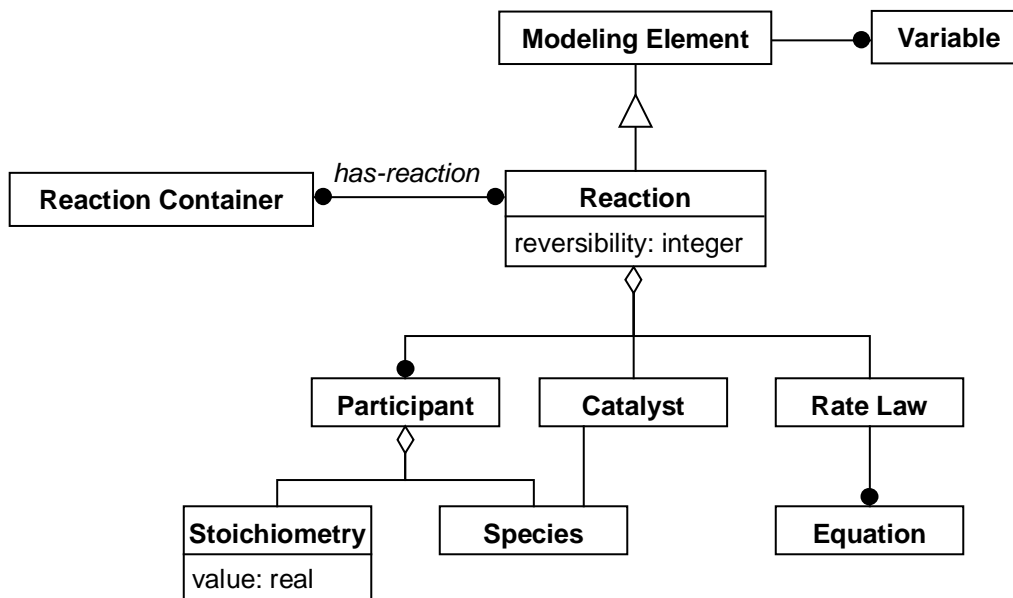


Figure 6-8: Reaction Class Object Model

The final modeling element object model is illustrated in Figure 6-9 for the *Species* class. Each species has a *database ID* attribute that uniquely identifies a species in the *Properties Database* of the MODEL.LA Modeling Environment.

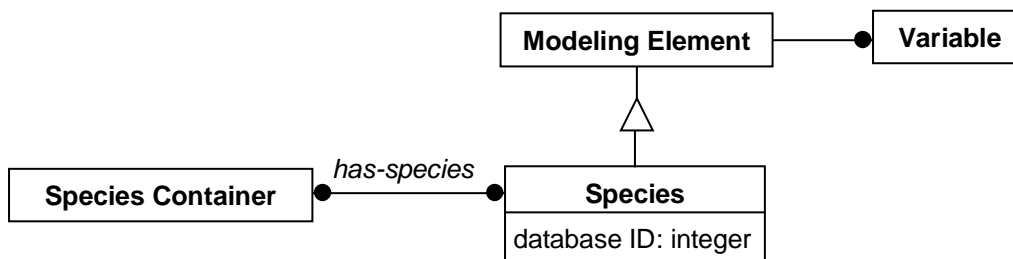


Figure 6-9: Species Class Object Model

The primary associations between the modeling element classes are illustrated in the integrated object model in Figure 6-10. This object model summarizes the structure of instances and associations of modeling element classes that characterize a phenomena-based model.

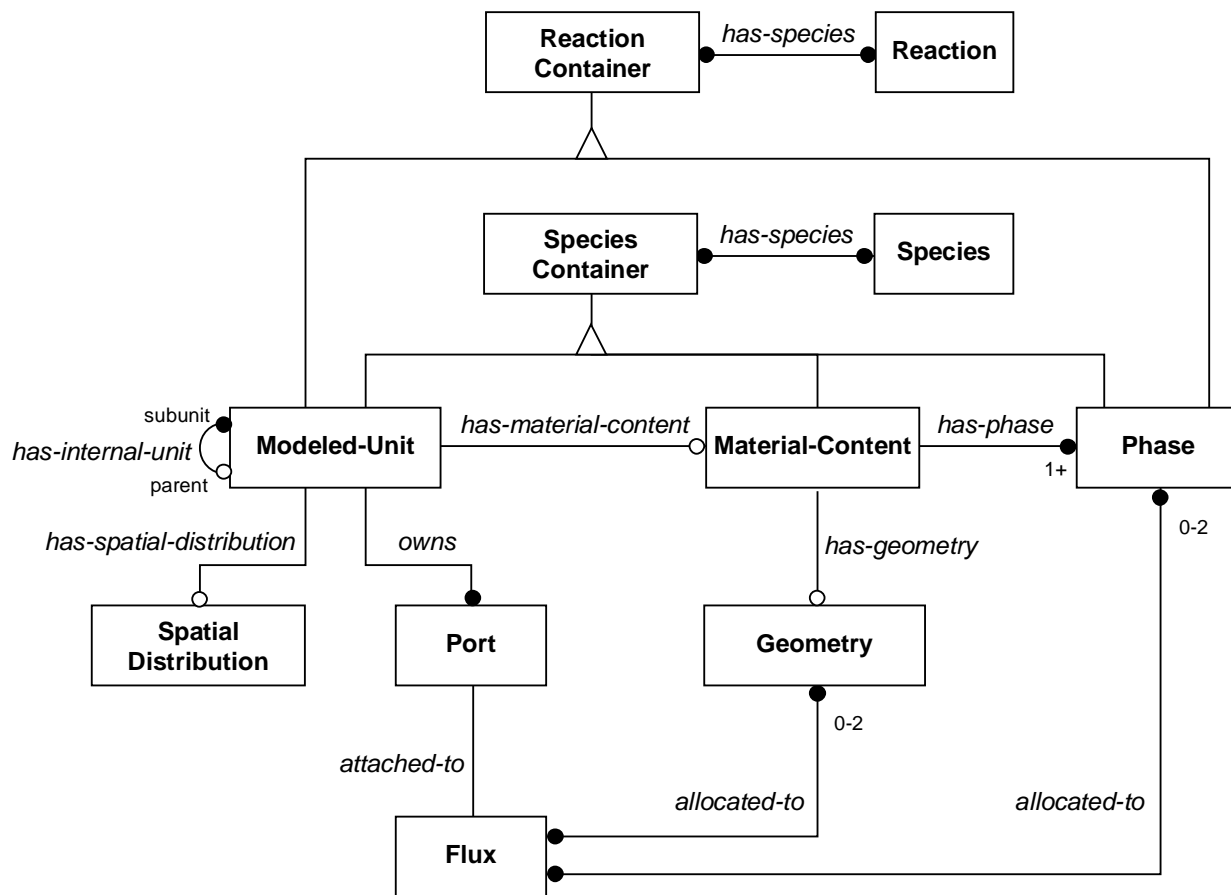


Figure 6-10: Modeling Elements Integrated Object Model

6.3 MODEL.LA Modeling Environment Object Models

In the preceding section, the object models for the MODEL.LA modeling elements were presented. In this section, these object models are integrated into the object models of the MODEL.LA Modeling Environment. The composition of the MODEL.LA object model is illustrated in Figure 6-11. As discussed in Chapter 5, the MODEL.LA environment is composed of the *Model Generator*, the *Properties Manager*, the *Operations Manager*, and the *Numerical Engine*. All of these software elements interact with the modeler through a *Graphical Interface* during the construction of a phenomena-based model and subsequent mathematical model derivation, specification, and solution.

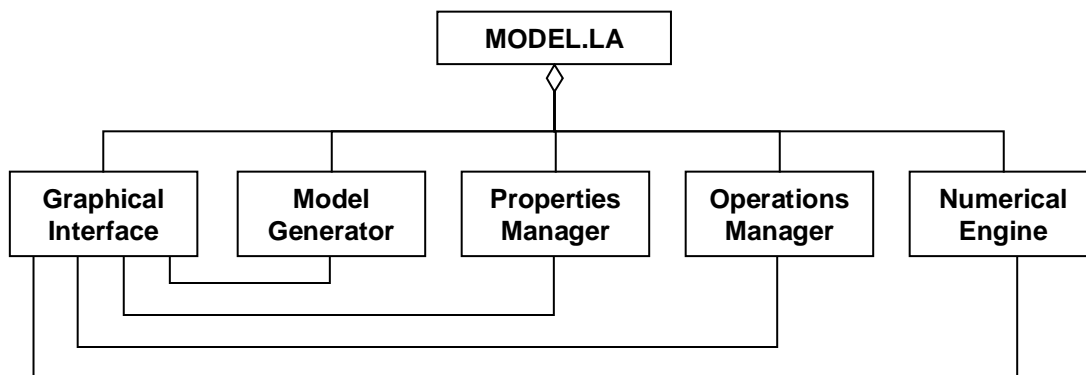


Figure 6-11: MODEL.LA Modeling Environment Object Model

Figure 6-12 illustrates the object model for the *Phenomena-Based Model* class. In addition to associations with the phenomena-based modeling elements, a *Phenomena-Based Model* has attributes that capture the name of the model, the developer, and any textual comments that provide supplementary information about the model.

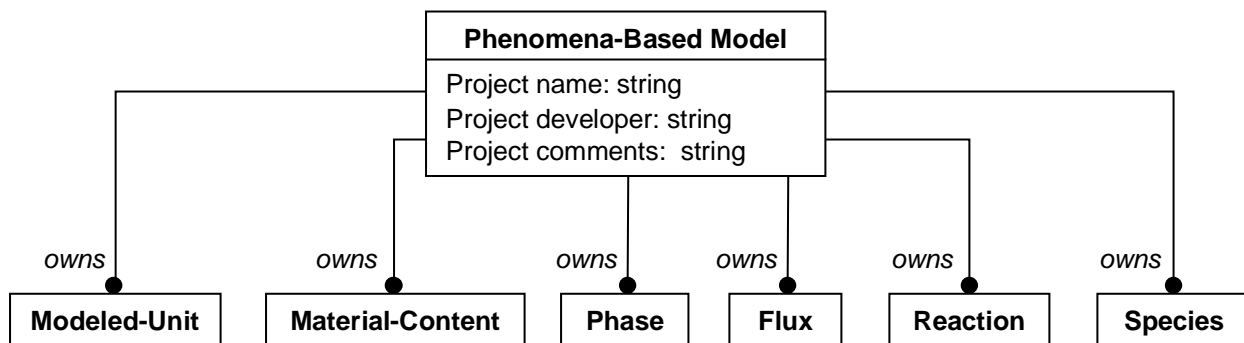


Figure 6-12: Phenomena-Based Model Object Model

In MODEL.LA, phenomena-based model descriptions are used to derive mathematical models. The object model for the *Mathematical Model* class is illustrated in Figure 6-13. Instances of a *Mathematical Model* are composed of instances of *Equations*, *Variables*, and an operational *Schedule*. Subclasses of *Equations* include *Conservation Equations*, *Constitutive Equations*, *Property Correlations*, and *Control Laws*.

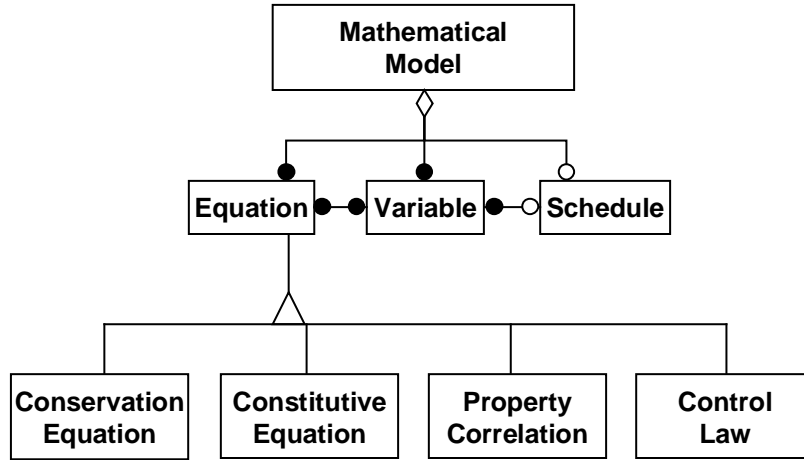


Figure 6-13: Mathematical Model Object Model

The object model for the *Model Generator* is illustrated in Figure 6-14. The *Model Generator* is associated an instance of a *Phenomena-Based Model*. The *Model Generator* uses elements of the *Phenomena-Based Model* to construct the elements of an associated *Mathematical Model*. *Conservation Equations* are constructed and associated with *Modeled-Units* and *Constitutive Equations* are generated and associated with *Modeled-Units*, *Fluxes*, *Reactions*, *Material-Contents*, and *Phases*. The *Model Generator* is also associated with the *Properties Manager*, for generation of *Property Correlations* for *Phases*, and the *Operations Manager* for generation of *Control Laws* and *Operational Schedules*. Finally, an association with the *Numerical Engine* allows solution of the resulting *Mathematical Model*.

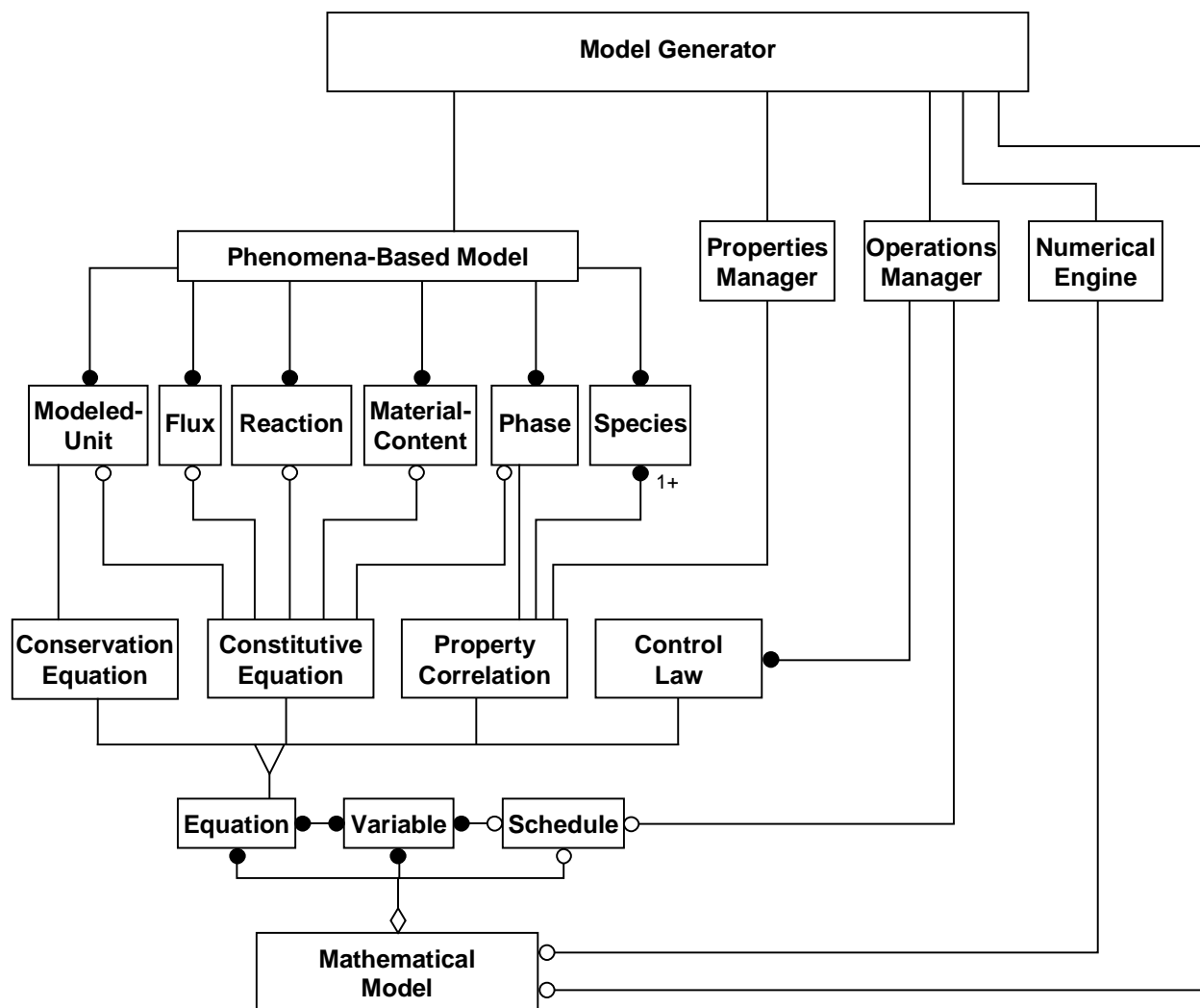


Figure 6-14: Model Generator Object Model

The object model of Figure 6-14 illustrates the structure of the *Model Generator* and its associations with the other elements of the MODEL.LA Modeling Environment, the *Properties Manager*, the *Operations Manager*, and the *Numerical Engine*. In the remainder of this section, the object model for these software elements are presented.

The object model for the *Properties Manager* is illustrated in Figure 6-15. The *Properties Manager* is associated with a *Phase* and the set of *Species* contained in the phase. Characterizations of the phase and its species enable construction of a *Property Correlation* for the thermodynamic or physical property identified by attribute *property ID*. Pure *Species Properties* relationships are obtained through an association with a *Properties Database*. When necessary, a *Departure Function* or *Excess Property* relationship is constructed by the

appropriate *Equation of State Model Manager* or *Activity Coefficient Model Manager*.

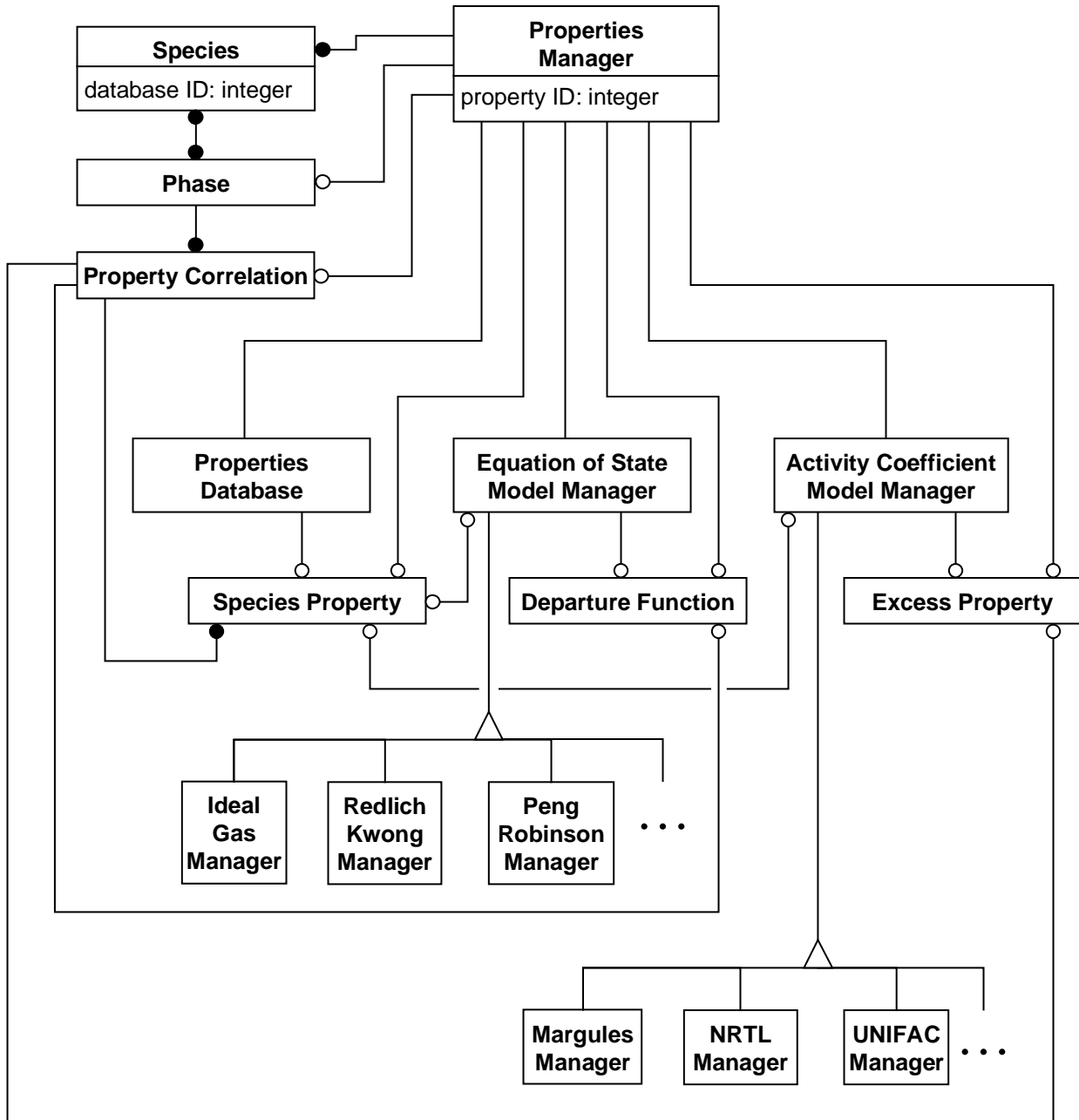


Figure 6-15: Properties Manager Object Model

The object model for the *Properties Database* of the *Properties Manager* is illustrated in Figure 6-16. The *Properties Database* is composed of *DIPPR Data* tables (consisting of identification data, constant property data, and temperature-dependent property correlation data), *UNIFAC Data* tables for VLE and LLE activity coefficient models, and *Binary Parameter Data* tables for binary species interaction parameters for equations of state and activity coefficient models.

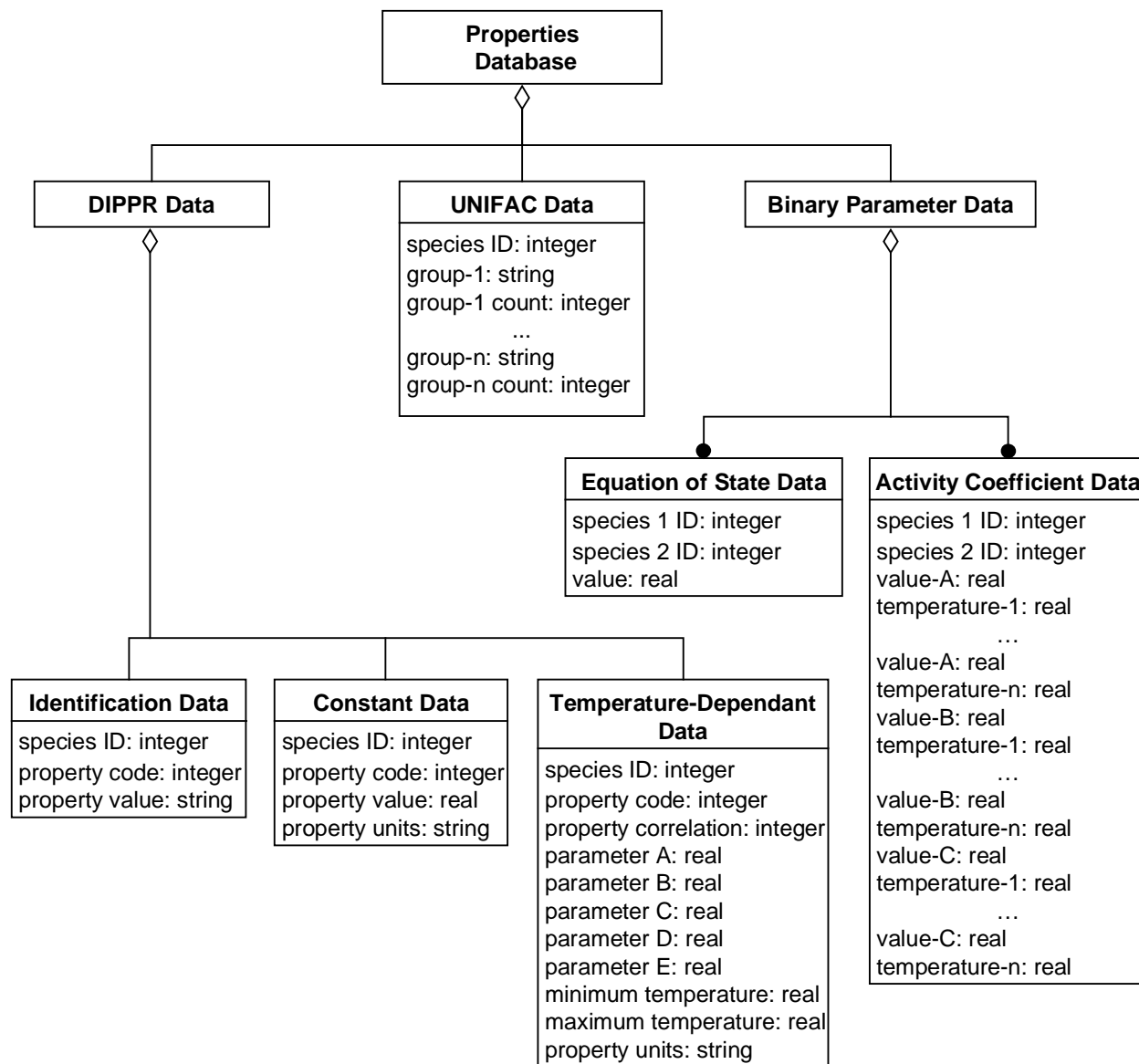


Figure 6-16: Properties Database Object Model

The object model for the *Operations Manager* is illustrated in Figure 6-17. The *Operations Manager* is associated with instances of *Controllers*, *Transmission Lines*, *Tasks*, and *Schedules*. Each *Controller* is associated with *Transmission Lines* and a *Control Law*. Each *Transmission Line* is associated with a *Manipulated Variable* or a *Measured Variable*. Each *Task* is also associated with *Transmission Lines*. A *Schedule* is composed of *Events* and *Actions*. *Events* are associated with *Tasks*, whose *Measured Variables* are used to construct *Conditionals* for *When* events and *While* events. *Events* trigger associated *Actions*. *Elementary* Actions are characterized by a change in an associated *Manipulated Variable*. *Condition* events are associated with a *Conditional* that determines the path taken at a branch in a schedule. Finally, a *Composite* event is itself associated with an instance of a *Schedule*, allowing a hierarchical composition of complex schedules.

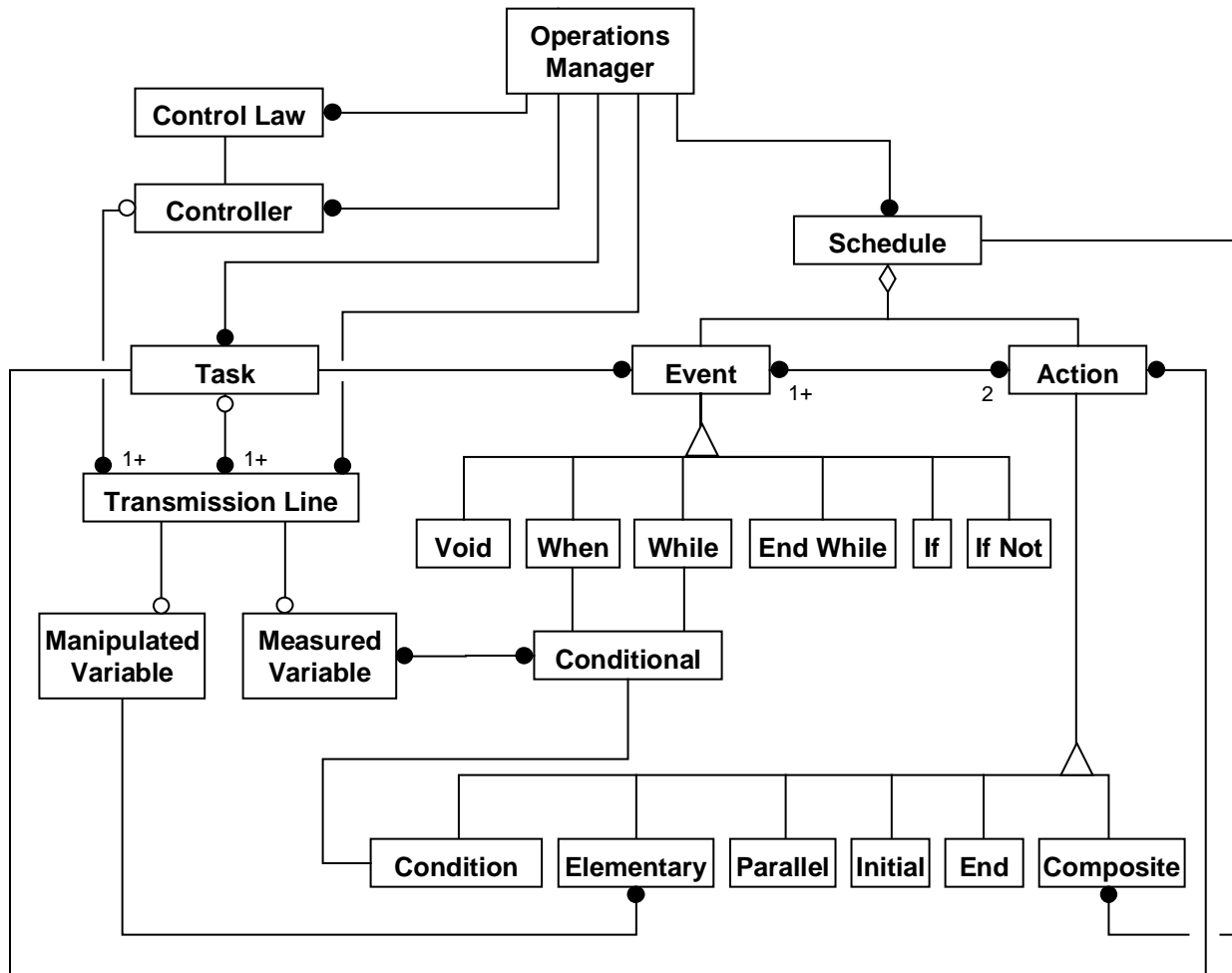


Figure 6-17: Operations Manager Object Model

The object model for the *Numerical Engine* is illustrated in Figure 6-18. In addition to the *Mathematical Model* containing model *Equations*, *Variables*, and operational *Schedule*, the Numerical Engine is associated with a *Degrees of Freedom Incidence Matrix* for consistent specification of design variables and detection of high index model formulations, an *Initial Conditions Incidence Matrix* for consistent specification of initial conditions for dynamic models, *gPROMS* for solution of the mathematical model, and a *Results Display* for tabular and graphical analysis of the resulting model behavior.

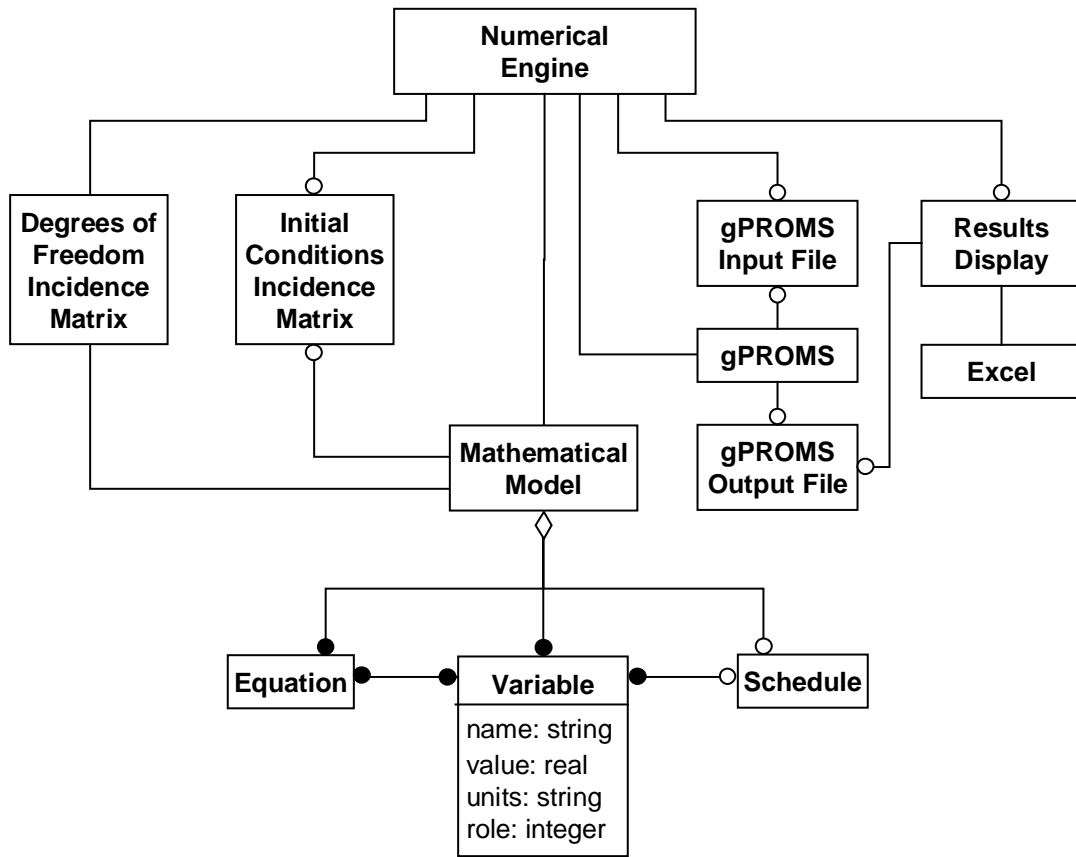


Figure 6-18: Numerical Engine Object Model

The object models described in this section and the previous section specify the static structure of the MODEL.LA Modeling Environment. These object models provide a basis for the phenomena-based modeling activities. In the following section, the functional model of the modeling environment is discussed, illustrating how these elements are used during subsequent mathematical model derivation and solution.

6.4 Functional Model of the MODEL.LA Modeling Environment

The functional models of the object modeling technique capture how output values are calculated from input values in a program. It consists of multiple *data flow diagrams* which show the flow of values from external inputs, through operations and internal data stores, to external outputs. A data flow diagram consists of *processes* (represented by ellipses) that transform data, data flows (represented by arrows) that move data, actor objects (represented by rectangles) that produce and consume data, and internal data stores (represented by two horizontal lines) that hold data for subsequent use.

A high-level functional model describing mathematical model derivation and solution is illustrated in Figure 6-19. The *Phenomena-Based Model* is represented as a static data store. *Assumptions* from the Phenomena-Based Model are used to derive the *model equations*. *Variable specifications* (i.e., design variables and initial conditions) for these equations are combined which the equations for model solution. The resulting *numerical results* are stored in the *Model Behavior* data store for subsequent analysis.

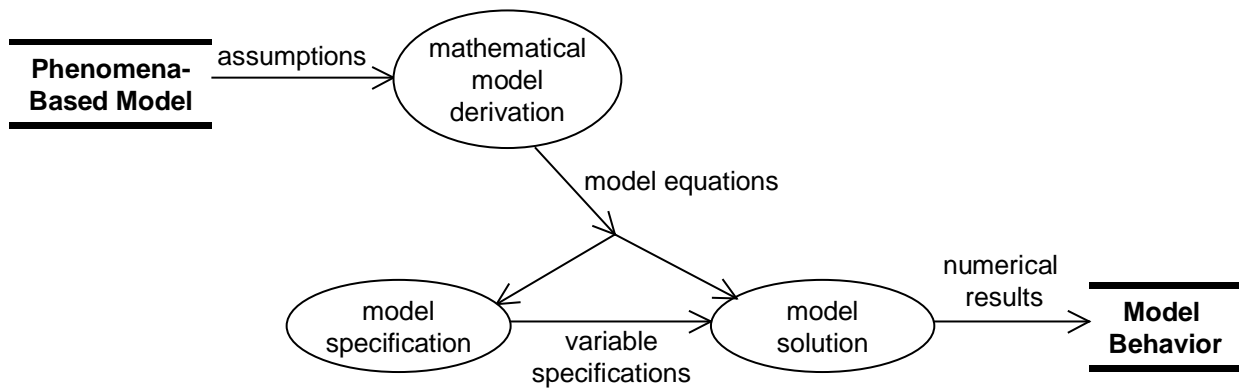


Figure 6-19: Overall Model Derivation and Solution Functional Model

In the remainder of this section, the abstract processes of this overall functional model are further specified by refined functional models.

Figure 6-20 illustrates the functional model for the process of mathematical model derivation. The phenomena-based *model assumptions* are used to derive *conservation equations*, *constitutive relationships*, and *property correlations*. Additionally, the *Operations Manager* provides *control structures* for construction of *control laws*, and the active schedule is used to generate an *operational schedule*, thus completing the mathematical model.

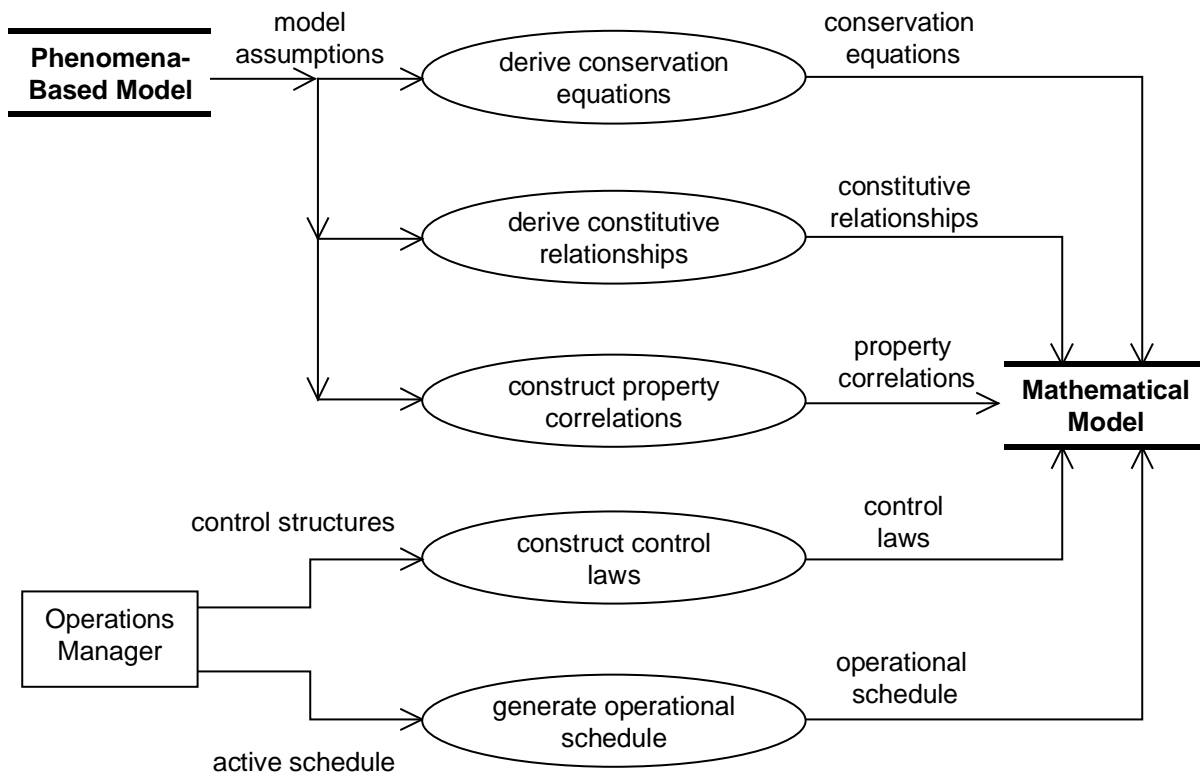


Figure 6-20: Model Derivation Functional Model

The functional model for the process that generates a property correlation is illustrated in Figure 6-21. The *Model Generator* supplies a *phase* for which the desired property correlation is to be constructed. The list of assumed species is extracted and stored in the *Phase Species* data store. Properties of these pure species are accessed from a *Property Database* using the proper *species id*. These pure species correlations are stored in a *Pure Species Properties* data store. When necessary, the assumed *equation of state model* or *activity coefficient model* is extracted from the phase and used to construct the appropriate *departure function*, *excess property relationship*, or overall *property correlation*. Relevant data from these processes are combined to form the overall *phase property correlation* which is added to the *Mathematical Model* data store.

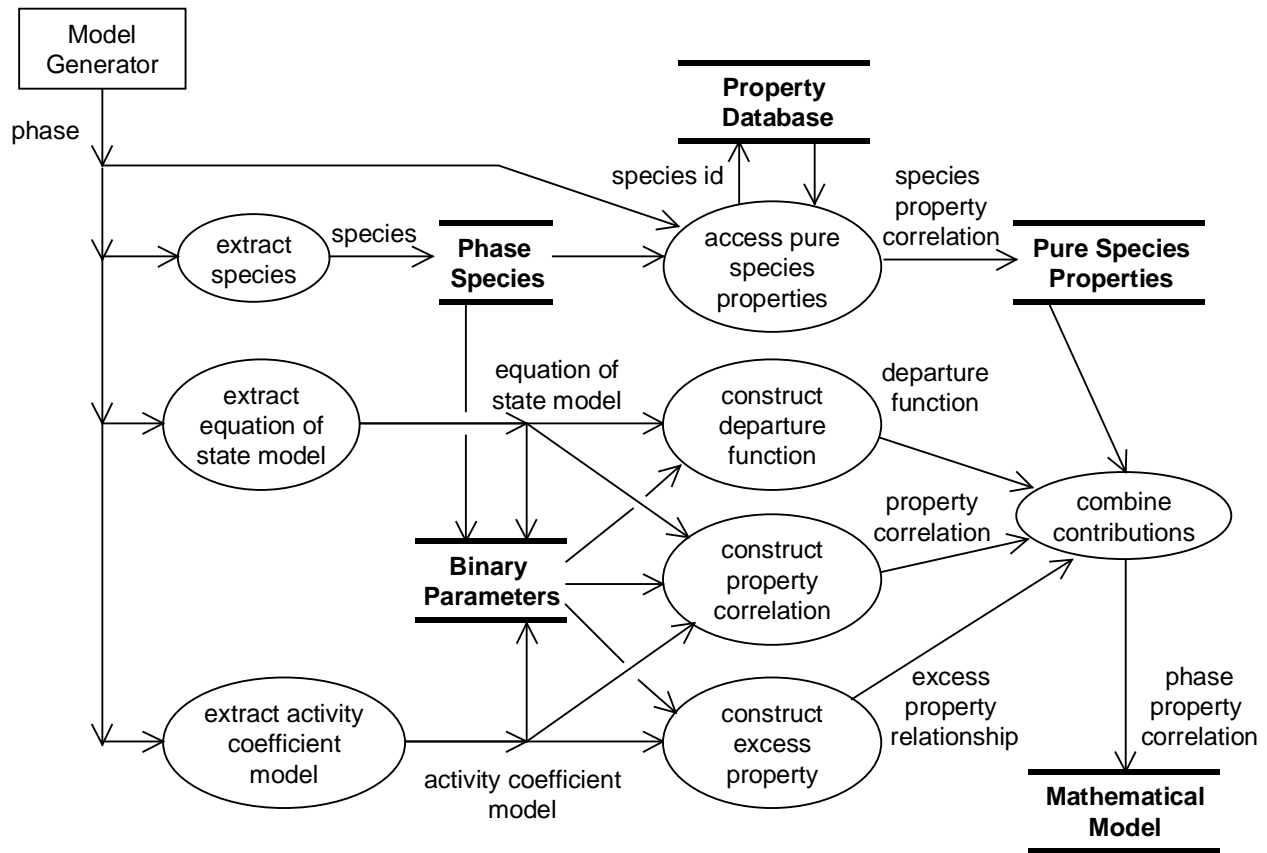


Figure 6-21: Property Correlation Generation Functional Model

The functional model for mathematical model specification and solution is illustrated in Figure 6-22. *Model equations* from the *Mathematical Model* are used for specification of degrees of freedom. An incidence matrix is used to maintain a structurally consistent set of selected *design variables*. If the model is dynamic, once design variables have been selected, a structural index analysis is performed on the model equations before specification of *initial conditions*. The structurally well-posed mathematical model is then solved. Numerical results are stored for subsequent display and analysis of the behavior of the model.

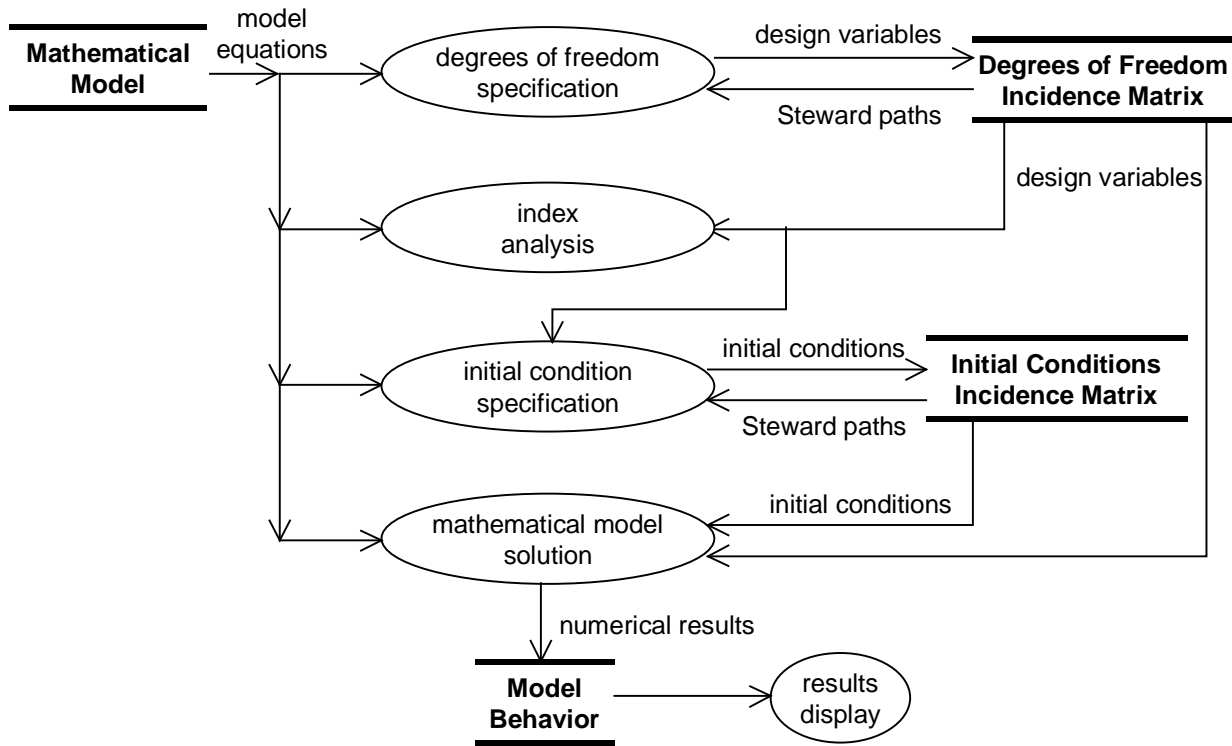


Figure 6-22: Mathematical Model Specification and Solution Functional Model

Finally, the functional model for mathematical model solution is illustrated in Figure 6-23. The mathematical model is combined with selected design variables and initial conditions to formulate a *gPROMS input file*. *gPROMS* is then executed for model solution and results are stored in a *gPROMS output file*. These numerical results are then processed by the *Numerical Engine* of MODEL.LA.

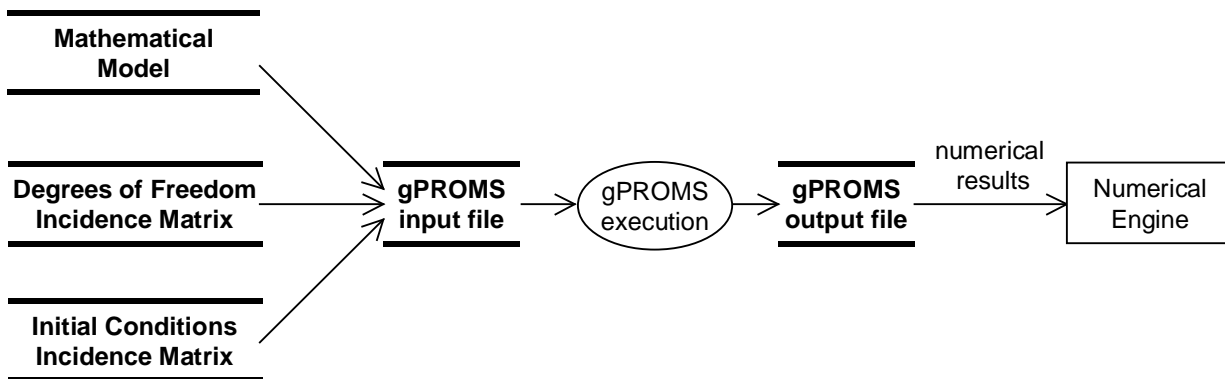


Figure 6-23: Model Solution Functional Model

6.5 Summary of MODEL.LA Modeling Environment Software Design

This chapter has presented an overview of the software design of the MODEL.LA Modeling Environment using the object modeling technique. The static structure of MODEL.LA has been presented using object models. These object models illustrate how object classes are used to represent the phenomena-based modeling elements and the associated software elements. Functional models were then presented to illustrate the use of these object classes during model derivation and solution. In order to demonstrate the resulting utility of the phenomena-based modeling approach embodied by the MODEL.LA Modeling Environment, the following chapter illustrates its use in application to a variety of chemical process modeling examples.

Chapter 7

Phenomena-Based Modeling Examples

The previous two chapters provided an overview of the functionality, graphical user interface, structure, and software design of the phenomena-based MODEL.LA Modeling Environment. In this chapter, the practical use of this environment is illustrated through application to a series of phenomena-based modeling examples. These examples include:

1. The hierarchical design of a chemical plant for the hydrodealkylation of toluene to produce benzene,
2. The hierarchical design and economic analysis of a chemical plant for the production of acetic anhydride from acetone and acetic acid,
3. A study of the open-loop and closed-loop behavior of a distillation column with side stripper for the separation of a mixture of benzene, toluene, and o-xylene,
4. A dynamic model of a chemical plant with a 1-D spatially distributed reaction and separation unit operations, and
5. A dynamic model of a 2-D spatially distributed tubular reactor with a cooling jacket.

These examples were selected to illustrate the use of the MODEL.LA Modeling Environment in modeling non-trivial processes in order to facilitate comparison of the phenomena-based modeling approach with existing computer-aided modeling approaches.

7.1.1 HDA Plant

The HDA plant case study illustrates the hierarchical design of a plant for the hydrodealkylation of toluene to produce benzene. It is based on an example given in Douglas (1988), which is an

adaptation of an AIChE student design problem described in (McKetta, 1977). Design objectives and constraints are given in Table 7-1.

Table 7-1: Design Objectives for HDA Plant

1. Reaction information
a. Reactions:
$\text{Toluene} + \text{H}_2 \rightarrow \text{Benzene} + \text{CH}_4$ $2 \text{ Benzene} \leftrightarrow \text{Diphenyl} + \text{H}_2$
b. Reaction inlet temperature > 1150 °C (to get a reasonable reaction rate); reactor pressure = 500 psia
c.
$\text{Selectivity} = \frac{\text{Moles Benzene at Reactor Outlet}}{\text{Moles Toluene Converted}} = S$ $\text{Conversion} = \frac{\text{Moles Toluene Converted in Reactor}}{\text{Moles Toluene Fed to Reactor}} = x$ $S = 1 - \frac{0.0036}{(1-x)^{1.544}}, \quad x < 0.97$
d. Gas phase
e. No catalyst
2. Production rate of benzene: 265 mol/hr
3. Product purity of benzene: $x_D=0.9997$
4. Raw materials: Pure toluene at ambient conditions; H ₂ stream containing 95% H ₂ , 5% CH ₄ at 550 psia, 100°F
5. Constraints: H ₂ /aromatic ≥ 5 at the reactor inlet (to prevent coking); reactor outlet temperature < 1300°F (to prevent hydrocracking); rapidly quench reactor effluent to 1150°F, $x < 0.97$ for the product distribution correlation

In this example, the base case design of the HDA plant is modeled. In order to concentrate on phenomena-based modeling aspects in this initial example, the economic analysis is neglected. However, if desired, pricing and cost correlations may be readily added to the model as user-defined equations, as is done for the subsequent Acetic Anhydride plant example.

Following the hierarchical design methodology for a continuous plant, the plant is first modeled at an abstract input-output level, where only raw material, product, byproduct, and waste streams are included. Thus, the HDA Plant illustrated in Figure 7-1 is modeled with two convective input streams for raw materials hydrogen and toluene, two convective output streams for product benzene and byproduct diphenyl, and a convective output stream representing a gaseous purge of methane. Complete recycle of toluene and hydrogen is assumed. Declaration of the two reactions of interest and their assignment to the HDA Plant are also illustrated in Figure

7-1.

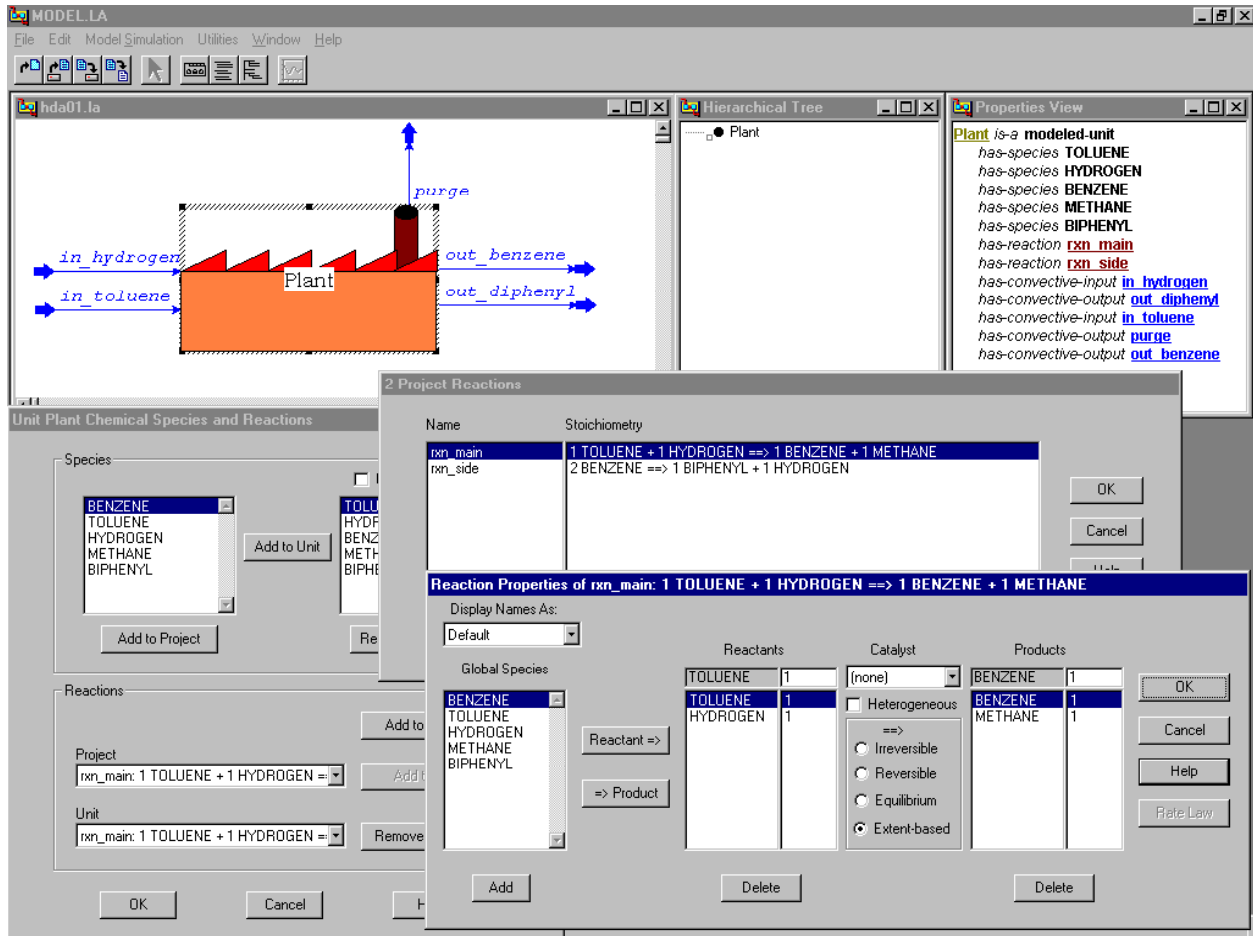


Figure 7-1: Input-Output Level Design for HDA Plant

Continuing with the hierarchical design approach, the input-output level view of the HDA plant is decomposed into a reaction section and a separation section (as illustrated in window *Plant* of Figure 7-2). Raw materials feeds to the plant are allocated to the reaction section, and an effluent stream is declared from the reaction section to the separation section. Vapor and liquid recycle streams for hydrogen and toluene, respectively, are also declared. For convenience, a mixing point is defined before the reaction section, so that the 5/1 ratio of H₂/Aromatics at the reactor inlet may be readily specified, and a split point is defined for the vapor recycle to establish a purge stream of the same composition as the gaseous recycle stream.

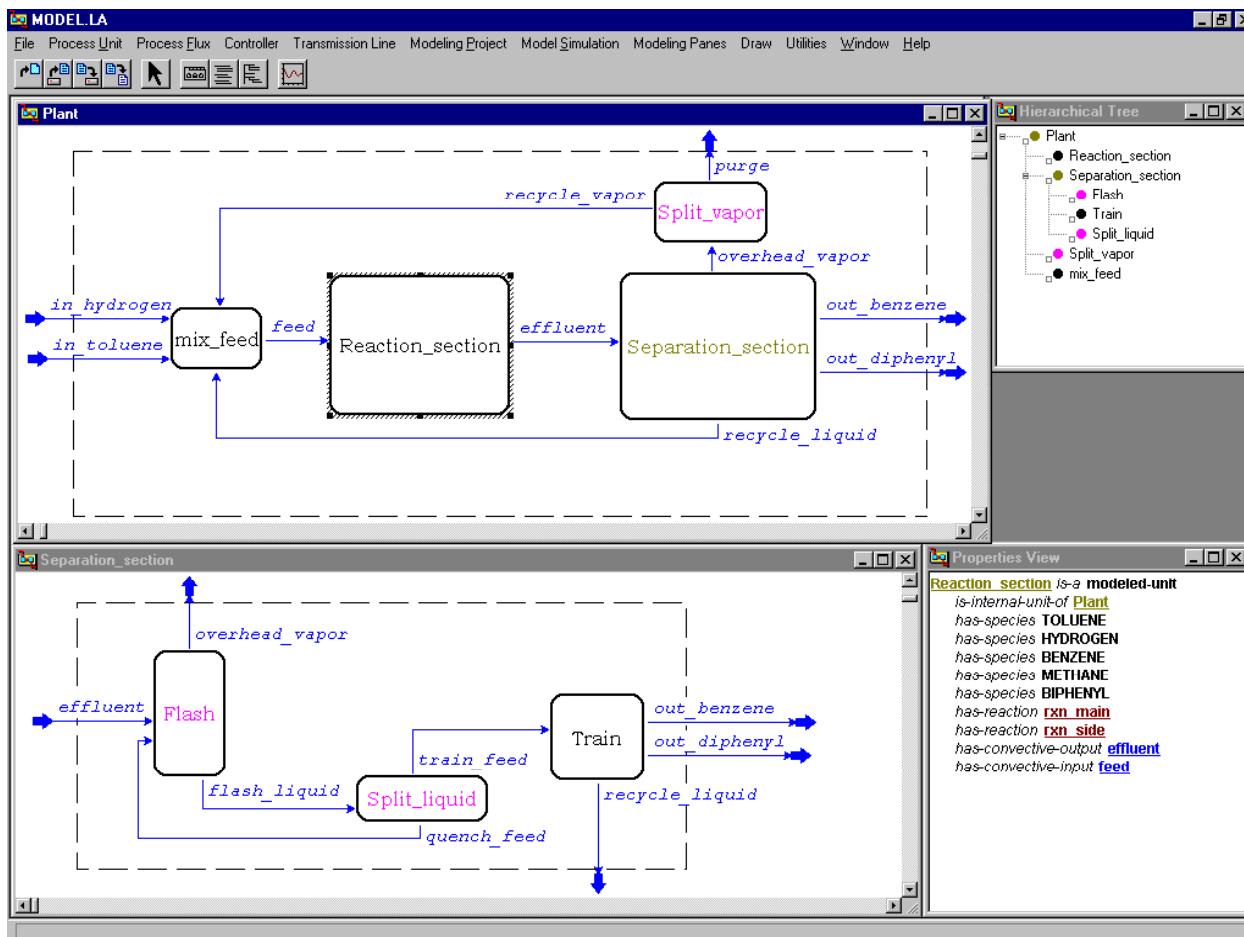


Figure 7-2: Reaction and Separation Section Design for HDA Plant

A preliminary structure for the design of the separation section is illustrated in window *Separation_section* of Figure 7-2. The effluent stream from the reaction section is fed to a flash (where vapor and liquid phases are modeled assuming ideal thermodynamic behavior). The vapor stream from the flash is recycled, and the liquid stream (after a portion is split to serve as a quench for the reactor effluent) is fed to a liquid separation section (i.e., *Train*) where toluene, benzene, and diphenyl are to be separated. Simulation of this design, however, indicates that trace amounts of methane are present in the liquid stream from the flash. Consequently, methane must also be separated in the liquid separation section. This requires an additional output stream from the liquid separation section. To accommodate this, however, the overall input-output level design of the HDA plant must first be modified (as illustrated in window *hda04.la* of Figure 7-3). The methane output stream, *out_methane*, is then allocated to the separation section, and subsequently the liquid separation section. For this base case design, the liquid separation section is modeled as

a distillation train, as illustrated in window *Train* of Figure 7-3. A series of three columns are proposed to separate methane, benzene, and diphenyl as products, before recycling the remaining toluene.

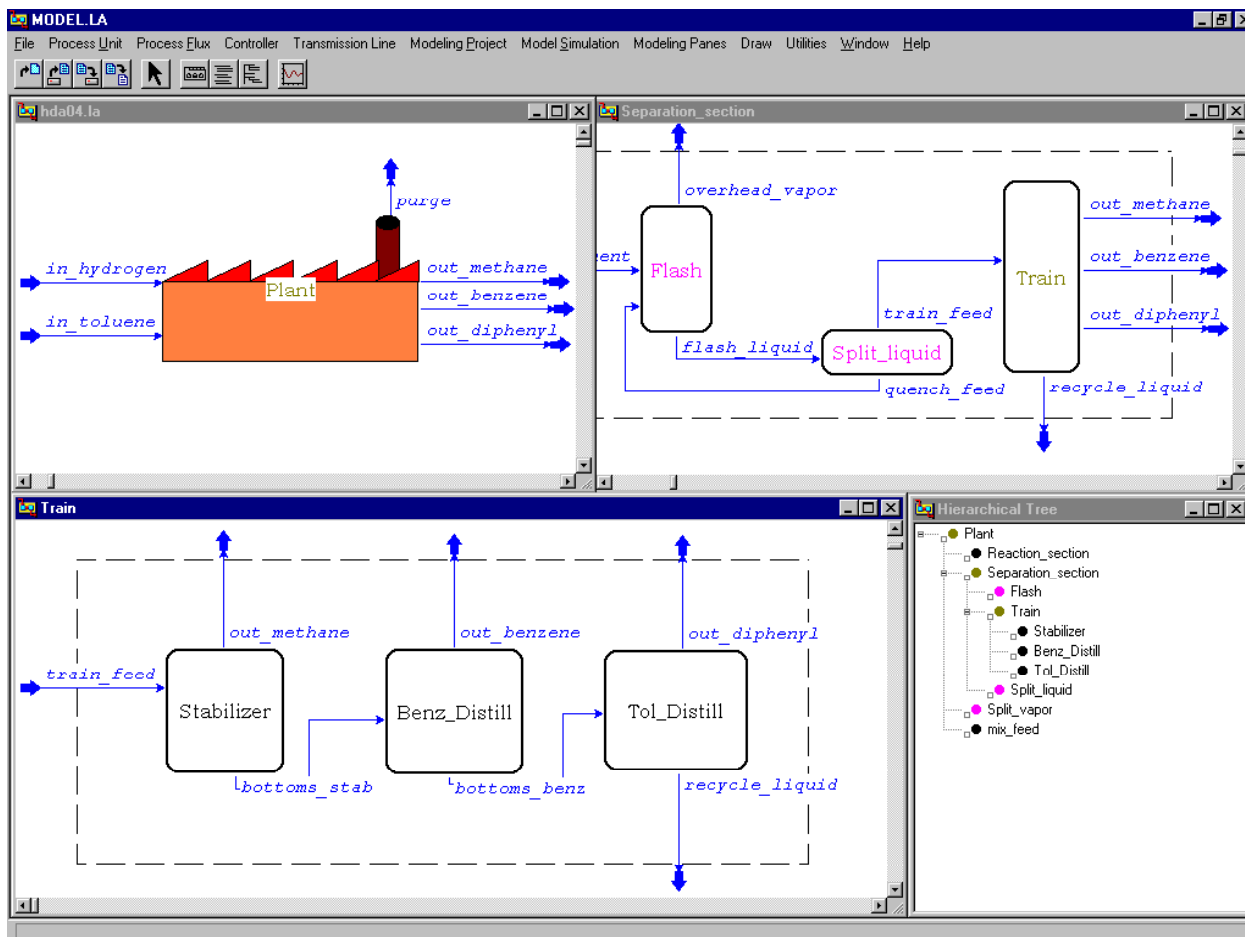


Figure 7-3: Separation Section Design for HDA Plant

Until this point in the design, only mass balances have been considered in modeling the HDA plant. To consider energy integration, energy balances must also be included. The resulting design of the reaction section, with energy integration, is illustrated in window *Reaction_section* of Figure 7-4. The feed stream to the furnace is first preheated in a heat exchanger. The preheated feed is further heated to the specified inlet temperature in a furnace, requiring a heat duty modeled as an energy input stream. The reactor effluent is then quenched with a portion of the liquid stream from the flash in the separation section. The resulting stream is used to preheat the reactor feed in the heat exchanger, before being cooled. The resulting material consists of a vapor and liquid at equilibrium. This is modeled as two separate streams that are fed to the

separation section. Energy integration is then considered for the separation section (illustrated in window *Separation* of Figure 7-4) of the HDA Plant. Here, the energy required to compress the vapor recycle stream and pump the liquid recycle stream are also included in the model.

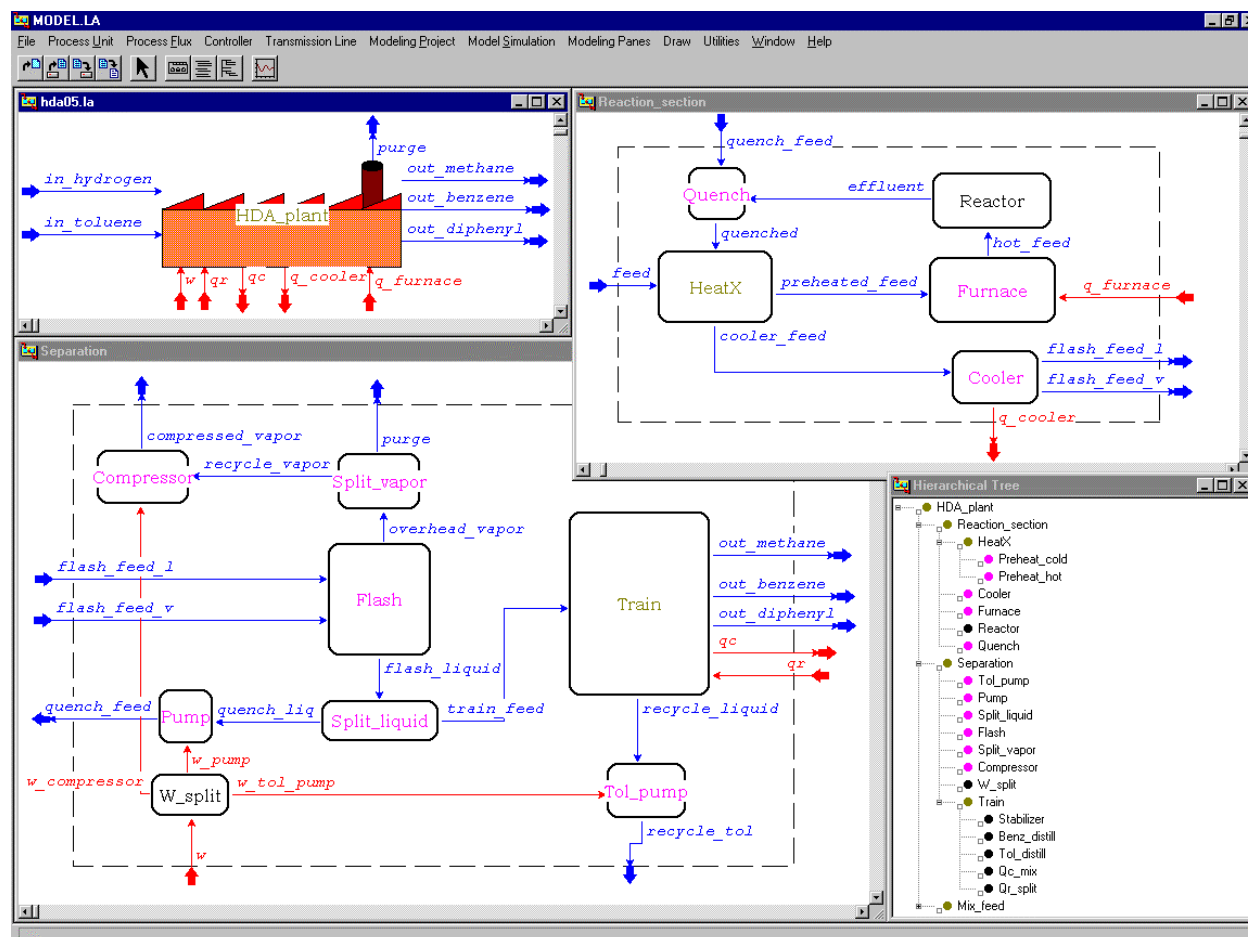


Figure 7-4: Reaction Section with Energy Integration for HDA Plant

The details for each of the distillation columns of the distillation train are then specified. Each column is modeled in a similar manner, illustrated by the benzene distillation column shown in Figure 7-5. Each distillation column consists of a column, a condenser, and a reboiler. Energy fluxes from the condenser and to the reboiler are declared. The column consists of a rectifying section and a stripping section, which are both modeled as staged units. Each stage in the rectifying section and stripping section, along with the condenser and reboiler, are modeled as material modeled-units with ideal vapor and liquid phases at equilibrium. Material is fed to the top stage in the stripping section, and withdrawn from the liquid phases of the condenser and reboiler, respectively. Vapor flows upward through the column through a series of vapor

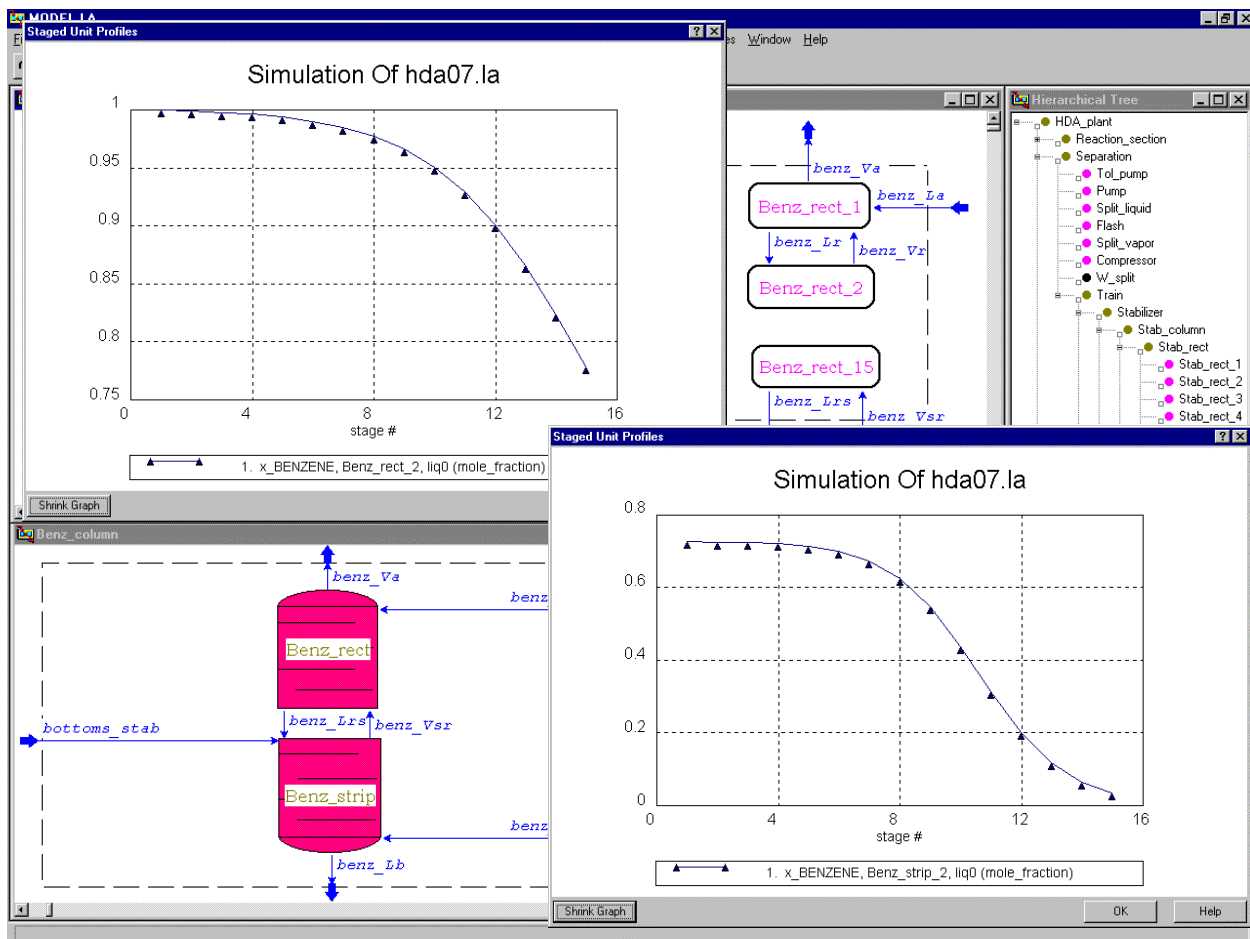


Figure 7-6: Simulation Results for HDA Plant Base Case Design

7.1.2 Acetic Anhydride Plant

This example is based on an AIChE student design problem described in (McKetta, 1977). The primary design objective is to design a plant for the production of acetic anhydride from raw materials acetone and acetic acid. Additional design objectives and constraints are given in Table 7-2. As in the previous example, the plant is designed hierarchically. The initial input-output level design of the plant is shown in Figure 7-7. At this level the plant has two raw material streams, a product stream, and a gaseous stream for reaction byproducts. Complete recycle of raw materials is assumed. Declaration of the three reactions of interest and their assignment to the overall plant are illustrated in Figure 7-8. Expressions for yield and conversion are entered as user-defined equations.

Table 7-2: Design Objectives for Acetic Anhydride Plant

1. Reaction information						
<p>a. Reactions:</p> $\text{Acetone} \rightarrow \text{Ketene} + \text{Methane}$ $2 \text{ Ketene} \rightarrow \text{Ethylene} + 2 \text{ CO}$ $\text{Ketene} + \text{Acetic Acid} \rightarrow \text{Acetic Anhydride}$						
<p>b. Furnace specifications (to crack acetone):</p> <p style="text-align: center;">temperature = 700 °C gas phase</p>						
<p>c. Quench reactor specifications (to produce acetic anhydride):</p> <p style="text-align: center;">Temperature = 80 °C liquid phase acetic acid concentration = 50 mol %</p>						
<p>d.</p> $\text{Ketene Yield} = 1 - \frac{\text{Moles Carbon Monoxide Formed in Furnace}}{\text{Moles Acetone Converted in Furnace}}$ $\text{Acetone Conversion} = \frac{\text{Moles Acetone Converted in Furnace}}{\text{Moles Acetone Fed to Furnace}}$ $\text{Ketene Yield} = 1 - \frac{4}{3} \text{ Acetone Conversion}$						
2. Production rate of acetic anhydride: 16.58 lb-mol/hr						
3. Product purity of acetic anhydride: 99 mole %						
4. Raw materials: Pure acetone and acetic acid at ambient conditions						
5. Cost data:						
<table> <tr> <td>a. Acetone:</td> <td>\$24/lb-mol</td> </tr> <tr> <td>b. Acetic Acid:</td> <td>\$23/lb-mol</td> </tr> <tr> <td>c. Acetic Anhydride:</td> <td>\$51.50/lb-mol</td> </tr> </table>	a. Acetone:	\$24/lb-mol	b. Acetic Acid:	\$23/lb-mol	c. Acetic Anhydride:	\$51.50/lb-mol
a. Acetone:	\$24/lb-mol					
b. Acetic Acid:	\$23/lb-mol					
c. Acetic Anhydride:	\$51.50/lb-mol					

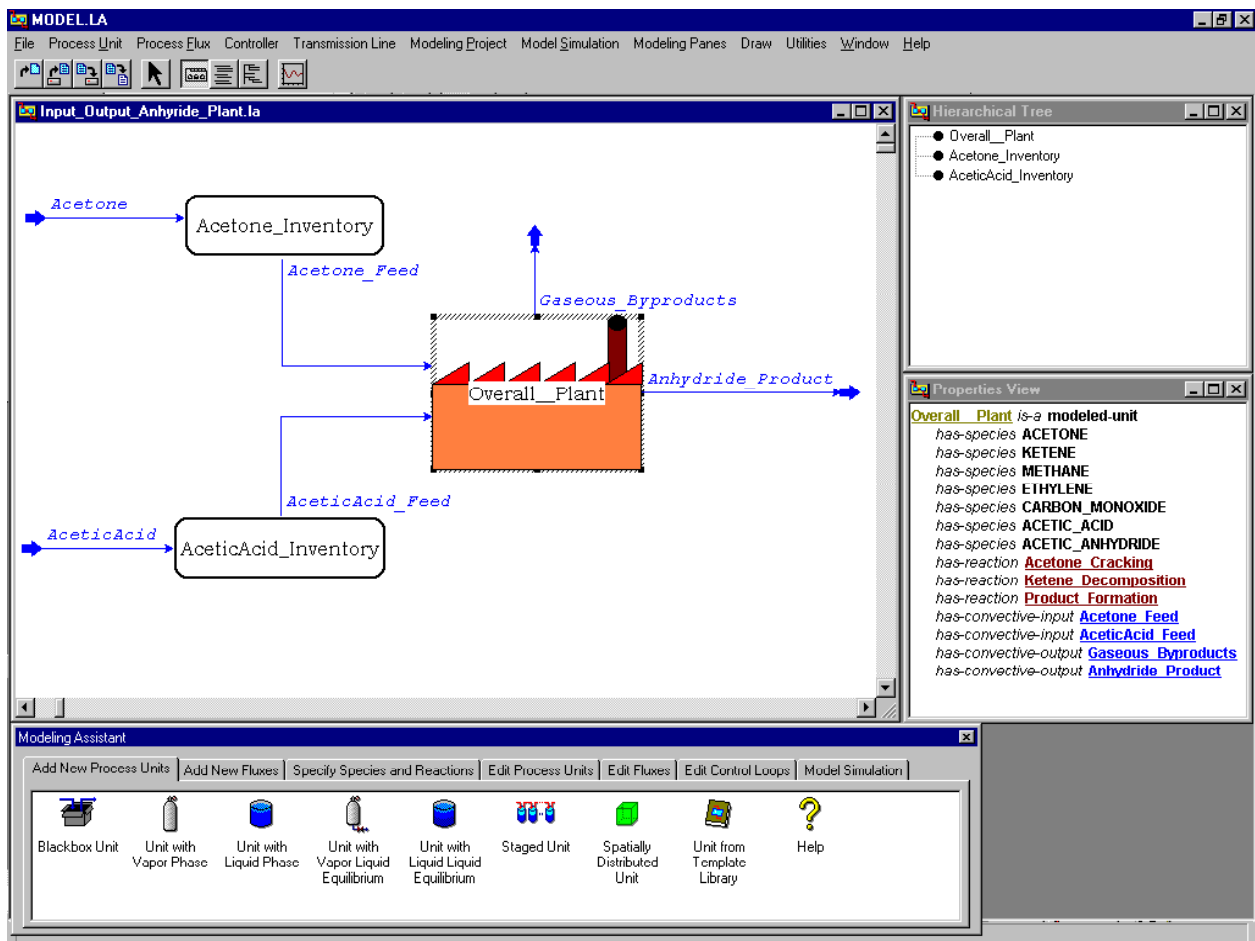


Figure 7-7: Input-Output Level Design for Acetic Anhydride Plant

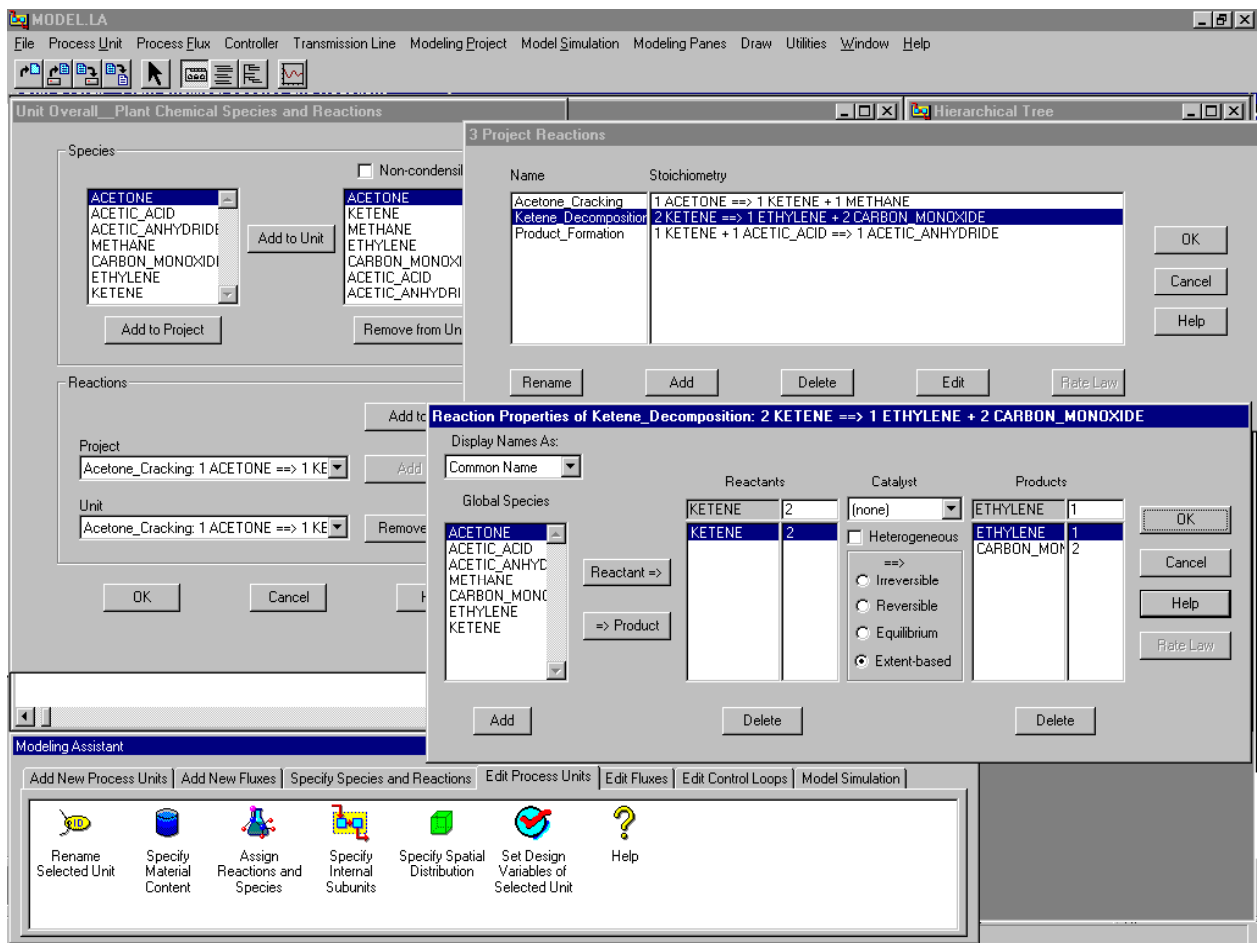


Figure 7-8: Chemical Species and Reactions for Acetic Anhydride Plant Design

Results from the numerical simulation (illustrated in Figure 7-9) of the input-output level view of the plant determine the maximum economic potential of the plant (\$650,000/yr) and a lower bound on the value of ketene yield required (0.83) for the plant to be profitable.

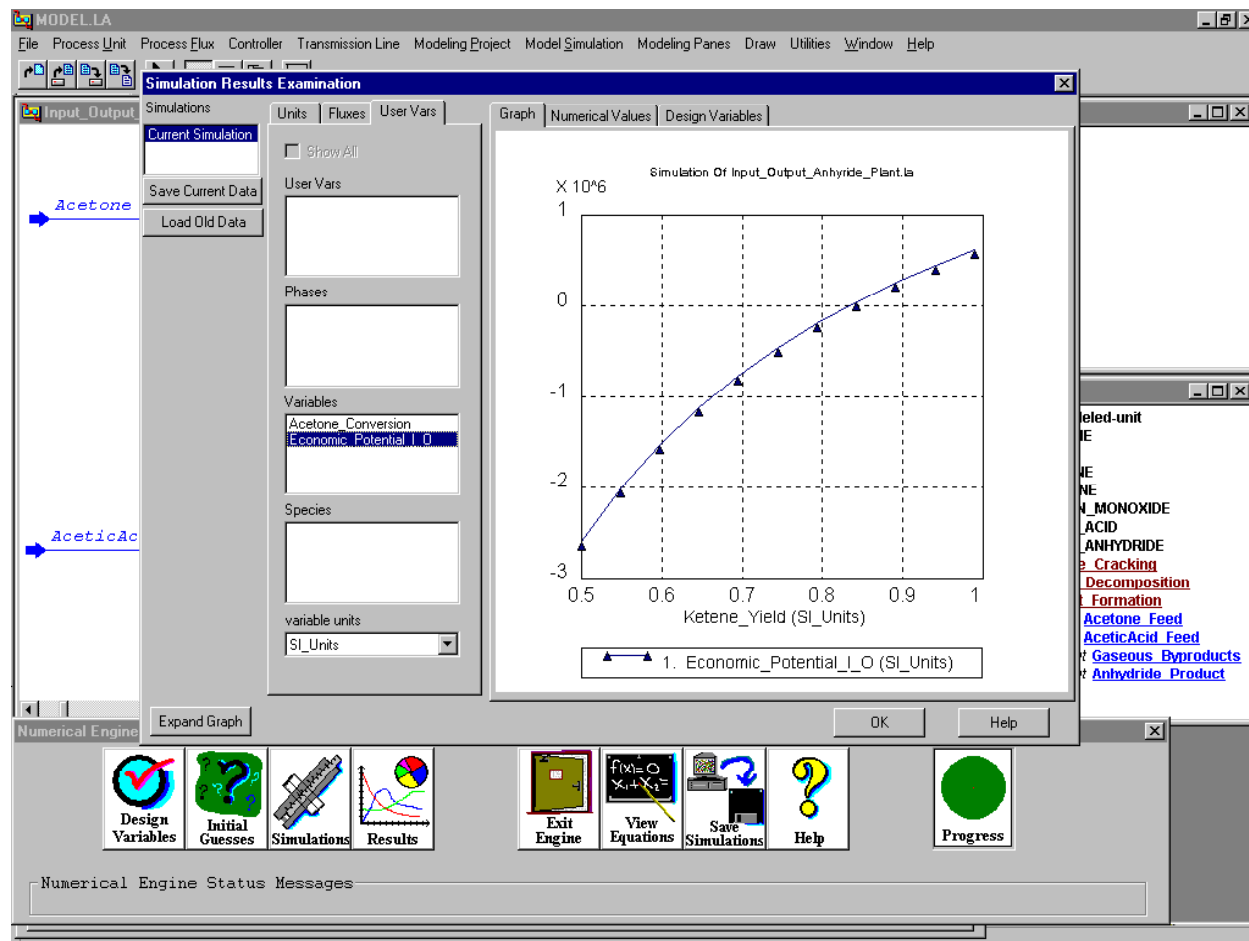


Figure 7-9: Simulation Results for Input-Output Level Design of Acetic Anhydride Plant

Following the hierarchical design approach, the input-output level view of the plant is decomposed into a reaction subsystem and a separation subsystem (illustrated in Figure 7-10). The assumption of full recycle of raw materials is maintained. A gaseous stream (consisting of mostly reaction byproducts) and a liquid stream (consisting of mainly product and unreacted raw materials) from the reaction subsystem to the separation subsystem are assumed. At this level, the concept of per pass conversion of acetone is defined in terms of acetone feed rates to the reaction section, thus allowing recycle rates of acetone to be calculated as a function of ketene yield. This illustrates the tradeoff between high product yield and high recycle requirements. As a result of the ketene yield to acetone conversion design correlation, as ketene yield approaches unity,

acetone conversion approaches zero, requiring a high rate of acetone recycle (e.g., a value of .95 for ketene yield results in acetone conversion of 0.04 and acetone recycle of 450 lb-mol/hr).

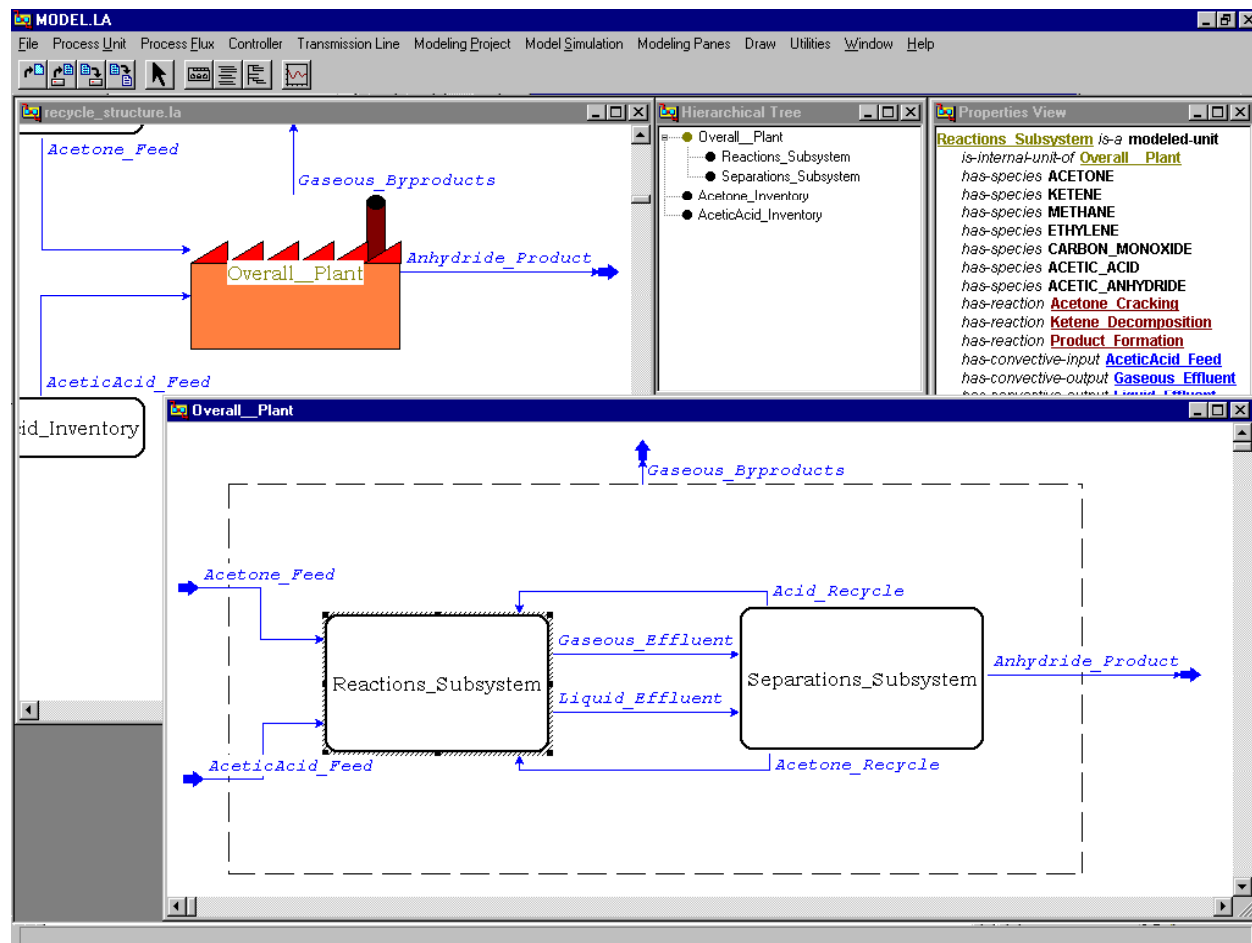


Figure 7-10: Reaction and Separation Section Design for Acetic Anhydride Plant

The design continues with the configuration of reactors in the reaction subsystem (illustrated in Figure 7-11). At this level, energy balances are also introduced. The pure acetone raw material stream is mixed with the acetone recycle stream and fed to an acetone cracking furnace which operates at 700°C. The effluent stream from this reactor (containing unstable ketene) is then rapidly quenched with pure and recycled acetic acid in a two-phase quench reactor. To maintain the quench reactor at 80°C, a liquid stream is withdrawn from the reactor, cooled with water in a heat exchanger, and recirculated back to the reactor. At this level, equipment sizing and cost correlations are introduced for the reactors and heat exchanger as user-defined equations. Also, energy costs for the required furnace duty are included in economic potential calculations.

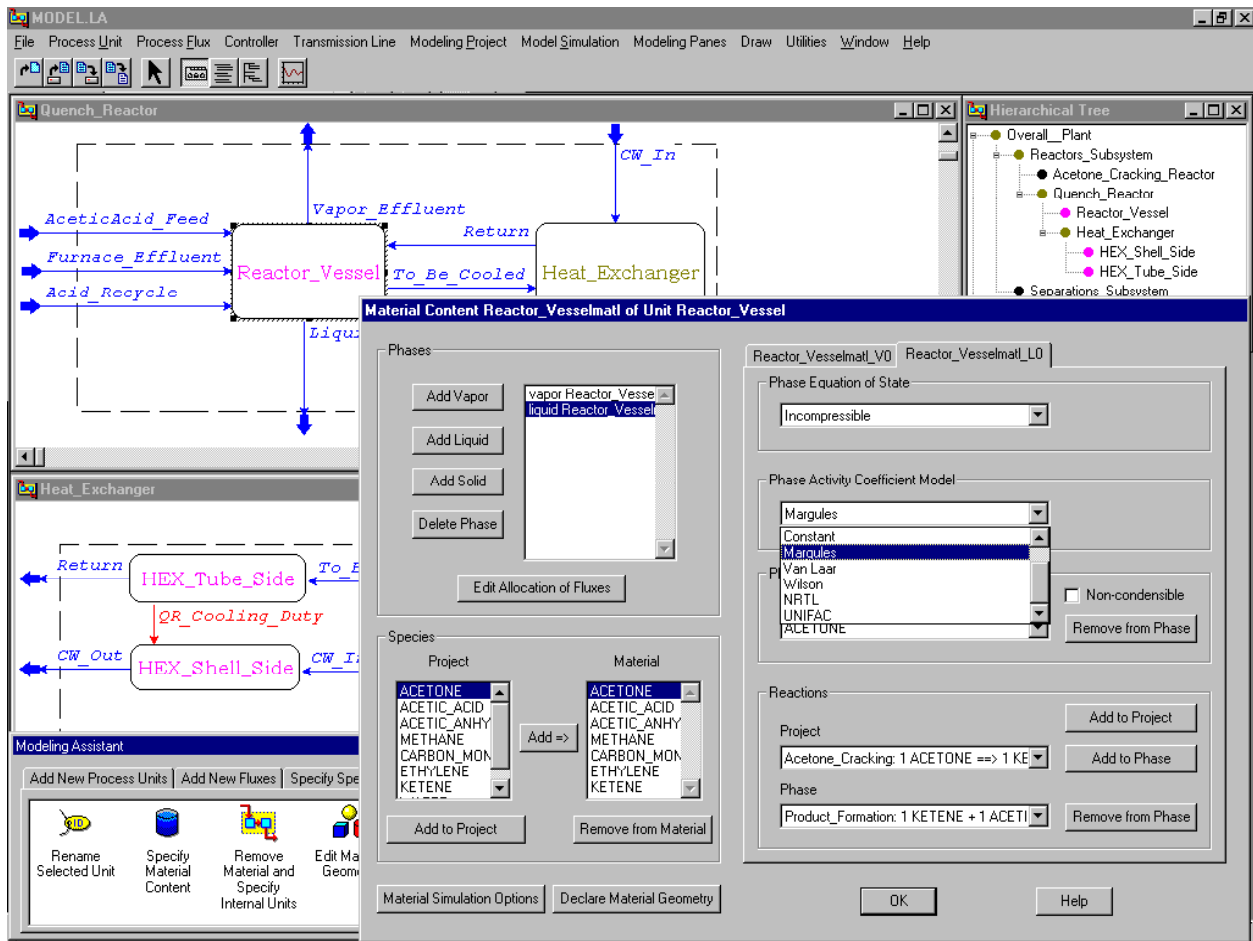


Figure 7-11: Reactor Design for Acetic Anhydride Plant

Since the quench reactor operates at conditions of vapor-liquid equilibrium, two effluent streams are fed to the separations subsystem. Initial design (illustrated in Figure 7-12) of the separations subsystem concentrates on the gaseous stream. To reduce loss of raw materials and product in the gaseous stream, the stream is first cooled to condense most of the valuable components, then separated in a flash. Additionally, a gas absorber is used to recover most of the remaining valuable components in the gaseous stream leaving the flash. An acetic acid stream from the liquid separation subsystem is used as the solvent in the absorber. The gaseous byproducts stream from the absorber leaves the plant, while the liquid solvent stream returns to the liquid separations subsystem.

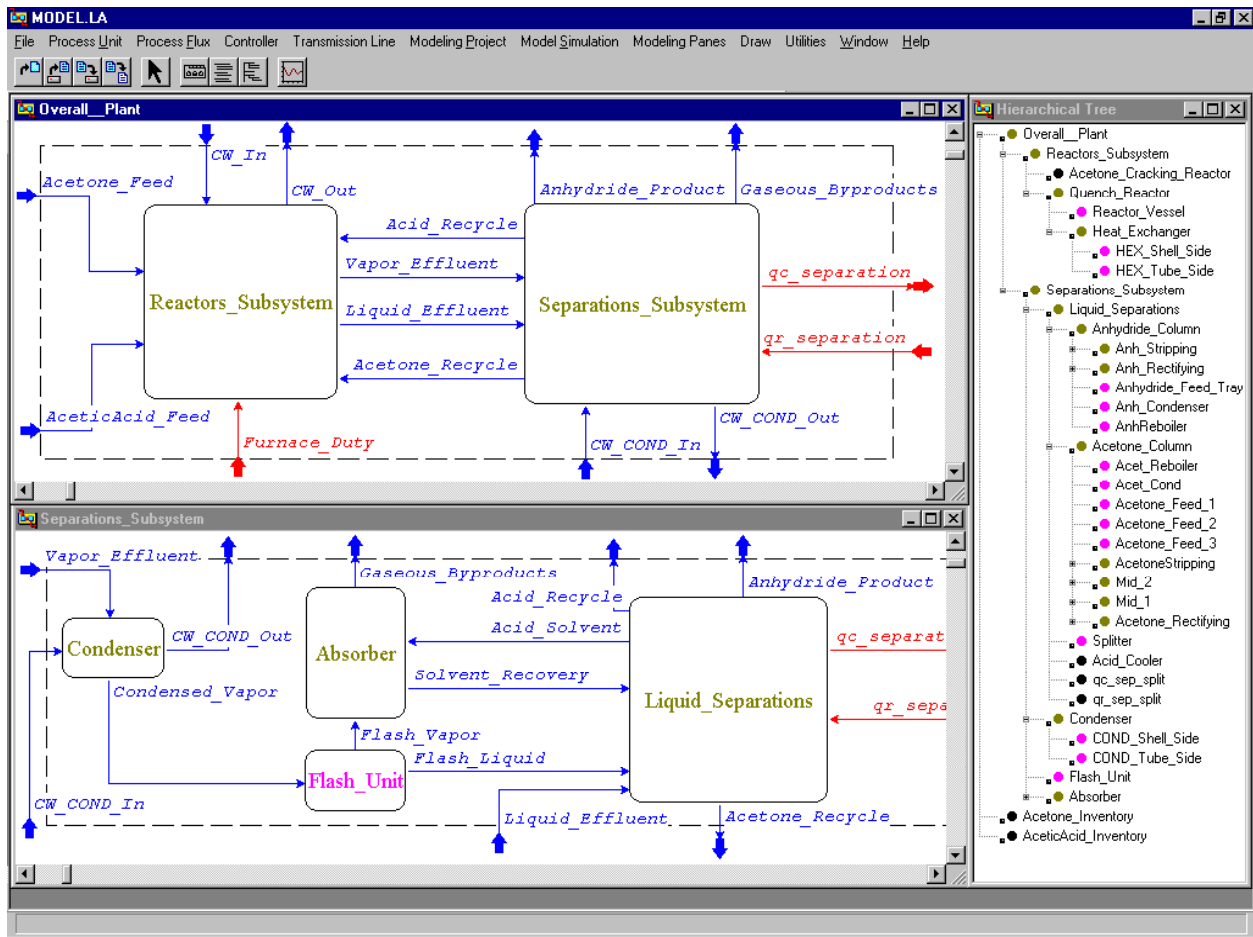


Figure 7-12: Separations Subsystem Design for Acetic Anhydride Plant

The liquid separations subsystem is then designed to separate three liquid streams: the quench reactor liquid effluent stream, the liquid stream condensed from the quench reactor vapor stream, and the used liquid solvent stream. Most of the anhydride product is in the quench reactor liquid effluent stream, so this stream is fed to a distillation column to separate the anhydride product as a bottoms stream. The remaining two streams, along with the acetone/acid overhead stream from the anhydride column are fed to a second distillation column. The overhead stream from this column is mostly pure acetone, while the bottoms stream is mostly pure acetic acid. The acetone stream is recycled to the acetone cracking furnace, while the acetic acid stream is split so that a portion is used as a solvent in the gas absorber and the remainder is recycled to the quench reactor.

The Kremser equation (King, 1980) is introduced as a user-defined equation to estimate the number of stages in the gas absorber (10 stages), and the rectifying (6 stages) and stripping

section (13 stages) of the anhydride column. Since the three streams separated by the Acetone/Acid column have varying concentrations, they are fed to different trays in the column. The Kremser equation is then again used to estimate the number of stages in each section of the column. These calculations result in a 23 stage column, where the liquid from the flash is fed to the fourth stage, the anhydride column overhead is fed to the ninth stage, and the absorber solvent is fed to the fourteenth stage. The absorber and columns are then modeled as explicit VLE stages where each section is modeled as a staged unit.

Sizing and cost correlations for the separation subsystem equipment are added as user-defined equations, and the economic potential of this base-case design is then evaluated through numerical simulation. These results (shown in Figure 7-13) reveal that the design is not profitable at any value for ketene yield.

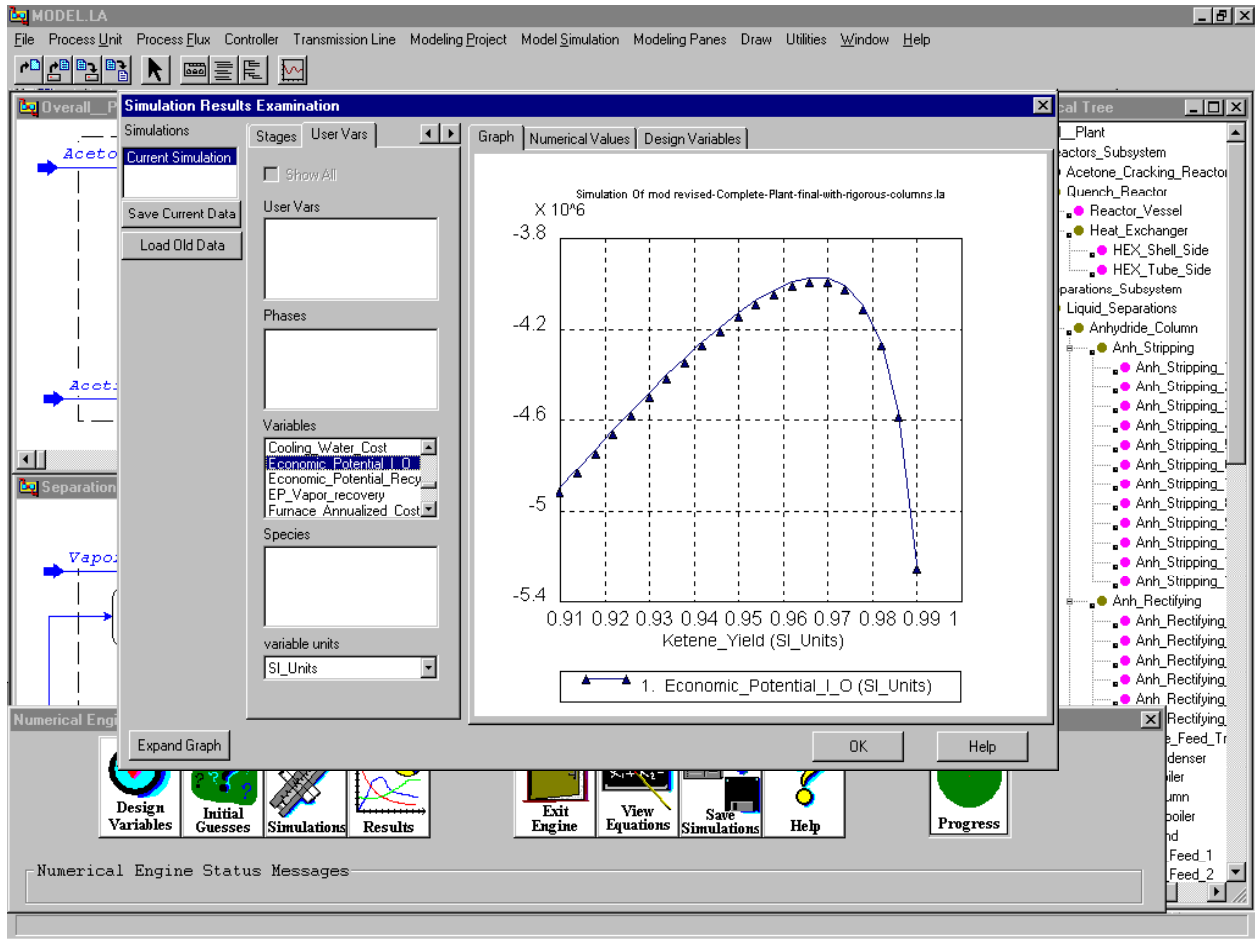


Figure 7-13: Economic Potential for Base Case Design of Acetic Anhydride Plant

7.1.3 Dynamic Distillation Column Example

The following phenomena-based model is based on a SPEEDUP example model file. It is a dynamic model of a distillation column with a side stripper for the separation of benzene, toluene, and o-xylene. The process is illustrated in Figure 7-14.

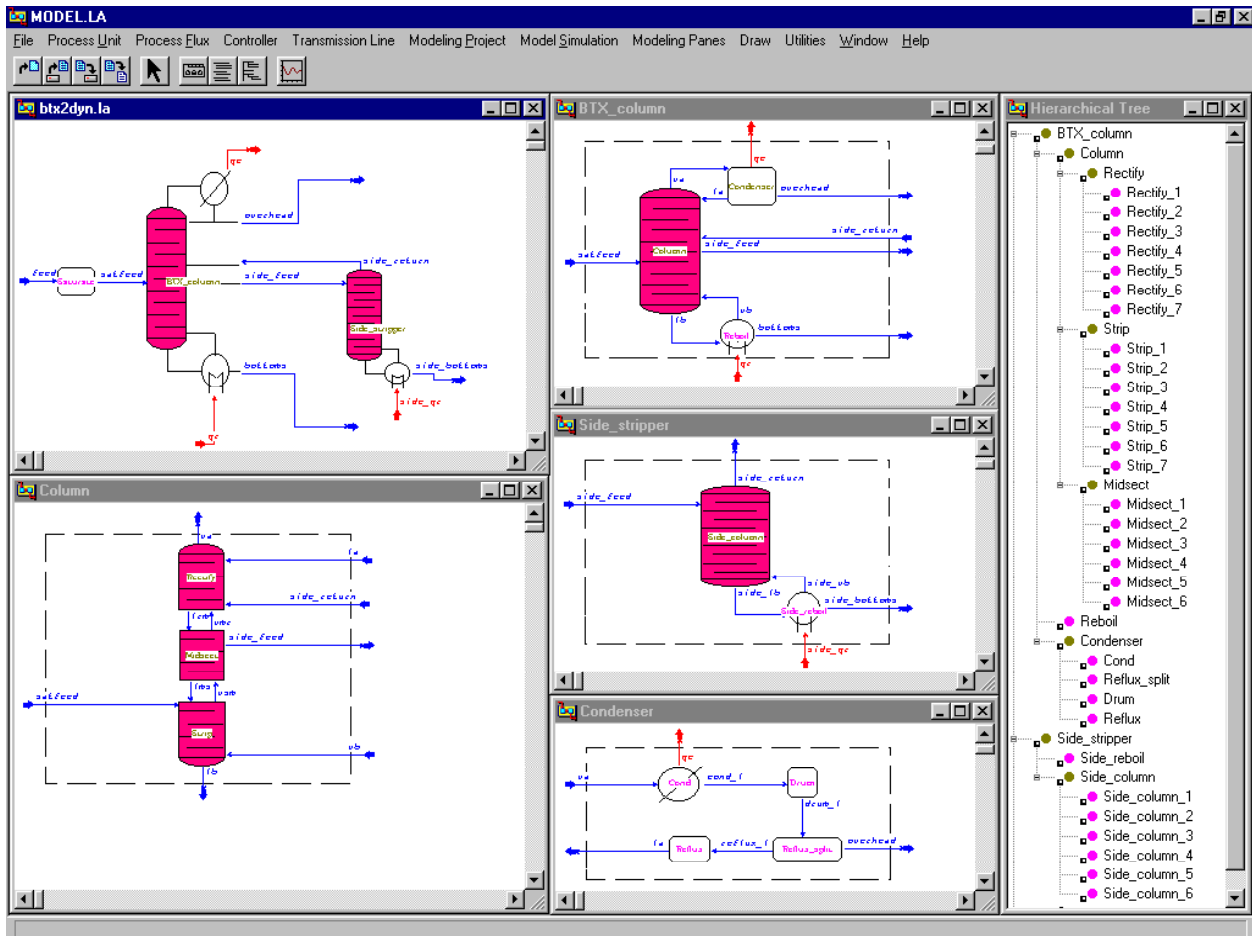


Figure 7-14: BTX Dynamic Distillation

A saturated 0.45/0.45/0.1 mole percent liquid mixture of benzene/o-xylene/toluene is fed to the fourteenth tray of a twenty tray distillation column. A sidestream is withdrawn from the eighth tray of the column and fed to the top of a six tray side stripper. A benzene-rich overhead stream is drawn from the top of the column, while o-xylene-rich and toluene-rich bottoms streams are drawn from the column and side stripper, respectively. Ideal vapor liquid equilibria are assumed for all trays in the column and side stripper. All tray-to-tray vapor flows in the column and side stripper are modeled using a pressure-driven transport mechanism, and all tray-to-tray liquid flows are modeled using Francis Weir overflow transport mechanisms.

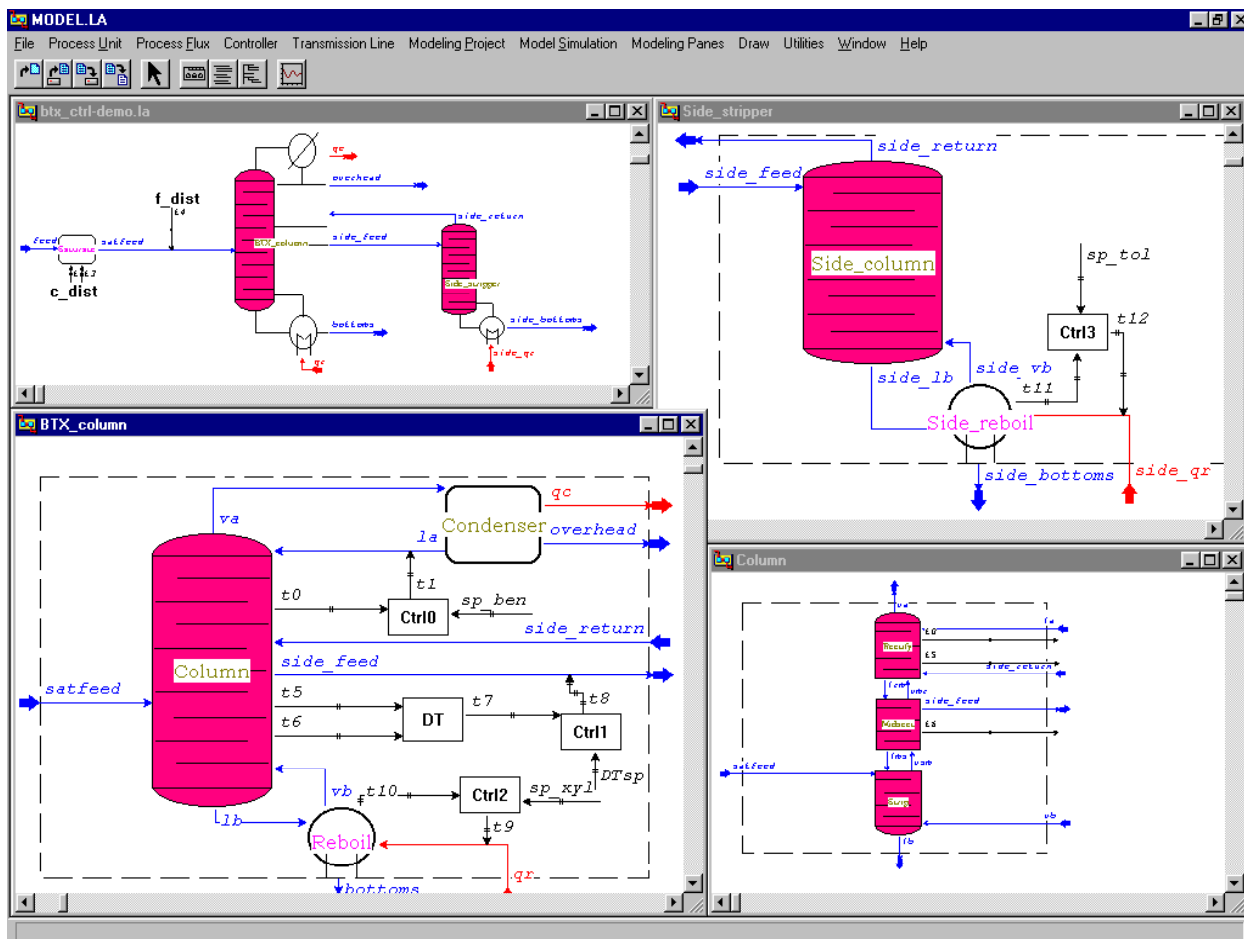


Figure 7-15: PI Control of Dynamic Distillation Column

The process was first modeled at steady state, with molar purity specifications of the benzene, toluene, and o-xylene streams set at 95%, 85%, and 95%, respectively, to determine nominal values for the reflux flow rate, side stripper feed stream flow rate, and energy inputs to both reboilers. Results from the steady-state simulation were then used to initialize a dynamic simulation for a study of the open-loop dynamic response of the process to disturbances in feed concentration and feed flow rate. Four PI controllers were then added to the process, as illustrated in Figure 7-15. These controllers are summarized in Table 7-3.

Table 7-3: PI Controllers of BTX Dynamic Distillation

<i>Controller</i>	<i>Measured Variable</i>	<i>Setpoint</i>	<i>Manipulated Variable</i>
Ctrl0	vapor mole fraction of benzene in tray 20	$x_{benzene}=0.95$	reflux flow rate
Ctrl1	temperature differential between trays 13 and 14	$T_{tray\ 13} - T_{tray\ 14}=10^{\circ}C$	flow rate of side stripper feed stream
Ctrl2	mole fraction of o-xylene bottoms stream	$x_{o-xylene}=0.95$	energy input to column reboiler
Ctrl3	mole fraction of toluene bottoms stream	$x_{toluene}=0.85$	energy input to side stripper reboiler

The resulting closed-loop response of the process to a disturbance in the feed flow rate is illustrated in Figure 7-16.

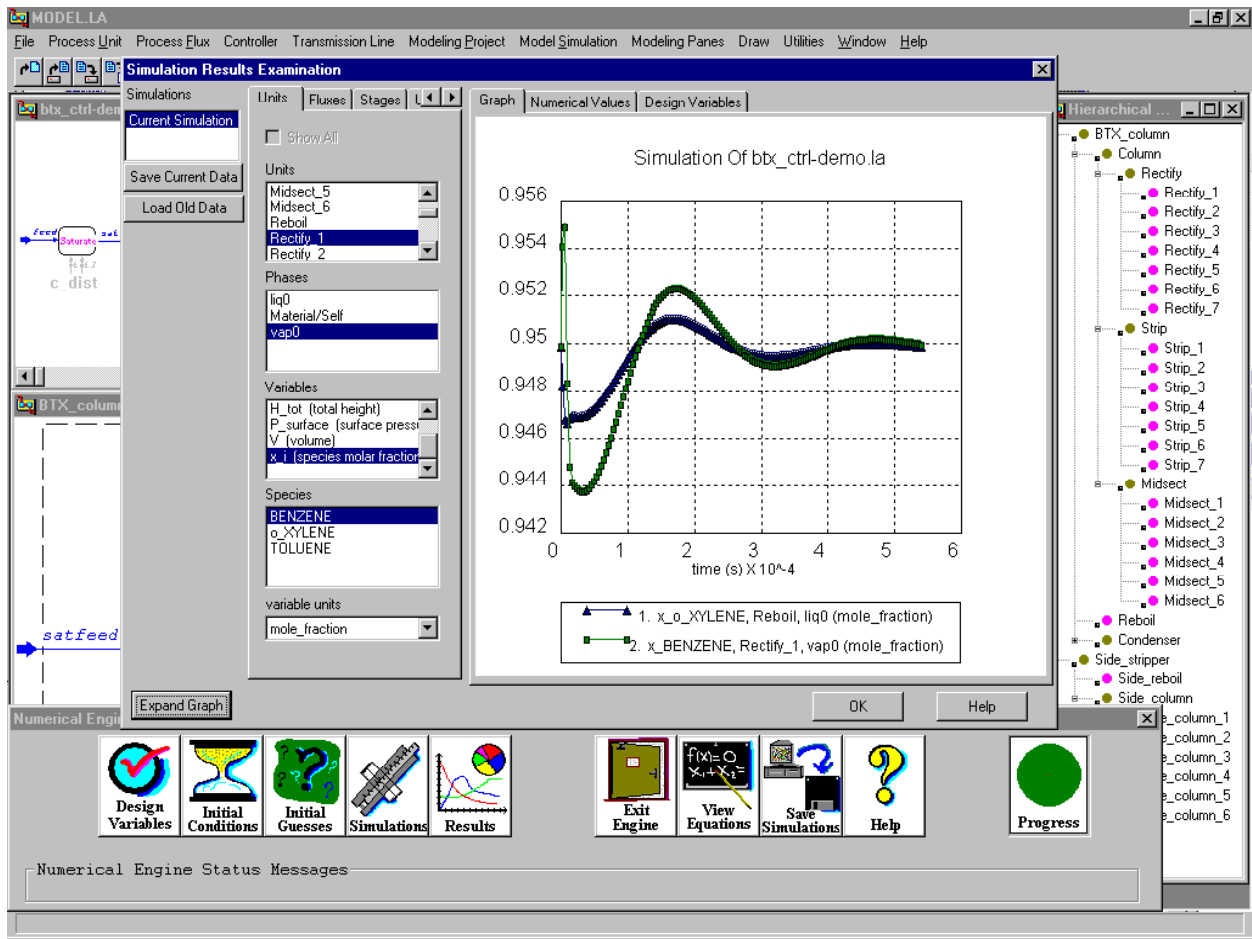


Figure 7-16: Closed Loop Dynamic Response of BTX Distillation Column

7.1.4 1-D Spatially Distributed Reaction and Separation Processes

The following phenomena-based model is based on an example by Heydweiller et al (1977). The model is of a process with three unit operations, a mixer, a tubular reactor, and a countercurrent absorption column. A gPROMS model of this process also appears in (Oh, 1995).

Specifications for the reaction of interest are summarized in Table 7-4.

Table 7-4: Reaction Data for 1-D Spatially Distributed Reaction and Separation Process

1. Stoichiometry:	$A + B \rightarrow 2 C$
2. Rate Law:	$r_A = k \cdot c_A c_B$
3. Gas phase	
4. Isothermal	

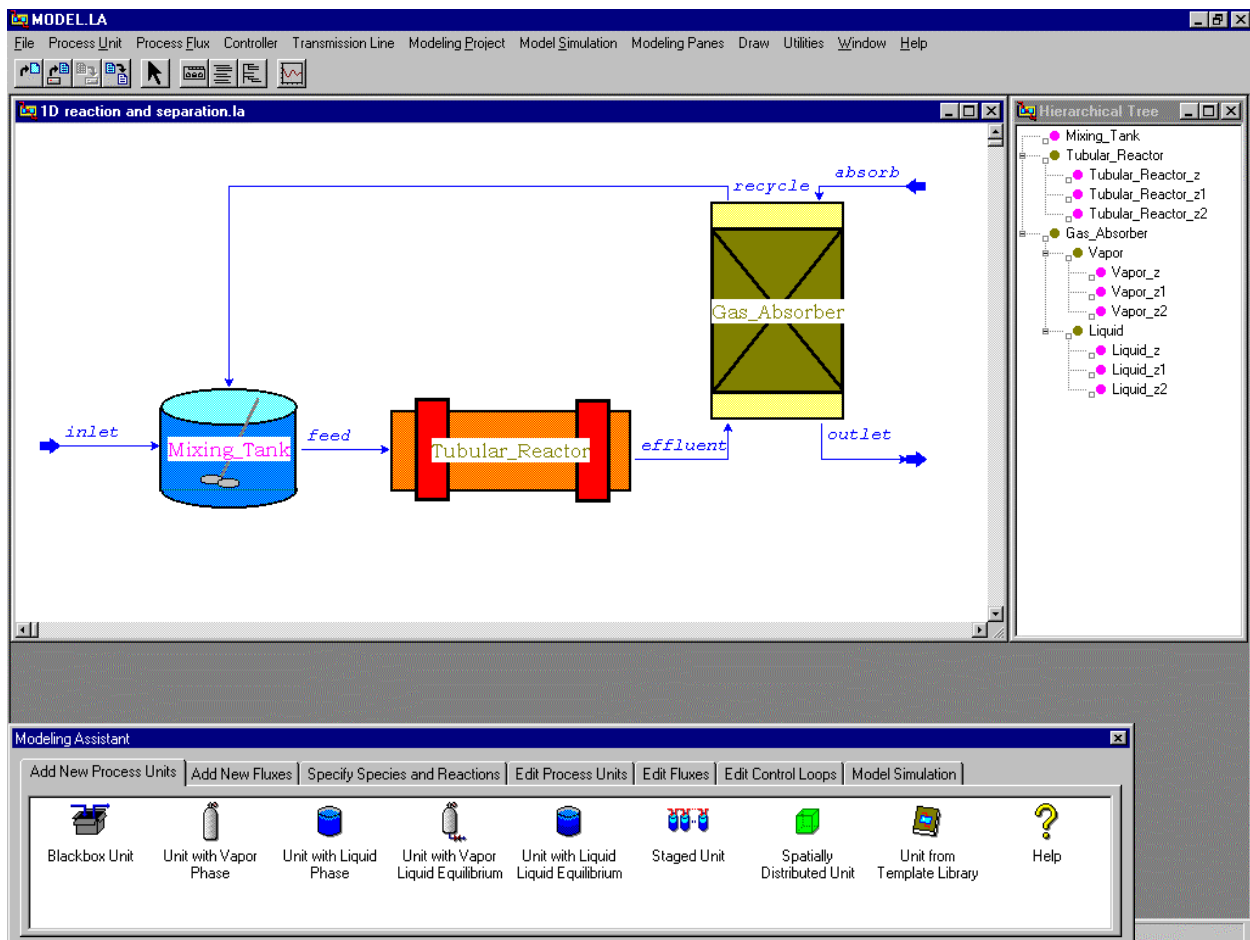


Figure 7-17: 1-D Spatially Distributed Reaction and Separation Process

The structure of the process is illustrated in Figure 7-17. Reactants are mixed with a recycle stream and fed to a tubular reactor. The reactor effluent enters the bottom of a countercurrent absorption column where C is partially absorbed into a liquid phase product stream. The remaining vapor is recycled.

The structure of the reactor and two-phase absorption column are illustrated in Figure 7-18. The reactor, and gas and liquid phases of the absorber are each modeled as 1-D spatially distributed systems. The reactor has convective and Fickian diffusive flux of all species along the distributed z-dimension. The vapor and liquid phases of the absorber have convective flow along their respective distributed z-dimensions, and diffusion of C occurs from the gas to the liquid phase (with the rate of diffusion proportional to the deviation from equilibrium).

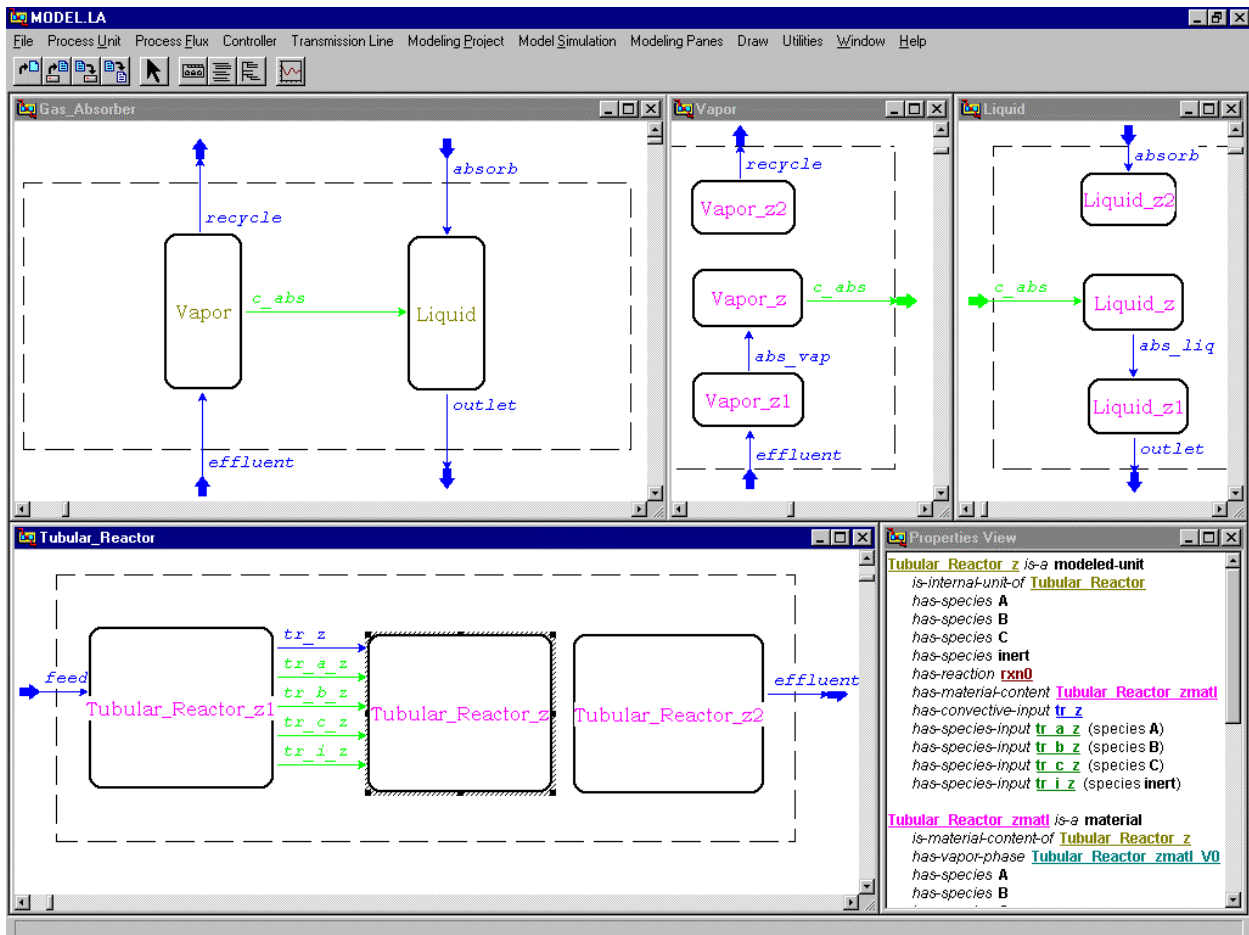


Figure 7-18: Structure of 1-D Spatially Distributed Tubular Reactor and Gas Absorption Column

The model equations derived by MODEL.LA from this phenomena-based model description include 1st-order partial differential, differential and algebraic equations. These equations are submitted symbolically to gPROMS and solved numerically using the partial differential equations modeling capabilities of the solver. The solution methods specified for solution of the resulting partial differential equations are summarized in Table 7-7:

Table 7-5: Solution Methods for 1-D Spatially Distributed Reactor and Absorption Column

1. Tubular Reactor	
a. Domain:	0 to 1 m
b. Elements:	10
c. Order of Approximation:	2
d. Solution Method:	Centered Finite Difference
2. Absorption Column Gas Phase	
a. Domain:	0 to 1 m
b. Elements:	10
c. Order of Approximation:	1
d. Solution Method:	Backward Finite Difference
3. Absorption Column Gas Phase	
a. Domain:	0 to 1 m
b. Elements:	10
c. Order of Approximation:	2
d. Solution Method:	Centered Finite Difference

Once the model equations are solved, the discretized results for spatially distributed variables are read by MODEL.LA, and OLE automation is used to load the simulation results into Microsoft Excel and generate surface plots of selected variables. Figure 7-19 displays results for the rate of reaction in the reactor, and the concentration of C in both the vapor and liquid phases of the absorption column.

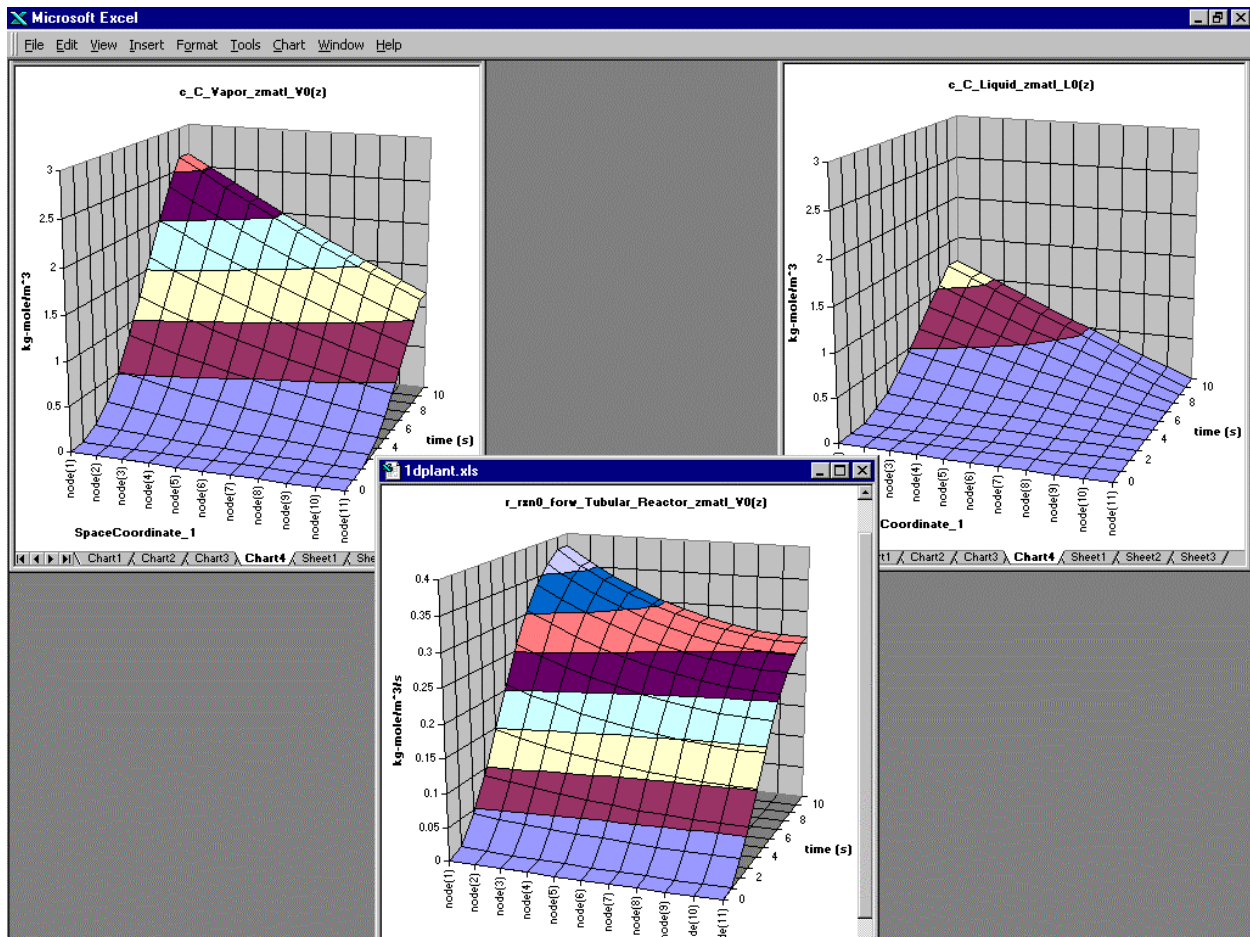


Figure 7-19: 1-D Spatially Distributed Reactor and Absorption Column Results

7.1.5 2-D Tubular Reactor

The final phenomena-based modeling example is a dynamic model of a 2-D spatially distributed tubular reactor with a cooling jacket. It is based on an example by Froment and Bischoff (1990), modified to introduce radial as well as axial distribution. A gPROMS model of this reactor appears in (Oh, 1995).

Specifications for the reaction of interest are summarized in Table 7-6. The structure of the reactor is illustrated in Figure 7-20. The reactor is declared to be cylindrical with radial and axial distribution. Axially, there is convective flux, Fickian diffusive flux of o-xylene and oxygen, and Fourier conductive flux of energy. Radially, there is also diffusion of o-xylene and oxygen and energy conduction. At the outer radial boundary, there is energy transport, modeled by a surface convection mechanism, to a lumped cooling jacket.

Table 7-6: Reaction Data for 2-D Spatially Distributed Tubular Reactor Example

1. Stoichiometry:	$\text{o-xylene} + 3 \text{ oxygen} \rightarrow \text{phthalic anhydride} + 3 \text{ water}$
2. Rate Law:	$r_{\text{o-xylene}} = A \cdot \exp\left(\frac{-E}{RT}\right) P_{\text{o-xylene}} P_{\text{oxygen}}$
3. Gas phase	

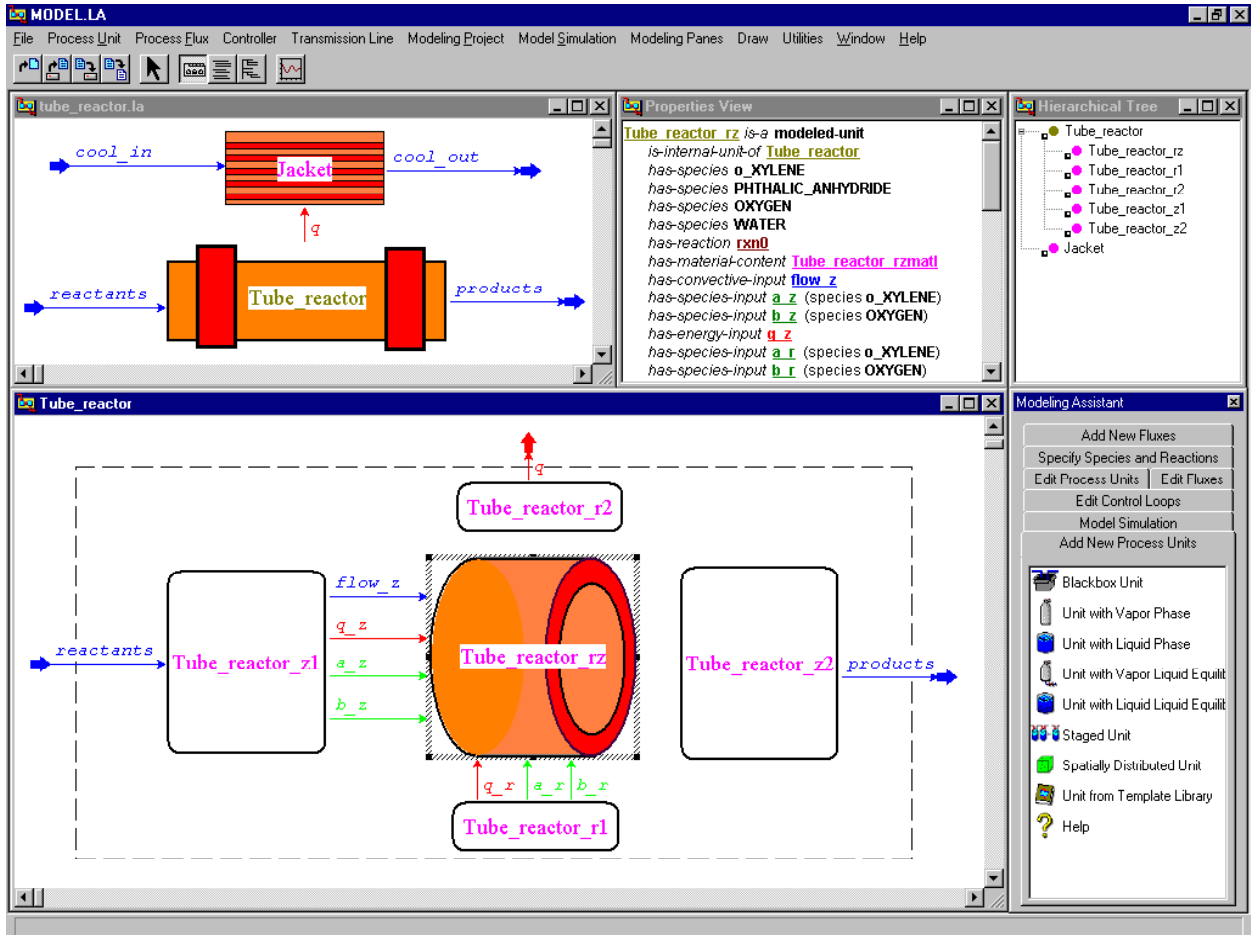


Figure 7-20: 2-D Spatially Distributed Tubular Reactor Example

The model equations derived by MODEL.LA, including integral, 2nd-order partial differential, differential and algebraic equations, are given in Appendix E. These equations are formulated as a gPROMS input file and solved using the solution methods summarized in Table 7-7 for the resulting partial differential equations.

Table 7-7: Solution Methods for 2-D Spatially Distributed Tubular Reactor Example

1. Radial (r) Dimension:	
a. Domain:	0 to 0.0127 m
b. Elements:	5
c. Order of Approximation:	2
d. Solution Method:	Centered Finite Difference
2. Axial (z) Dimension:	
a. Domain:	0 to 3 m
b. Elements:	14
c. Order of Approximation:	1
d. Solution Method:	Backward Finite Difference

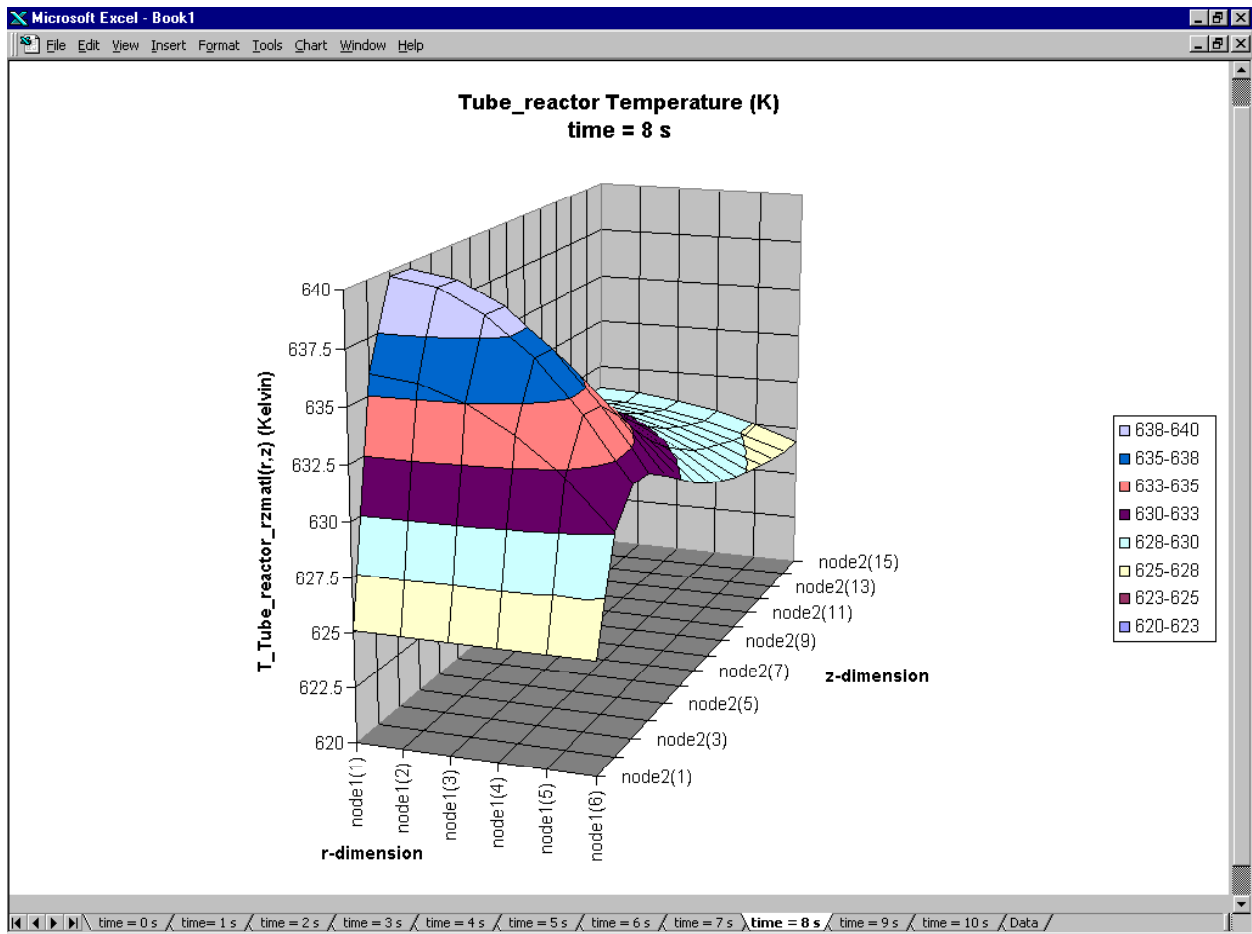


Figure 7-21: 2-D Spatially Distributed Tubular Reactor Simulation Results

To facilitate analysis of the behavior of 2-D spatially distributed processes, MODEL.LA uses OLE automation to load the simulation results into Microsoft Excel, generate surface plots, and animate these plots. An excerpt from the results for the 2-D spatially distributed tubular reactor, which exhibits a hot spot near the reactor inlet at the radial center, is illustrated in Figure 7-21.

7.2 Summary of Model Examples

The examples in this chapter illustrate the use of the MODEL.LA Modeling Environment in a variety of modeling contexts. These examples and the form of the resulting mathematical models are summarized in Table 7-8.

Table 7-8: Summary of Phenomena-Based Modeling Examples

<i><u>Example</u></i>	<i><u>Steady-State or Dynamic Process</u></i>	<i><u>Type of Mathematical Model</u></i>	<i><u>Number of Equations</u></i>	<i><u>Comments</u></i>
HDA Plant	Steady-State	Algebraic	3293	Hierarchical design
Acetic Anhydride Plant	Steady-State	Algebraic	3894	Hierarchical design with economic analysis
BTX Distillation	Dynamic	DAE	2266	Open loop and closed loop control structures
1-D Reaction and Separation	Dynamic	PDAE	138	Isothermal process
2-D Tubular Reactor	Dynamic	IPDAE	93	Non-isothermal process

Chapter 8

Conclusions and Recommendations

This final chapter summarizes the primary research contributions of this work and the potential impact envisioned it may have on chemical process modeling in engineering practice and in undergraduate education. It concludes with recommendations for future research.

8.1 Research Contributions

The main contributions of this research include:

1. The development of a high-level phenomena-based modeling language for the representation of chemical process models in terms of interacting physicochemical phenomena.
2. The description of modeling logic, which allows systematization of the model development process through explicit representation of modeling tasks as operators that act on a language-based model description.
3. The integration of the phenomena-based modeling language and logic into a computer-aided modeling environment that enables rapid, reliable, and documented chemical process model development from first principles.

Each of these aspects will now be discussed.

8.1.1 Phenomena-Based Modeling Language

The phenomena-based modeling language of MODEL.LA provides a high-level language for representing chemical processes in terms of interacting physicochemical phenomena. This language is designed to enable chemical engineers and computers to communicate in a language based on the principles of chemical engineering science. The high-level nature of this language

can allow all chemical engineers, not just modeling experts, to develop and use chemical process models formulated from first principles. Compared to mathematical equation-based models, the resulting models are much easier to construct, edit, debug, analyze, reuse, and understand. Furthermore, the phenomena-based model representation preserves modeling knowledge, by explicitly retaining the assumptions behind a process model.

8.1.2 Formalized Modeling Logic

The MODEL.LA modeling logic operators provide a basis for systematizing the process of model development by explicitly characterizing modeling tasks that are currently carried out in an informal and implicit manner. These operators allow the computer to understand the procedural and declarative aspects of the modeling activity. This enables it to provide assistance for analyzing and constructing phenomena-based models, to detect inconsistencies and incompleteness in the phenomena-based model description, and to derive and explain the resulting mathematical model equations. In addition, by recording operators activated during model development, explicit documentation of the modeling activity that produces a process model can be maintained.

8.1.3 Computer-Aided Modeling Environment

The motivation behind the development of the phenomena-based language and logic of MODEL.LA is to enhance the modeling capability and productivity of any chemical engineer. However, the concepts of phenomena-based language and logic expressed on paper alone would essentially limit the impact of this work to an academic exercise. The MODEL.LA modeling environment embodies these ideas of language and logic, and provides a system that enables phenomena-based modeling of dynamic systems of arbitrary structure and spatial distribution, hierarchical levels of detail, and multicontext depictions. Components of the MODEL.LA environment provide automated mathematical model derivation, incorporation of thermodynamic and physical property data, integration of control structures, operational task scheduling, and external models, and assistance for analysis, specification, and solution of the resulting mathematical model. The features of the MODEL.LA environment enable evaluation of the phenomena-based modeling methodology through application to wide variety of modeling examples and case studies. Such examples have highlighted the enhanced productivity, reliability,

and maintainability of models developed in this environment.

8.2 Potential Impact on Modeling in Engineering Practice

The industrial use of the MODEL.LA modeling environment was subjected to preliminary evaluation at modeling workshops which took place at the Mitsubishi Chemical Corporation in Kurashiki, Japan and the Dow Chemical Corporation in Midland, Michigan. At each location, 15-20 process engineers assessed the use of the MODEL.LA environment in application to dynamic process simulation, spatially distributed system modeling, and hierarchical process design.

These experiments revealed that the high-level phenomena-based modeling approach of MODEL.LA can have a unique impact on chemical process modeling by:

1. Reducing the time required for equation-based process model development by an order of magnitude,
2. Enabling process models to be readily used and reused in different contexts (e.g., process design, steady state or dynamic optimization, training, controller design),
3. Enforcing the consistency and completeness of assumptions that characterize complex process models,
4. Supporting the multiple resolution modeling and analysis of hierarchical and spatially distributed systems, and
5. Retaining the knowledge and explicit assumptions behind the development of a process model.

Specifically with regard to process design, MODEL.LA can have a unique impact by:

1. Accelerating process model development and thus increasing the number of alternatives that can be considered,
2. Allowing experts in varying backgrounds to readily contribute to a design in a collaborative manner by raising the level of model development from the equation or procedural level to the knowledge level,
3. Providing complete flexibility in process specification, since models are not limited to an existing library, and
4. Facilitating evolutionary process development by allowing addition of detail in a hierarchical manner.

In summary, the MODEL.LA modeling environment can enhance process modeling in engineering practice by not only guiding and expediting the process of model development, but also by transforming the product of modeling from a procedural or mathematical equation-based representation to an chemical engineering phenomena-based representation.

8.3 Potential Impact on Undergraduate Chemical Engineering Education

The pedagogical approach of chemical engineering education was established many decades ago by the concept of unit operations. Unit operations were identified as common types of equipment (e.g., distillation column) that were then characterized by generalized sets of equations or other methods of analysis (e.g., McCabe-Thiele diagrams for distillation columns). These classic set pieces of instruction were developed to concisely introduce methods of analysis to students and engineers. In the 1960s, an emphasis on more fundamental concepts (i.e., transport phenomena) introduced more science and mathematics into the chemical engineering curriculum, yet also within the context of what have become classic set pieces of analysis (e.g., heated fin, Navier-Stokes equations, etc.). “To a large extent,” writes Cussler (1999), “[chemical engineering curricula] reflect the scheme first suggested in 1917.” As a result of this tradition that often limits instruction to the analysis of idealized situations, the education of many contemporary students is left incomplete.

The Bloom Taxonomy of Educational Objectives (Bloom, 1956) identifies six ascending levels of understanding: (i) *translation* (i.e., memorization), (ii) *interpretation* (i.e., paraphrased repetition), (iii) *application* (i.e., “analysis” in an engineering context), (iv) *analysis* (i.e., “modeling” in an engineering context), (v) *synthesis* (i.e., creative design), and (vi) *evaluation* (i.e., critical appraisal). Unfortunately, most of the current curricula focuses only on the first three levels, and does not adequately nurture higher-level understanding in students. Creative exercises, coupled with computer-based material, are one way to allow students to develop these higher-level thinking skills (Montgomery and Felder, 1996).

The MODEL.LA modeling environment allows students to develop models at the high-level of elementary physical and chemical phenomena that they assume to occur in a chemical process. This enables students to freely express their assumptions of what constitutes the physical and chemical makeup of a model, while being guided through a structured process of model

development.

8.3.1 Structuring of Modeling Activities

While students are free to express their own notions of the assumptions behind a chemical process model, the framework of the MODEL.LA modeling environment enforces an explicit, yet natural, structure on the process of model development. Key tasks that a student must tackle during this process are summarized below:

1. Decide what are the appropriate control volumes (i.e., systems) for a process,
2. Declare how these systems interact through transport of mass and energy,
3. Specify and characterize chemical species, reactions, and materials present in the process,
4. Refine and specify the internal content of the control volumes,
5. Characterize the boundary interactions mechanistically,
6. Check the model for consistency and prepare it for solution, and
7. Specify values of known parameters, solve the model, and analyze the results.

These tasks simply represent good modeling practice, whether a student is using MODEL.LA, another computer-aided modeling tool, or paper and pencil. However, only in MODEL.LA are all of these steps always explicit. This rigor helps to enforce a sound modeling methodology in students, and provides a framework for subsequent model development with or without computer-aided assistance.

8.3.2 Classroom Deployment of MODEL.LA

MODEL.LA was deployed in the senior-year Integrated Chemical Engineering (10.490) course at MIT during the Fall 1998 semester. The educational context of this course focused on hierarchical process synthesis. These experiments were designed to investigate the pedagogical use of MODEL.LA in the classroom. Using questionnaires (with 34 responses from 38 participating students), the instructional experiments at MIT revealed how students benefited the most from use of MODEL.LA:

1. The students recognized in MODEL.LA the basic principles of chemical engineering science and felt confident invoking them during the modeling of processing systems (85% of the respondents indicated good to excellent

recognition and usage of the principles).

2. In contrast to traditional flowsheet design, 100% of respondents preferred the evolutionary, hierarchical modeling approach of MODEL.LA, which enabled group members to distribute the work load, while maintaining the consistency and integrity of the overall design.
3. A significant majority (70%) of respondents indicated that MODEL.LA allowed them to effectively shift the focus of attention from the algebraic manipulation of modeling equations to the engineering problem at hand (i.e. how to synthesize a chemical processing scheme).
4. Almost all of the respondents (90%) found the graphic user interfaces of MODEL.LA very natural to their modeling tasks, and far more “relevant”, “intuitive”, “effective”, and “powerful” than other computer-aided modeling system they had used (including programming languages, spreadsheets, flowsheet simulators, and equation-based modeling tools).
5. Students also suggested that MODEL.LA be incorporated into several other undergraduate core courses in chemical engineering, including, (i) Introductory Course in Chemical Engineering, (ii) Thermodynamics, (iii) Separation Processes, (iv) Kinetics and Reaction Engineering, (v) Chemical Engineering Laboratory, and (vi) Process Design.

MODEL.LA was also deployed at the University of California at Berkeley using small-size groups of sophomore students. The educational context in these groups focused on learning the art of modeling. In written evaluations, these students commented that MODEL.LA was successful in (i) focusing their learning on the concepts and phenomena behind a problem, (ii) providing a physical feel for a model that is missing from equation writing, and (iii) deepening their understanding of phenomena and behavior with rapidly produced numerical feedback.

8.3.3 Pedagogical Use of MODEL.LA

By taking responsibility for mathematical model derivation, MODEL.LA can allow students to freely express their notions of what assumptions characterize an adequate process model, without trepidation over subsequent mathematical manipulations. Of course, students must master

mathematical model derivation skills. However, students must not dissociate description of the underlying physical and chemical phenomena from the problem-solving activity.

Based on the preliminary experiments described above, the following proposes the appropriate pedagogical use of MODEL.LA in undergraduate chemical engineering education:

1. For novice students, MODEL.LA acts as a virtual laboratory, where students use predefined models to investigate qualitative cause-and-effect interactions between phenomena, process parameters and observed behavior (e.g., adding heat to a vapor-liquid equilibrium system).
2. For beginning students, who are comfortable writing balance equations for blackbox systems, MODEL.LA helps students extend these models to more complex systems by familiarizing them with the functional form, limitations, and parameters of the equations at every stage of assumption-making. This approach seeks to instill a disciplined approach to modeling so that students are not immediately overwhelmed when faced with a new problem-solving context.
3. For intermediate students, MODEL.LA provides a toolbox where the seemingly disjoint concepts taught in various core chemical engineering courses can be integrated, applied in the context of open-ended problem-solving, and truly understood through such applications.
4. For advanced students, MODEL.LA allows them to concentrate on matters of process synthesis, where they investigate operating alternatives and generate novel process structures using the building blocks of elementary physical and chemical phenomena, not the hard-wired unit operation models of commercial simulators. MODEL.LA allows these students to concentrate on the creative “*what-if*” aspects of synthesis, without being bogged down or intimidated by the mathematical manipulations required to investigate new ideas.

8.3.4 Unique Impact on Undergraduate Education

It is envisioned that appropriate pedagogical use of MODEL.LA can result in a unique impact on the use of process modeling in chemical engineering undergraduate education. Several such aspects are summarized below:

1. *An advanced background in mathematics for the student will not be a prerequisite.* Many students are overwhelmed by modeling because they are dismayed or distracted by the intricacies of the mathematical calculations involved. The declarative phenomena-based modeling language approach can alleviate this by allowing students to concentrate on the chemical engineering concepts and principles employed during the process of modeling, disjoint of any complex mathematical activity. Obviously, mathematical skills are indeed a critical educational requirement for engineering students. However, beginning students who do not yet possess these skills should not be restricted from exercises that nurture creative and critical thinking.
2. *The learning rate of the student can be increased.* Students are currently taught the set pieces of the foundation material early in the chemical engineering curriculum, but then struggle to apply these concepts later in senior-year laboratory and design projects. While it is through this type of struggle that the student begins to truly understand and appreciate the basic principles, this learning should take place far earlier in the curriculum. The MODEL.LA modeling environment can allow the student to apply these concepts within the core courses to non-idealized situations where decisions beyond “picking the right equation” must be made. Students may develop their understanding by experimenting with their ideas and assumptions, and then be given immediate feedback so they may evaluate the impact of their decisions. This will enable students to gain a deeper understanding of basic principles when they are first taught by making the link between concepts and open-ended engineering problem solving.
3. *The scope of engineering problems the students investigate can be broadened.* The culmination of most chemical engineering curricula is one or more senior-year design projects which, ideally, should force students to integrate all the knowledge they have learned in the core courses. Typically, the task is to design and optimize a system of interrelated processes and operations in order to meet certain objectives. However, in many cases the challenge to the student becomes not integrating chemical engineering knowledge, but learning how to use some unit

operation-based flowsheet simulator. This experience may benefit certain students who will use such simulators later in their professional careers. However, a flowsheet simulator is not an appropriate pedagogical tool for this situation because by definition its focus is on the use of predefined unit operation models. A phenomena-based tool will enable students to integrate all concepts they have learned in application to a design problem. While the student may still implement such units as “heat exchangers”, “packed towers” or “reactors”, they will be defined through a physicochemical phenomena-based approach, where all assumptions become explicit and understood, rather than just selected as “black boxes”. With the assistance of the high-level environment such as MODEL.LA, students may even feel inspired to play with the design of novel processing units.

8.4 Directions for Future Research

Use of the MODEL.LA modeling environment in undergraduate education and industrial practice reinforced several of the anticipated benefits of the phenomena-based modeling approach. In addition, these experiences also revealed several opportunities for future directions of research and development.

8.4.1 Phenomena-Based Modeling Language Extensions

The phenomena-based modeling language of MODEL.LA reflects the focus of the traditional chemical engineering curricula, in that it is best suited for the modeling of processes that involve traditional petrochemical materials. Likewise, few simulation tools currently exist that are well-suited for the modeling of specialty chemicals. To address this need, the modeling language of MODEL.LA should be extended to fully encompass the modeling of solids, aggregate phases, electrolytes, polymers, biological systems, etc. This would require the addition of appropriate mechanistic characterizations of these materials and associated phenomena, along with the corresponding logic for derivation of the requisite model equations (including formulation of population balances).

8.4.2 Integration with Molecular Modeling Tools

The chemical processing industry places a growing emphasis on both product, as well as process,

design (Cussler, 1999). While a variety of molecular simulation algorithms and computer-aided tools are available for such purposes, there has been little progress in incorporating these tools in a generic manner into macroscopic (i.e., continuum) modeling tools. Ideally, the macroscopic modeling capabilities of MODEL.LA could be integrated with molecular modeling tools for true *multi-scale modeling*, where thermodynamic and physical properties and transport mechanisms determined through simulation on a molecular level are solved simultaneously with the modeling continuum processes on a macroscopic level.

8.4.3 Implementation of Supervisory Logic

The discussion of supervisory logic in this work proposes a basic template for incorporating contextual modeling knowledge that can enable the computer to guide the modeling activity. Significant further research would be required to explore the implementation of such guidance. Ideally, a language is needed that would allow the modeler to express the context of a particular modeling task. Rules that would allow the computer to recognize a particular context and implement appropriate guidance (by proposing alternatives and critiquing decisions) would then need to be formulated. Appropriate mechanisms that would allow the computer to subsequently analyze simulation results to verify assumptions made during the modeling activity would also be desirable.

8.4.4 Standardization and Integration with External Modeling Tools

The growing consensus in the process engineering community is that no one modeling tool is capable of meeting all the modeling needs of the chemical process industry. Rather, certain specialized models or tools are frequently required for a specific modeling objective. To address this need, standardization of computer-aided modeling tools has been proposed by organizations such as Global CAPE-OPEN (formerly CAPE-OPEN). These standards propose that all chemical process modeling tools be capable of communicating through standard interfaces. In theory, this approach would allow the integration of any number of modeling tools, databases, numerical solvers, etc., in a single model formulation. Extension of the MODEL.LA modeling environment to encompass these standards is anticipated as primarily a matter of software engineering, but may be a pivotal requirement for its widespread acceptance for modeling in the chemical process industry.

8.5 Conclusions

Since the concept of phenomena-based chemical process modeling was proposed nearly a decade ago, several proponents and skeptics have voiced their conflicting opinions on its potential benefits and ultimate feasibility. This research effort has sought to resolve this debate by *(i)* developing a phenomena-based modeling language that encompasses the modeling of dynamic, hierarchical, and spatially distributed processes, *(ii)* presenting a formalized modeling logic that explicitly represents the declarative and procedural aspects of the modeling activity, and *(iii)* integrating these concepts of language and logic in a computer-aided modeling environment in order to provide an experimental apparatus suitable evaluating these ideas. Through application of this environment to the modeling of a wide range of examples, as well as its deployment in classroom and industrial settings, the potential benefits of rapid, reliable, and documented chemical process modeling that may be realized from this high-level phenomena-based approach have been demonstrated.

Bibliography

- Abelson, H., G.J. Sussman and J. Sussman (1996). *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge.
- AIChE (1982). *Design Institute for Physical Property Data (DIPPR)*. American Institute for Chemical Engineers, New York.
- Aris, R. (1979). *Mathematical Modeling Techniques*. Pitman Publishing Ltd., London.
- Barton, P.I. (1992). *The Modelling and Simulation of Combined Discrete/Continuous Processes*. Ph.D. Thesis, Imperial College of Science, Technology and Medicine.
- Barton, P.I. and C.C. Pantelides (1993). gPROMS - A combined discrete/continuous modelling environment for chemical processing systems. *Simulation Series*, **25**, pp. 25-34.
- Bird, R.B., W.E. Stewart, and E.N. Lightfoot (1960). *Transport Phenomena*. Wiley, New York.
- Bloom, B.S. (1956). *Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain*. David McKay, New York.
- Cho. J.H. (1997). *Computer Aids for Mathematical Model Building*. Ph.D. Thesis, Imperial College of Science, Technology and Medicine.
- Cussler, R.A. (1999). Do changes in the chemical industry imply changes in curriculum? *Chemical Engineering Education*, **33**, No. 1, pp. 12-17.
- Denn, M.M. (1986). *Process Modeling*. Wiley, New York.
- Douglas, J.M. (1985). Hierarchical design procedure for continuous plants. *AIChE Journal*. **33**, pp. 353-362.
- Douglas, J.M. (1988). *Conceptual Design of Chemical Processes*. McGraw-Hill, New York.
- Duff, I.S., (1981a). On algorithms for obtaining a maximum transversal. *ACM Trans. Math. Softw.*, **4**, pp. 315-330.

- Duff, I.S., (1981b). Algorithm 575. Permutations for a zero-free diagonal. *ACM Trans. Math. Softw.*, **4**, pp. 387-390.
- Feehery, W. and P.I. Barton (1996). A differentiation-based approach to dynamic simulation and optimization with high-index differential-algebraic equations. In M. Berz, C. Bischof, G. Corliss, and A. Griewank (Eds.), *Computational Differentiation*, SIAM. pp. 239-253.
- Felder, R.M. and R.W. Rousseau (1986). *Elementary Principles of Chemical Processes*. John Wiley, NY.
- Foss, A.S. (1995). Report of Modeling Conversations at Berkeley. University of California at Berkeley Faculty Interviews.
- Froment, G.F. and K.B. Bischoff (1990). *Chemical Reactor Analysis and Design*. Wiley, New York.
- Gear, C.W. (1971). Simultaneous numerical solution of differential-algebraic equations. *IEEE Transactions on Circuit Theory*, **18**, pp. 89-95.
- Heydweiller, J.C., R.F. Sincovec, and L.T. Fan (1977). Dynamic simulation of chemical processes described by distributed and lumped parameter models. *Comp. Chem. Eng.*, **1**, pp. 125-131.
- Jarke, M. and W. Marquardt (1995). Design and evaluation of computer-aided process modeling tools. *Preprints of ISPE '95*.
- Jarvis, R.B. (1993). *Robust Dynamic Simulation of Chemical Engineering Processes*. Ph.D. Thesis, Imperial College of Science, Technology and Medicine.
- King, C.J. (1980). *Separation Processes*. McGraw-Hill, New York.
- Kowalski, R. (1979). *Logic for Problem Solving*. North-Holland, New York.
- Lloyd, M. (1985). Graphical function chart programming for programmable controllers (Grafcet). *Control Engineering*, **32**, pp.73-76.
- Maher, M.L. (1988). Expert systems for structural design. *Expert Systems in Engineering*, IFS Publications, New York.
- Marquardt, W. (1992). An object-oriented representation of structured process models. *Comp. Chem. Eng.*, **16**, suppl., pp. S329-S336.

- Marquardt, W. (1996). Trends in computer-aided process modeling. *Comp. Chem. Eng.*, **20**, No. 6/7, pp. 591-609.
- McKetta, J.J., (1977). *Encyclopedia of Chemical Processing and Design*. Dekker, New York.
- Moe, H.I. (1995). *Dynamic Process Simulation Studies on Modeling and Index Reduction*. Ph.D. Thesis, Norwegian Institute of Technology.
- Mohr, C.M. (1995). Summary of Faculty Interviews. Massachusetts Institute of Technology Faculty Interviews.
- Montgomery, S. and H.S. Fogler (1996). Selecting computer-aided instruction software. *Journal of Engineering Education*, **85**, No. 1, pp. 53-60.
- Nilsson, B. (1993). *Object-Oriented Modeling of Chemical Processes*. Ph.D. Thesis, Lund Institute of Technology.
- Nilsson, B. (1995). Experiences of developing process model libraries in OMOLA. *Preprints of ISPE '95*.
- Oh, M. (1995). *Modelling and Simulation of Combined Lumped and Distributed Processes*. Ph.D. Thesis, Imperial College of Science, Technology and Medicine.
- Oh, M. and C.C. Pantelides (1996). Modelling and simulation language for combined lumped and distributed parameter systems. *Computers & Chemical Engineering*, **20**, pp. 611-633.
- Pantelides, C.C. (1988). The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comput.*, **9**, No. 2, pp. 213-231.
- Pantelides, C.C. and H.I. Britt (1995). Multipurpose process modeling environments. In L. T. Biegler and M. F. Doherty (Eds.), *Foundations of Computer-Aided Process Design, AIChE Symposium Series*, **91**, No. 304, pp. 128-141.
- Park, T. and P.I. Barton, (1996). State event location in differential-algebraic models. *ACM Transactions on Modelling and Computer Simulation*, **6**, pp. 137-165.
- Perkins, J.D. and R.W.H. Sargent (1982). SPEEDUP—A computer program for steady-state and dynamic simulation and design of chemical processes. *AIChE Symposium Series*, No. 214, pp. 1-11.
- Perkins, J.D., R. W. H. Sargent, R. Vazquez-Roman and J. H. Cho (1996). Computer generation of process models. *Comp. Chem. Eng.*, **20**, No. 6/7, pp. 635-639.

- Petri, C.A. (1963). Fundamentals of a theory of asynchronous information flow. *Proceedings of IFIP Congress*, **No. 62**, pp. 386-390.
- Petzold, L.R. (1982). Differential/algebraic equations are not ODEs. *SIAM Journal of Scientific and Statistical Computing*, **3**, pp. 367-384.
- Piela, P. (1989). *ASCEND—An Object Oriented Environment for the Development of Quantitative Models*. Ph.D. Thesis, Carnegie Mellon University.
- Piela, P., T.G. Epperly, K.M. Westerberg, and A.W. Westerburg (1991). Ascend: An object-oriented computer environment for modeling and analysis: The modeling language. *Comp. Chem. Eng.*, **15**, 53-72.
- Ponton, J.W., P.J. Gawthrop (1991). Systematic construction of dynamic models for phase equilibrium problems. *Comp. Chem. Eng.*, **15**, 803-808.
- Preisig, H.A. (1995). MODELLER—An object-oriented computer-aided modelling tool. In L. T. Biegler and M. F. Doherty (Eds.), *Foundations of Computer-Aided Process Design, AIChE Symposium Series*, **91, No. 304**, pp. 328-331.
- Rumbaugh, J., M. Blaha, W. Premerlani, R. Eddy, and W. Lorensen (1991). *Object-Oriented Modeling and Design*. Prentice-Hall, New Jersey.
- Sipser, M. (1998). *Introduction to the Theory of Computation*. PWS Publishing, Boston.
- Steward, D.V., (1962). On an approach to techniques for the analysis of the structure of large systems of equations. *SIAM Rev.*, **4**, pp. 321-342.
- Steward, D.V., (1965). Partitioning and tearing systems of equations, *J. SIAM Numer. Anal. Ser. B.*, **2, No. 2**. 345-365.
- Sussman, G.J., and G.L. Steele (1980). Constraints--a language for expressing almost-hierarchical descriptions. *Artificial Intelligence*, **14**, pp. 1-39.
- Stephanopoulos, G., G. Henning, and H. Leone (1990a). MODEL.LA. A modeling language for process engineering—I: The formal framework. *Comp. Chem. Eng.*, **14, No. 8**, pp. 813-846.
- Stephanopoulos, G., G. Henning, and H. Leone (1990b). MODEL.LA. A modeling language for process engineering—II: Multifaceted modeling of processing systems. *Comp. Chem. Eng.*, **14, No. 8**, pp. 847-869.

- Tester, J.W. and M. Modell (1997). *Thermodynamics and Its Applications*. Prentice-Hall, New Jersey.
- Vazquez-Roman, R. (1992). *Computer Aids for Process Model-Building*. Ph.D. Thesis, Imperial College of Science, Technology and Medicine.
- Woods, E.A., (1993). *The Hybrid Phenomena Theory*. Ph.D. Thesis, Norwegian Institute of Technology.
- Zeigler, B.P. (1984). *Multifaceted Modeling and Discrete Event Simulation*. Academic Press, London.

Appendix A

MODEL.LA Context-Free Grammar

- **Overall Phenomena-Based Model**

<phenomena-based model> → <structural characterization> <chemical characterization> <derivation context>

<structural characterization> → <modeled-units> <fluxes>

<chemical characterization> → <chemical species list> <chemical reactions> <material-contents> <phases>

<derivation context> → <dynamic assumption> <mole or mass basis> <level of resolution>

<intensive or extensive characterization> <energy balance inclusion>...

- **Element Lists**

<modeled-units> → <modeled-units> <modeled-unit> | <modeled-unit>

<fluxes> → <fluxes> <flux> |

<chemical species list> → <chemical species list> <chemical species> |

<chemical reactions> → <chemical reactions> <chemical reaction> |

<material-contents> → <material-contents> <material-content> |

<phases> → <phases> <phase> |

- **Modeled-Unit**

<modeled-unit> → <unit identification> <hierarchical structure> <topological structure> <chemical content>
<modeled-unit behavioral characterizations>

<unit identification> → [modeled-unit id] *is-a* modeled-unit

<hierarchical characterization> → <parent unit> <internal characterization>

<parent unit> → *is-internal-unit-of* [modeled-unit id] |

<internal characterization> → <subunits> | <spatial distribution> | <material> | <blackbox>

<subunits> → <subunits> <subunit> | <subunit>

<subunit> → *has-internal-unit* [modeled-unit id]

<spatial distribution> → *has-spatial-distribution* <coordinate system> <differential subunits>

<coordinate system> → <rectangular coordinate> | <cylindrical coordinate> | <spherical coordinate>

<rectangular coordinate> → *rectangular* <x-characterization> <y-characterization> <z-characterization>

<cylindrical coordinate> → *cylindrical* <r-characterization> <theta-characterization> <z-characterization>

<spherical coordinate> → *spherical* <r-characterization> <theta-characterization> <phi-characterization>

<x-characterization> → <distributed x-dimension> | <undistributed x-dimension>

<y-characterization> → <distributed y-dimension> | <undistributed y-dimension>
 <z-characterization> → <distributed z-dimension> | <undistributed z-dimension>
 <r-characterization> → <distributed r-dimension> | <undistributed r-dimension>
 <theta-characterization> → <distributed theta-dimension> | <undistributed theta-dimension>
 <phi-characterization> → <distributed phi-dimension> | <undistributed phi-dimension>
 <distributed x-dimension> → *has-distributed-dimension* x <distributed solution specification>
 <distributed y-dimension> → *has-distributed-dimension* y <distributed solution specification>
 <distributed z-dimension> → *has-distributed-dimension* z <distributed solution specification>
 <distributed r-dimension> → *has-distributed-dimension* r <distributed solution specification>
 <distributed theta-dimension> → *has-distributed-dimension* theta <distributed solution specification>
 <distributed phi-dimension> → *has-distributed-dimension* phi <distributed solution specification>
 <undistributed x-dimension> → *has-undistributed-dimension* x <undistributed solution specification>
 <undistributed y-dimension> → *has-undistributed-dimension* y <undistributed solution specification>
 <undistributed z-dimension> → *has-undistributed-dimension* z <undistributed solution specification>
 <undistributed r-dimension> → *has-undistributed-dimension* r <undistributed solution specification>
 <undistributed theta-dimension> → *has-undistributed-dimension* theta <undistributed solution specification>
 <undistributed phi-dimension> → *has-undistributed-dimension* phi <undistributed solution specification>
 <distributed solution specification> → <solution method> <nodes> <minimum> <maximum>
 <solution method> → *has-solution-method* <difference method>
 <difference method> → BFDM | CFDM | FFDM | UFDM | OCFEM
 <nodes> → *has-nodes* <integer>
 <minimum> → *has-minimum* <number>
 <maximum> → *has-maximum* <number>
 <undistributed solution specification> → <minimum> <maximum>
 <differential subunits> → <differential subunits> <differential subunit> | <differential subunit>
 <differential subunit> → *has-differential-subunit* [modeled-unit id]

<material> → *has-material-content* [material-content id]

<blackbox> →

<topological structure> → <boundary inputs> <boundary outputs>
 <boundary inputs> → <boundary inputs> <boundary input> |
 <boundary outputs> → <boundary outputs> <boundary output> |
 <boundary input> → <convective input> | <energy species input> | <species input>
 <boundary output> → <convective output> | <energy output> | <species output>
 <convective input> → *has-convective-input* [flux id]
 <convective output> → *has-convective-output* [flux id]
 <energy input> → *has-energy-input* [flux id]
 <energy output> → *has-energy-output* [flux id]
 <species input> → *has-species-input* [flux id] [species id]
 <species output> → *has-species-output* [flux id] [species id]

<chemical content> → <species content list> <reactions content>
 <species content list> → <species content list> <species content> |
 <species content> → *has-species* [species id]
 <reactions content> → <reactions content> <reaction content> |
 <reactions content> → *has-reaction* [reaction id]

<modeled-unit behavioral characterizations> → <modeled-unit behavioral characterizations>
<modeled-unit behavioral characterization> |

<modeled-unit behavioral characterization> → *is-modeled-as* <modeled-unit behavioral type>
<modeled-unit behavioral type> → *is-modeled-as* no-holdup | adiabatic |...

- **Flux**

<flux> → <flux identification> <flux type> <flux connectivity>

<flux identification> → [flux id] *is-a* flux

<flux type> → <convective flux> | <energy flux> | <species flux>

<convective flux> → <convective type> <equation of state> <convective mechanism>

<convective type> → *transports* material <phase state>

<convective mechanism > → *is-modeled-by* transport-mechanism <convective mechanism type>

<convective mechanism type> → constant | pressure-driven | francis-weir |...

<energy flux> → <energy type> <energy mechanism>

<energy type> → *transports* energy

<energy mechanism > → *is-modeled-by* transport-mechanism <energy mechanism type>

<energy mechanism type> → constant | surface-convection | fourier-conduction | surface-radiation |
shaft-work | thermal-equilibrium |...

<species flux> → <species type> <species mechanism>

<species type> → *transports* species [species id]

<species mechanism > → *is-modeled-by* transport-mechanism <species mechanism type>

<species mechanism type> → constant | surface-diffusion | fickian-diffusion | knudsen-diffusion |
partial-pressure-diffusion | chemical equilibrium | partition-coefficient |...

<flux connectivity> → <source unit> <sink unit>

<source unit> → *from* [modeled-unit id]

<sink unit> → *to* [modeled-unit id]

- **Material-Content**

<material-content> → <material-content identification> <modeled-unit association> <phase instances>
<species content> <vessel geometry> <flux allocations> <material-content behavioral characterizations>

<material-content identification> → [material-content id] *is-a* material-content

<modeled-unit association> → *is-material-content-in* [modeled-unit id]

<phase instances> → <phase instances> <phase instance> | <phase instance>

<phase instance> → <vapor phase> | <liquid phase> | <solid phase>

<vapor phase> → *has-vapor-phase* [phase id]

<liquid phase> → *has-liquid-phase* [phase id]

<solid phase> → *has-solid-phase* [phase id]

<vessel geometry> → *has-vessel-geometry* <geometry type> |

<geometry type> → rectangular | spherical | vertical-cylinder | horizontal-cylinder |
vertical-annulus | horizontal-annulus | conical |...

<flux allocations> → <flux allocations> <flux allocation> |

<flux allocation> → *has-boundary-flux* [flux id] *allocated-to* <allocated element>

<allocated element> → [phase id] | geometry | self

<material-content behavioral characterizations> → <material-content behavioral characterizations>
<material-content behavioral characterization> |
<material-content behavioral characterization> → *is-modeled-as* <material-content behavioral type>
<material-content behavioral type> → constant-volume | constant-pressure | include-PV-work |
has-vessel-void...

- **Phase**

<phase> → <phase identification> <material-content association> <thermodynamic characterization>
<chemical content>

<phase identification> → [phase id] *is-a* <phase state> phase

<phase state> → vapor | liquid | solid

<material-content association> → *is-phase-in* [material-content id]

<thermodynamic characterization> → <equation of state> | <activity coefficient>

<equation of state> → *is-modeled-by* equation-of-state <equation of state type>

<equation of state type> → ideal-gas | incompressible | van-der-waals | redlich-kwong |
redlich-kwong-soave | peng-robinson |...

<activity coefficient> → *is-modeled-by* activity-coefficient-model <activity coefficient model>

<activity coefficient model> → ideal | margules | van-laar | wilson | nrtl | unifac |...

- **Chemical Species**

<chemical species> → <species identification> <database id>

<species identification> → [species id] *is-a* chemical-species

<database id> → *has-database-id* <integer>

- **Chemical Reaction**

<chemical reaction> → <reaction identification> <participants> <kinetics>

<reaction identification> → [reaction id] *is-a* chemical-reaction

<participants> → <reactants> <reversibility> <products> <catalyst>

<reactants> → <reactants> <stoichiometry> | <stoichiometry>

<stoichiometry> → + <number> [species id]

<reversibility> → ⇒|⇌|=

<products> → <products> <stoichiometry> | <stoichiometry>

<catalyst> → *has-catalyst* [species id] |

<kinetics> → <forward rate law> <reverse rate law>

<forward rate law> → *has-forward-kinetics* [equation]

<reverse rate law> → *has-reverse-kinetics* [equation] |

- **Miscellaneous**

[modeled-unit id] → <string>

[flux id] → <string>

[material-content id] → <string>

[phase id] → <string>

[reaction id] → <string>

[species id] → <string>

<string> → <letter><characters>

<characters> → <letter>|<digit>|_|

<letter> → A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z|a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

<digit> → 1|2|3|4|5|6|7|8|9|0

<integer> → <sign><digits>

<sign> → + | - |

<digits> → <digits><digit>|<digit>

<number> → <integer><fraction>

<fraction> → .<digits>|

Appendix B

Properties Manager

This appendix provides additional details regarding the features and functionality of the MODEL.LA *Properties Manager*.

- **Purpose**

The purpose of the MODEL.LA *Properties Manager* is to construct equation-based correlations that describe the thermodynamic and physical properties of pure chemical species and mixtures. Within the MODEL.LA Modeling Environment, these property equations supplement the conservation and constitutive equations that are derived by the MODEL.LA *Model Generator* from the phenomena-based model description.

The *Properties Manager* also provides additional facilities for the analysis of the thermodynamic and physical behavior of pure chemical species and mixtures, independently from a phenomena-based modeling context.

- **Properties of Pure Chemical Species**

The *Properties Manager* integrates a database of thermodynamic and physical properties for pure chemical species. The constant value and temperature-dependent properties contained in this database mirror those of the DIPPR pure species database (AIChE, 1982). Use of this common format enables straightforward incorporation of data provided by DIPPR (which currently encompasses the measured and predicted properties of over 1400 species), and is also readily extended to incorporate the property data of other species through a standard representation.

Through use of the *Properties Manager*, the modeler is not required to access and manipulate the pure species property database tables directly. Rather, the data is accessed through a series of graphical dialogs. A list of all chemical species contained in the database is

accessed using the *Project Species Dialog* (illustrated in Figure 5-11). Species identification properties for each species, accessed using the dialog of Figure 5-27, are summarized in Table B-1. Constant value properties for each species, accessed using the dialog of Figure 5-28, are summarized in Table B-2. Temperature dependant properties for each species, accessed using the dialog of Figure 5-29, are summarized in Table B-3. Each temperature-dependent property is characterized by a temperature-dependent correlation with up to five parameters, and a temperature range, characterized by a minimum and maximum bounds, over which the correlated data is valid. Available correlations are summarized in Table B-4. The database also contains UNIFAC group characterizations for each species, used for calculation of activity coefficients for vapor-liquid equilibrium and liquid-liquid equilibrium systems, which are accessed using the dialog illustrated in Figure 5-30. The structure of the properties database is summarized by the object model illustrated in Figure 6-16.

Within a phenomena-based modeling context, pure species thermodynamic and physical property data are primarily used for the construction of thermodynamic and physical property correlations of mixtures of components (i.e., phases in a phenomena-based model).

Table B-1: Pure Species Identification Properties

- | |
|------------------------|
| 1. Database Index |
| 2. Common Name |
| 3. IUPAC Name |
| 4. CAS Registry Number |
| 5. Molecular Formula |
| 6. Structural Formula |
| 7. Chemical Group |

Table B-2: Pure Species Constant Value Properties

<u>Property</u>	<u>Units</u>	<u>Conditions</u>
8. Molecular Weight	kg/kg-mol	—
9. Critical Temperature	K	—
10. Critical Pressure	Pa	—
11. Critical Volume	m ³ /kg-mol	—
12. Critical Compressibility Factor	—	—
13. Melting Point	K	1 atm
14. Triple Point Temperature	K	—
15. Triple Point Pressure	Pa	—
16. Normal Boiling Point	K	1 atm
17. Liquid Molar Volume	m ³ /kg-mol	298.15 K
18. Enthalpy of Formation of Ideal Gas	J/kg-mol	298.15 K
19. Gibbs Free Energy of Formation of Ideal Gas	J/kg-mol	298.15 K and 1atm
20. Absolute Entropy of Ideal Gas	J/kg-mol K	298.15 K and 1atm
21. Enthalpy of Fusion at Melting Point	J/kg-mol	—
22. Net Enthalpy of Combustion	J/kg-mol	298.15 K
23. Acentric Factor	—	—
24. Solubility Parameter	J ^{1/2} /m ^{3/2}	298.15 K
25. Dipole Moment	c m	—
26. van der Waals Reduced Volume	m ³ /kg-mol	—
27. van der Waals Area	m ² /kg-mol	—
28. Lower Flammability Limit	vol % in air	—
29. Upper Flammability Limit	vol % in air	—
30. Auto-ignition Temperature	K	—

Table B-3: Pure Species Temperature-Dependent Properties

<i>Property</i>	<i>Units</i>	<i>Conditions</i>
31. Liquid Density	kg-mol /m ³	1 atm below normal boiling point
32. Solid Density	kg-mol /m ³	—
33. Heat of Vaporization	J/kg-mol	—
34. Ideal Gas Heat Capacity	J/kg-mol K	—
35. Liquid Heat Capacity	J/kg-mol K	1 atm below normal boiling point
36. Solid Heat Capacity	J/kg-mol K	—
37. Vapor Pressure	Pa	—
38. Second Virial Coefficient	m ³ /kg-mol	—
39. Absolute Liquid Viscosity	Pa s	1 atm below normal boiling point
40. Vapor Viscosity	Pa s	—
41. Liquid Thermal Conductivity	W/ m K	—
42. Vapor Thermal Conductivity	W/ m K	1 atm or below
43. Surface Tension	N/m	1 atm below normal boiling point

Table B-4: Pure Species Temperature-Dependent Property Correlations

<u>Correlation Number</u>	<u>Correlation</u> $Y = \text{property}, T = \text{temperature (K)}, T_c = \text{critical temperature (K)}$	<u>Parameters</u>
100	$Y = A + BT + CT^2 + DT^3 + ET^4$	A, B, C, D, E
101	$Y = \exp\left(A + \frac{B}{T} + C \ln(T) + DT^E\right)$	A, B, C, D, E
102	$Y = \frac{AT^B}{1 + \frac{C}{T} + \frac{D}{T^2}}$	A, B, C, D
103	$Y = A + B \exp\left(-\frac{C}{T^D}\right)$	A, B, C, D
104	$Y = A + \frac{B}{T} + \frac{C}{T^3} + \frac{D}{T^8} + \frac{E}{T^9}$	A, B, C, D, E
105	$Y = \frac{A}{B^{1+(1-T/C)^D}}$	A, B, C, D
106	$Y = A(1 - T_r)^{B+CT_r+DT_r^2+ET_r^3}$ $T_r = \frac{T}{T_c}$	A, B, C, D, E
107	$Y = A + B\left(\frac{(C/T)}{\sinh(C/T)}\right)^2 + D\left(\frac{(E/T)}{\cosh(E/T)}\right)^2$	A, B, C, D, E
114	$Y = \frac{A^2}{t} + B - 2ACt - ADt^2 - \frac{C^2t^3}{3} - \frac{CDt^4}{2} - \frac{D^2t^5}{5}$ $t = 1 - \frac{T}{T_c}$	A, B, C, D
115	$Y = \exp\left(A + \frac{B}{T} + C \ln(T) + DT^2 + \frac{E}{T^2}\right)$	A, B, C, D, E

- **Properties of Phases**

Correlations that describe the physical and thermodynamic properties of phases in a phenomena-based model are constructed by the *Properties Manager* based on pure component species data and, when appropriate, equation of state or activity coefficient models selected by the modeler to characterize the thermodynamic behavior of the phase. Thermodynamic equation of state models supported by the *Properties Manager* are listed in Table B-5. Further details regarding these equations of state and the suggested mixing rules shown are discussed in Reid et al (1984). The equation of state mixing rules introduce binary interaction parameters for each pair of chemical species in the phase. These binary interaction parameters are also stored in the *Properties Manager* database and may be accessed using the dialog shown in Figure 5-31. Thermodynamic activity coefficient models supported by the *Properties Manager* are listed in Table B-6. Activity coefficient correlations in Table B-6 are shown for binary systems. Generalized multicomponent correlations used are also summarized in Reid et al (1984).

Table B-5: Properties Manager Equations of State

<u>Equation of State</u>	<u>Correlation</u> (k_{ij} = Binary Interaction Parameter)
Ideal Gas	$P = RT / \underline{V}$
van der Waals	$P = \frac{RT}{\underline{V} - b_m} - \frac{a_m}{\underline{V}^2}$ $a_m = \sum_i \sum_j x_i x_j (a_i a_j)^{1/2} (1 - k_{ij}) \quad b_m = \sum_i x_i b_i$ $a_i = \frac{27R^2 T_{c,i}^2}{64P_{c,i}} \quad b_i = \frac{RT_{c,i}}{8P_{c,i}}$
Redlich-Kwong	$P = \frac{RT}{\underline{V} - b_m} - \frac{a_m}{T^{1/2} \underline{V}(\underline{V} + b_m)}$ $a_m = \sum_i \sum_j x_i x_j (a_i a_j)^{1/2} (1 - k_{ij}) \quad b_m = \sum_i x_i b_i$ $a_i = \frac{0.42748R^2 T_{c,i}^{2.5}}{P_{c,i}} \quad b_i = \frac{0.08664RT_{c,i}}{P_{c,i}}$
Redlich-Kwong-Soave	$P = \frac{RT}{\underline{V} - b_m} - \frac{a_m(T)}{\underline{V}(\underline{V} + b_m)}$ $a_m(T) = \sum_i \sum_j x_i x_j (a_i a_j)^{1/2} (1 - k_{ij}) \quad b_m = \sum_i x_i b_i$ $a_i(T) = \frac{0.42748R^2 T_{c,i}^2}{P_{c,i}} [1 + f\omega_i (1 - T_{r,i}^{1/2})]^2 \quad b_i = \frac{0.08664RT_{c,i}}{P_{c,i}}$ $f\omega_i = 0.48 + 1.574\omega_i - 0.176\omega_i^2$
Peng-Robinson	$P = \frac{RT}{\underline{V} - b_m} - \frac{a_m(T)}{\underline{V}(\underline{V} + b_m) + b_m(\underline{V} - b_m)}$ $a_m(T) = \sum_i \sum_j x_i x_j (a_i a_j)^{1/2} (1 - k_{ij}) \quad b_m = \sum_i x_i b_i$ $a_i(T) = \frac{0.45724R^2 T_{c,i}^2}{P_{c,i}} [1 + f\omega_i (1 - T_{r,i}^{1/2})]^2 \quad b_i = \frac{0.07780RT_{c,i}}{P_{c,i}}$ $f\omega_i = 0.37464 + 1.54226\omega_i - 0.26992\omega_i^2$
Virial (simplified)	$P = \frac{RT}{\underline{V} - B_m}$ $B_m = \sum_i \sum_j x_i x_j B_{ij} \quad B_{ij} = \frac{RT_{c,ij}}{P_{c,ij}} (B_{ij}^{(0)} + \omega_{ij} B_{ij}^{(1)})$ $T_{c,ij} = (1 - k_{ij})(T_{c,i} T_{c,j})^{1/2}$

Table B-6: Properties Manager Activity Coefficient Models

<u>Activity Coefficient Model</u>	<u>Correlation</u> (shown for binary component mixtures)	<u>Binary Interaction Parameters</u>
Ideal Solution	$\gamma = 1$	—
Margules (four suffix)	$T \ln \gamma_1 = (A + 3B + 5C)x_2^2 - 4(B + 4C)x_2^3 + 12Cx_2^4$	A, B, C
van Laar	$T \ln \gamma_1 = A \left(1 + \frac{Ax_1}{Bx_2} \right)^{-2}$	A, B
Wilson	$\ln \gamma_1 = \ln(x_1 + \Lambda_{12}x_2) + x_2 \left(\frac{\Lambda_{12}}{x_1 + \Lambda_{12}x_2} - \frac{\Lambda_{21}}{x_1 + \Lambda_{21}x_2} \right)$ $\Lambda_{12} = \frac{V_2}{V_1} \exp\left(-\frac{a_{12}}{T}\right) \quad \Lambda_{21} = \frac{V_1}{V_2} \exp\left(-\frac{a_{21}}{T}\right)$	a_{12}, a_{21}
NRTL	$\ln \gamma_1 = x_2^2 \left[\tau_{21} \left(\frac{G_{21}}{x_1 + x_2 G_{21}} \right)^2 + \tau_{12} \frac{G_{12}}{(x_2 + x_1 G_{12})^2} \right]$ $\tau_{12} = \frac{g_{12}}{T} \quad \tau_{21} = \frac{g_{21}}{T} \quad \ln G_{12} = -a_{12} \tau_{12} \quad \ln G_{21} = -a_{12} \tau_{21}$	g_{12}, g_{21}, a_{12}
UNIQUAC	$\ln \gamma_1 = \ln \frac{\Phi_1}{x_1} + \frac{z}{2} q_1 \ln \frac{\theta_1}{\Phi_1} + \Phi_2 \left(l_1 - \frac{r_1}{r_2} l_2 \right) - q_1 \ln(\theta_1 + \theta_2 \tau_{21})$ $+ \theta_2 q_1 \left(\frac{\tau_{21}}{\theta_1 + \theta_2 \tau_{21}} - \frac{\tau_{12}}{\theta_2 + \theta_1 \tau_{12}} \right)$ $\ln \tau_{12} = -\frac{u_{12}}{RT} \quad \ln \tau_{21} = -\frac{u_{21}}{RT}$	u_{12}, u_{21}
UNIFAC	based on UNIQUAC equation, calculates interactions based on structural subgroups of components	—

The thermodynamic and physical properties correlations constructed by the *Properties Manager* for phases are listed in Table B-7.

Table B-7: Thermodynamic and Physical Property Correlations for Phases

<u>Property</u>	<u>Units</u>
1. Average Molecular Weight	kg/kg-mol
2. Molar Density	kg-mol/m ³
3. Molar Volume	m ³ /kg-mol
4. Component Fugacity	Pa
5. Component Fugacity Coefficient	—
6. Component Activity	—
7. Component Activity Coefficient	—
8. Internal Energy	J/kg-mol
9. Enthalpy	J/kg-mol
10. Gibbs Free Energy	J/kg-mol
11. Entropy	J/kg-mol K
12. Helmholtz Energy	J/kg-mol
13. Heat Capacity	J/kg-mol K

The formulation of each of these property correlations for phases is a phenomena-based model is now discussed.

1. Average Molecular Weight:

The average molecular weight of a phase is calculated as:

$$MW_{ave} = \sum_{species} x_i MW_i$$

where

1. MW_{ave} is the average molecular weight of the phase,
2. x_i is the mole fraction of the i^{th} species component in the phase, and
3. MW_i is the molecular weight of the i^{th} species component in the phase.

2. Molar Density:

For phases modeled as incompressible (i.e., the density of the phase is not dependent on pressure), the molar density is calculated assuming ideal solution behavior:

$$\rho(T, \underline{\mathbf{x}}) = \left(\sum_{\text{species}} \frac{x_i}{\rho_i(T)} \right)^{-1}$$

where

1. ρ is the molar density of the phase,
2. x_i is the mole fraction of the i^{th} species component in the phase, and
3. ρ_i is the molar density of the i^{th} species component in the phase.

For phases modeled using an equation of state, the molar density is calculated as:

$$\rho(P, T, \underline{\mathbf{x}}) = \underline{V}^{-1}(P, T, \underline{\mathbf{x}})$$

where

1. ρ is the molar density of the phase, and
2. \underline{V} is the molar volume of the phase (determined by the equation of state model selected for the phase).
3. Molar Volume:

For phases modeled as incompressible, the molar volume is calculated assuming ideal solution behavior:

$$\underline{V}(T, \underline{\mathbf{x}}) = \sum_{\text{species}} \frac{x_i}{\rho_i(T)}$$

where

1. \underline{V} is the molar volume of the phase,
2. x_i is the mole fraction of the i^{th} species component in the phase, and
3. ρ_i is the molar density of the i^{th} species component in the mixture.

For phases modeled using an equation of state, the molar density is expressed as a function of pressure, temperature, and composition:

$$f(P, \underline{V}, T, \underline{\mathbf{x}}) = 0$$

where

1. P is the pressure of the phase,
2. \underline{V} is the molar volume of the phase,
3. T is the temperature of the phase, and
4. $\underline{\mathbf{x}}$ is the vector of mole fractions for all species in the phase.

As shown in Table B-5, non-ideal equation of state models will also contain a set of binary interaction parameters for all pairs of species in the phase.

4. Fugacity:

For phases modeled as incompressible, the fugacity of each species in the phase is calculated (neglecting the Poynting correction factor) as:

$$f_i = x_i \gamma_i P_{vp,i}$$

where

1. f_i is the fugacity of the i^{th} species component in the phase,
2. x_i is the mole fraction of the i^{th} species component in the phase,
3. γ_i is the activity coefficient of the i^{th} species component in the phase, and
4. $P_{vp,i}$ is the vapor pressure of the i^{th} species component in the phase.

For phases modeled using an equation of state, the fugacity of each species in the phase is expressed as:

$$f_i = x_i \phi_i P$$

where

1. f_i is the fugacity of the i^{th} species component in the phase,
2. x_i is the mole fraction of the i^{th} species component in the phase,
3. ϕ_i is the fugacity coefficient of the i^{th} species component in the phase, and
4. P is the pressure of the phase.

5. Fugacity Coefficient:

For phases modeled using an equation of state, the fugacity coefficient of each species in the phase is determined by evaluation of the following integral:

$$RT \ln \phi_i = - \int_{\infty}^V \left[\left(\frac{\partial P}{\partial N_i} \right)_{T, V, N_j [i]} - \frac{RT}{V} \right] dV - RT \ln(Z)$$

where

1. ϕ_i is the fugacity coefficient of the i^{th} species component in the phase,
2. T is the temperature of the phase,
3. P is the pressure of the phase,
4. V is the molar volume of the phase,

5. Z is the compressibility factor of the phase, and
6. \underline{x} is the vector of mole fractions for all species in the phase.

Integrated forms of this equation for the equations of state in Table B-5 are given in Reid et al (1984).

6. Activity:

For incompressible phases modeled using an activity coefficient model, the activity of each species in the phase is expressed as:

$$a_i = x_i \gamma_i$$

where

1. a_i is the activity of the i^{th} species component in the phase, and
2. γ_i is the activity coefficient of the i^{th} species component in the phase.

7. Activity Coefficient:

For incompressible phases modeled using an activity coefficient model, the activity coefficient of each species in the phase is expressed as:

$$\gamma_i = f(T, \underline{x})$$

where

1. γ_i is the activity coefficient of the i^{th} species component in the phase,
2. T is the temperature of the phase, and
3. \underline{x} is the vector of mole fractions for all species in the phase.

The activity model correlations for binary systems are summarized in Table B-6. Correlations for multicomponent mixtures may be found in Reid et al (1984).

8, 9, 10, 11, 12. Internal Energy, Enthalpy, Gibbs Free Energy, Entropy, Helmholtz Energy :

For phases modeled as incompressible, thermodynamic variables are calculated using excess properties based on the selected activity coefficient model:

$$G = G^E + \sum_{\text{species}} x_i G_i + RT \sum_{\text{species}} x_i \ln x_i$$

$$S = S^E + \sum_{\text{species}} x_i S_i - R \sum_{\text{species}} x_i \ln x_i$$

$$A = A^E + \sum_{\text{species}} x_i A_i + RT \sum_{\text{species}} x_i \ln x_i$$

$$H = H^E + \sum_{\text{species}} x_i H_i$$

$$U = U^E + \sum_{\text{species}} x_i U_i$$

where

1. G is the Gibbs free energy of the phase,
2. S is the entropy of the phase,
3. A is the Helmholtz energy of the phase,
4. H is the enthalpy of the phase,
5. U is the internal energy of the phase,
6. x_i is the mole fraction of the i^{th} species component in the phase, and
7. E (superscript) represents an excess property.

For phases modeled using an equation of state, the thermodynamic variables are calculated using departure functions based on the selected equation of state model:

$$A(T, \underline{V}) - A^o(T, \underline{V}^o) = - \int_{\infty}^{\underline{V}} \left(P - \frac{PT}{\underline{V}} \right) d\underline{V} + RT \ln \left(\frac{\underline{V}^o}{\underline{V}} \right)$$

$$S(T, \underline{V}) - S^o(T, \underline{V}^o) = \frac{\partial}{\partial T} \int_{\infty}^{\underline{V}} \left(P - \frac{PT}{\underline{V}} \right) d\underline{V} + R \ln \left(\frac{\underline{V}^o}{\underline{V}} \right)$$

$$U(T, \underline{V}) - U^o(T, \underline{V}^o) = [A(T, \underline{V}) - A^o(T, \underline{V}^o)] + T[S(T, \underline{V}) - S^o(T, \underline{V}^o)]$$

$$H(T, \underline{V}) - H^o(T, \underline{V}^o) = [U(T, \underline{V}) - U^o(T, \underline{V}^o)] + PV - RT$$

$$G(T, \underline{V}) - G^o(T, \underline{V}^o) = [H(T, \underline{V}) - H^o(T, \underline{V}^o)] - T[S(T, \underline{V}) - S^o(T, \underline{V}^o)]$$

where

1. A is the Helmholtz energy of the phase,
2. S is the entropy of the phase,
3. U is the internal energy of the phase,
4. H is the enthalpy of the phase,
5. G is the Gibbs free energy of the phase, and
6. O (superscript) represents a reference state property.

Integrated forms of the departure functions for the equations of state in Table B-5 are given in Reid et al (1984). The use of departure functions for the calculation of thermodynamic mixtures

using equation of state models is discussed in detail by Tester and Modell (1997).

13. Heat Capacity:

For phases modeled as incompressible (i.e., the density of the phase is not dependent on pressure), the molar heat capacity is calculated assuming ideal solution behavior:

$$c_p(T, \underline{x}) = \sum_{\text{species}} x_i c_{p,i}(T)$$

where

1. c_p is the molar heat capacity of the phase,
2. x_i is the mole fraction of the i^{th} species component in the phase, and
3. $c_{p,i}$ is the molar heat capacity of the i^{th} species component in the phase.

For phases modeled using an equation of state, the molar heat capacity is calculated using a departure function based on the selected equation of state model:

$$c_p(T, \underline{V}) - c_p^O(T, \underline{V}^O) = T \int_{\underline{V}^O}^{\underline{V}} \left(\frac{\partial^2 P}{\partial T^2} \right)_{\underline{V}} d\underline{V} - \frac{T(\partial P / \partial T)_{\underline{V}}^2}{(\partial P / \partial \underline{V})_T} - R$$

where

1. c_p is the molar heat capacity of the phase,
2. P is the pressure of the phase,
3. \underline{V} is the molar volume of the phase,
4. T is the temperature of the phase, and
5. O (superscript) represents a reference state property.

- **Analysis of Phase Behavior**

Analysis of the thermodynamic and physical behavior of phases is facilitated through the *Phase Properties Dialog* (illustrated in Figure 5-34). Here, property correlations constructed based on pure species properties and selected equation of state or activity coefficient models may be viewed and plotted.

Appendix C

Operational Schedules

This appendix provides additional details regarding the declaration of operational schedules through use of the MODEL.LA *Operations Manager*.

- **Purpose**

The purpose of an operational schedule declared within the MODEL.LA *Operations Manager* is to impose discrete actions on the behavior of an otherwise continuous process model. Within the MODEL.LA Modeling Environment, this results in hybrid discrete and continuous behavior exhibited during subsequent simulation of a phenomena-based process model.

- **Hybrid Systems**

The mathematical model derived from a phenomena-based model representation consists of a set of continuous equations that are based on conservation principles. The behavior of such a model is characterized by a set of state variables (e.g., mass, energy, temperature, composition, etc.) that vary continuously in time. However, external discrete actions imposed on the system through manipulation of some process quantity (e.g., set flow rate to zero) result in discontinuous process behavior. Such models that exhibit both discrete and continuous behavior have been termed *hybrid systems*.

The general structure of a hybrid system may be interpreted as shown in Figure C-1. The state of a physical system is monitored by sensors that relay this information to an external control system. Based on the state of the physical system, the control system may implement discrete (or continuous) actions on the physical system through actuators that manipulate a variable or variables associated with the physical system.

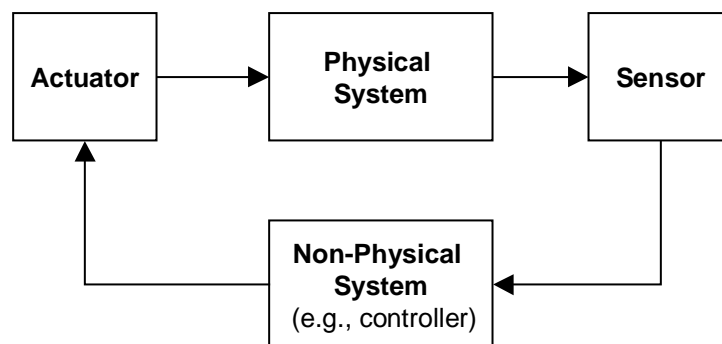


Figure C-1: Generic Structure of Hybrid System

- **Operational Schedules**

In chemical plants, the two predominant types of procedures that result in hybrid systems are control procedures and operating procedures. In general, control procedures sample variables at regular intervals of time, which may be interpreted as time events. At each time event an action is performed—the value of a manipulated variable is changed based on a specified control law. In general, operating procedures encompass both time events and state events. These operating procedures may be interpreted as operational schedules that encompass a sequence of actions that are performed on the process when some conditions are satisfied (i.e., time events or state events). These operational schedules are an inherent part of the operation of a batch plant and appear commonly during the startup, shutdown, and maintenance of a continuous plant.

Several formalisms have been proposed to describe discrete systems. Examples include finite state automata (Sipser, 1998), DEVS formalism (Zeigler, 1984), Petri Nets (Petri, 1963), and Grafset (Lloyd, 1985). Computer-aided packages that allow the simulation of hybrid discrete/continuous systems include gPROMS (Barton, 1992), OMOLA (Nilsson, 1993), and ABACUSS (Feehery and Barton, 1996).

- **Graphical Declaration of Schedules**

In the MODEL.LA *Operations Manager*, operational schedules are declared using a graphical syntax based on a set of graphical elements. A schedule is composed of a sequence of elementary and/or composite actions. An elementary action represents the manipulation of a process variable. A composite action is itself a schedule that is composed of a sequence of elementary and/or composite actions. The sequence of actions carried out in a schedule is dictated by events that link the actions. Certain events are associated with a condition based on a set of one or more

measured process variables. When an event has control in a schedule and the condition associated with the event becomes true, the subsequent action in the schedule is triggered. Control then passes to the subsequent event, as dictated by the topology of the schedule events.

In MODEL.LA, tasks are declared as icons on a process flowsheet where they are associated with a one or more measured variables and one or more manipulated variable through a set of transmission lines. Subsequently, these tasks are used in the construction of any number of operational schedules in the MODEL.LA *Operations Manager*. Here, schedules are constructed graphically as flowcharts. The graphical elements that are used to compose a schedule are summarized in Table C-1. Each of these elements are now discussed.

1. Events:

An event in a schedule is represented by an arrow line that connects two actions. The orientation of the arrow depicts the flow of control (i.e., sequence of manipulations) in a schedule. An event may be of type *void*, *when*, *while*, *end while*, *if*, and *if not*.

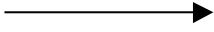
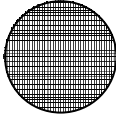

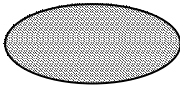
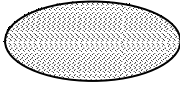
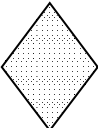

1.A Void Event:

When control is passed to a *void* event, the subsequent associated action is immediately triggered. In this manner, a sequence of actions connected by *void* events are used to trigger a set of actions simultaneously.

1.B When Event:

When control is passed to a *when* event, the subsequent action is triggered once the conditional defined for the event becomes true. A *when* event is associated with a task that has been declared on the process flowsheet. Variables measured by the task are used to construct the conditional associated with the *when* event.

Table C-1: Operational Schedule Elements

<u>Element</u>	<u>Icon</u>	<u>Purpose</u>
Event	 (black arrow)	1. <i>Void</i> events trigger simultaneous actions 2. <i>When</i> event triggers action based on time or state event 3. <i>While/end while</i> events establish loop in schedule 4. <i>If/if not</i> events establish conditional branches in schedule
Initial Action	 (green circle)	Establishes starting point of schedule
End Action	 (red circle)	Establishes termination of branch in schedule
Elementary Action	 (blue ellipse)	Represents manipulation of process variable
Composite Action	 (olive ellipse)	Represents abstraction of sequence of events and actions
Condition Action	 (yellow diamond)	Establishes conditional branches during execution of schedule
Parallel Action	 (black bar)	Establishes multiple independent execution paths in schedule

1.C While and End While Events:

A *while* event establishes the beginning of a loop of actions in the schedule. The closure of the loop is established by a subsequent *end while* event. Similar to a *when* event, a *while* event is associated with a task that has been declared on the process flowsheet. Variables measured by the task are used to construct the conditional associated with the *while* event. When control is passed to a *while* event, the associated action is activated if the conditional associated with the *while* event is true. Execution then continues until control passes to the subsequent *end while* event in the schedule. At that point control returns to the initial *while* event. In this manner, an iterative loop of actions is established. When control is passed to a *while* event and the associated conditional is not true, the action subsequent to the *end while* event that ends the loop is triggered. In this manner, the loop is terminated.

1.D If and If Not Events:

A condition action establishes a pair of alternative branches encountered during the execution of a schedule. Each condition action is associated a conditional and a subsequent *if* and an *if not* event. When a condition action is triggered, control is passed to the *if* event if the conditional associated with the condition action is true. If it is false, control is passed to the *if not* event. Similar to a void event, when control is passed to an *if* or an *if not* event, the subsequent associated action is immediately triggered.

2. Initial Action:

Every schedule must have exactly one initial action. It is represented in a schedule by a green circle. The initial action establishes the starting point of the schedule. The initial action is associated with a null action (i.e., no process manipulation occurs) and thus immediately passes control to the subsequent event.

3. End Action:

Every path in a schedule must terminate at an end action. It is represented in a schedule by a red circle. The end action is associated with a null action and represents termination of the operational schedule.

4. Elementary Action:

An elementary action is represented in a schedule by a blue ellipse. It represents a change in the value of a manipulated variable in the process. An elementary action is associated with a task that

has been declared on the process flowsheet. Variables manipulated by the task are used to construct the action (represented by an equation) associated with the elementary action.

5. Composite Action:

A composite action represents an abstraction of a sequence of actions and events. It is represented in a schedule by an olive ellipse. A composite action is itself a well-defined schedule. This enables a hierarchical declaration of operational schedules. The actions and events that compose a composite action are declared on a separate decomposition flowchart. When a composite action is triggered, control passes to the initial action in the decomposition of the composite action. Execution of the sub-schedule continues until an associated end action is triggered. Control is then passed to the event subsequent to the composite action in the parent schedule.

6. Condition Action:

An condition action is represented in a schedule by a yellow diamond. A condition action establishes a pair of alternative branches encountered during the execution of a schedule. A condition action is associated with a null action. Therefore no process variable is manipulated. Rather, the condition action is associated with a task that has been declared on the process flowsheet. Variables measured by the task are used to construct the conditional associated with the condition action. Each condition action is also associated with a subsequent *if* and *if not* event. When the condition action is triggered, control is passed to the *if* event if the conditional is true. If it is false, control is passed to the *if not* event.

7. Parallel Action:

An parallel action is represented in a schedule by a solid black bar. A parallel action enables the execution of two or more simultaneous paths in a operational schedule. A parallel action is associated with a null action. Therefore no process variable is manipulated. Rather, when a parallel action is triggered, control is passed to all events immediately subsequent to the action. Each path is then executed independently and concurrently. The overall operational schedule terminates when all resulting paths in the schedule terminate at end actions.

- **Schedule Construction**

The first step in constructing a schedule in MODEL.LA is to declare the necessary tasks as icons on the phenomena-based model flowsheet. Each task is associated with a set of measured process variables (declared by transmission lines incident to the task) and a set of manipulated process

variables (declared by transmission lines incident from the task). For example, in Figure 5-41, *feed_task* represents a task that measures the volume of the *Reactor* and *Storage* systems through transmission lines *reactor_vol_fill* and *storage_vol*, respectively, and manipulates the volumetric flow rate of the *feed* stream through transmission line *feed_flow*. The measured process variables are used to construct conditionals for *when* events, *while* events, and condition actions. The manipulated process variables are used to construct action equations for elementary actions.

Each schedule is constructed using the elements described in the preceding section. These are presented to the modeler on the *Modeling Assistant* (illustrated at the bottom of Figure 5-41). Rules for construction of a valid schedule are now presented.

- All schedules must contain exactly one initial action that has no events incident to it and exactly one *void*, *when*, or *while* event incident from it.
- Every branch in a schedule must terminate at an end action, which must have exactly one *void*, *when*, *if*, *if not*, or *end while* event incident to it and no events incident from it.
- Every elementary, composite, condition, and parallel action in a schedule must have exactly one event (of any kind) incident to it.
- Every elementary and composite action in a schedule must have exactly one *void*, *when*, *while*, or *end while* event incident from it.
- Every condition element must have exactly one *if* and exactly one *if not* event incident from it.
- Every parallel branch may have any number (greater than zero) of events incident from it of type *void*, *when*, or *while*.
- Every loop established by a *while* event must terminate at a *end while* event.
- Every *when* event, *while* event, and condition element must either be associated with a time event or a state event characterized by a conditional. Events associated with state events must be associated with a task that measures the variables used to construct the characterizing conditional.
- Every initial, final, conditional, and parallel events are by default associated with null actions.
- Every elementary action must be associated with a task that manipulates a process

variable and an equation that characterizes the manipulation of the variable.

– Every composite action must be associated with a distinct, valid sub-schedule.

- **Schedule Execution**

For a given phenomena-based model description and set of tasks, any number of schedules may be declared. However, when the model is simulated, only the selected active schedule (which may integrate other sub-schedules through composite actions) is implemented for execution. The schedule is posed in the gPROMS input language for simulation with the model equations derived from the phenomena-based model description. Each elementary action in a schedule is declared as a gPROMS TASK which RESETs the value of a process variable. Table C-2 illustrates the declaration of the schedule depicted in Figure 5-41. The first four entries represent elementary actions *feed_on*, *feed_off*, *product_on*, and *product_off*. Each action manipulates a process variable from the phenomena-based model equations (contained in MODEL *The_Model*). The main schedule and any composite action sub-schedules are also declared as gPROMS TASKs which list the schedule actions in a SEQUENCE associated with the SCHEDULE of the TASK. Elementary or composite actions linked by *void* events are listed sequentially in a SEQUENCE. *When* events are declared as CONTINUE UNTIL statements that separate elementary or composite events. *While* events are declared as WHILE [condition] DO statements that terminate at *End While* events declared as END statements. Conditional actions are declared as IF/THEN/ELSE statements. Finally, parallel actions are declared as PARALLEL statements.

Table C-2: Example gPROMS Schedule Translation

<u><i>gPROMS Tasks</i></u>
<pre>TASK feed_on PARAMETER The_Model AS MODEL The_Model SCHEDULE SEQUENCE RESET The_Model.v_flux_feed := 0.1 ; END END END</pre>
<pre>TASK feed_off PARAMETER The_Model AS MODEL The_Model SCHEDULE SEQUENCE RESET The_Model.v_flux_feed := 0.001 ; END END END</pre>
<pre>TASK product_on PARAMETER The_Model AS MODEL The_Model SCHEDULE SEQUENCE RESET The_Model.v_flux_product := 0.1 ; END END END</pre>
<pre>TASK product_off PARAMETER The_Model AS MODEL The_Model SCHEDULE SEQUENCE RESET The_Model.v_flux_product := 0.001 ; END END END</pre>
<pre>TASK main PARAMETER The_Model AS MODEL The_Model SCHEDULE SEQUENCE WHILE The_Model.V_Storageatl_L0 < 100 AND TIME <= 5000 DO SEQUENCE feed_on(The_Model IS The_Model); CONTINUE UNTIL The_Model.V_Reactormatl_L0 >= 10 OR TIME > 5000 feed_off(The_Model IS The_Model); CONTINUE UNTIL The_Model.r_rxn0_forw_Reactormatl_L0 <= 0.001 OR TIME > 5000 product_on(The_Model IS The_Model); CONTINUE UNTIL The_Model.V_Reactormatl_L0 <= 0.1 OR TIME > 5000 product_off(The_Model IS The_Model); END END END END</pre>
<pre>END</pre>

Appendix D

Jacketed-CSTR Model Equations

1. Unit Jacketed_Cstr species ACETIC_ACID mole quantity decomposition

$$\mathbf{N}_{ACETIC_ACID, Jacketed_Cstr} = \mathbf{N}_{ACETIC_ACID, Vessel}$$

2. Unit Vessel species ACETIC_ACID balance

$$\partial \mathbf{N}_{ACETIC_ACID, Vessel} / \partial t = (\mathbf{n}_{ACETIC_ACID, reactants_input, source} - \mathbf{n}_{ACETIC_ACID, products_output, source}) + (-ext_rxn0_{Vessel, liq0})$$

3. Unit Vessel species ACETIC_ACID mole quantity decomposition

$$\mathbf{N}_{ACETIC_ACID, Vessel} = \mathbf{N}_{ACETIC_ACID, Vessel, liq0}$$

4. Unit Jacketed_Cstr species WATER mole quantity decomposition

$$\mathbf{N}_{WATER, Jacketed_Cstr} = \mathbf{N}_{WATER, Vessel} + \mathbf{N}_{WATER, Jacket}$$

5. Unit Vessel species WATER balance

$$\partial \mathbf{N}_{WATER, Vessel} / \partial t = (\mathbf{n}_{WATER, reactants_input, source} - \mathbf{n}_{WATER, products_output, source}) + ext_rxn0_{Vessel, liq0}$$

6. Unit Vessel species WATER mole quantity decomposition

$$\mathbf{N}_{WATER, Vessel} = \mathbf{N}_{WATER, Vessel, liq0}$$

7. Unit Jacket species WATER balance

$$\partial \mathbf{N}_{WATER, Jacket} / \partial t = \mathbf{n}_{WATER, coolant_inlet, source} - \mathbf{n}_{WATER, coolant_outlet, source}$$

8. Unit Jacket species WATER mole quantity decomposition

$$\mathbf{N}_{WATER, Jacket} = \mathbf{N}_{WATER, Jacket, liq0}$$

9. Unit Jacketed_Cstr species 1_BUTANOL mole quantity decomposition

$$\mathbf{N}_{1_BUTANOL, Jacketed_Cstr} = \mathbf{N}_{1_BUTANOL, Vessel}$$

10. Unit Vessel species 1_BUTANOL balance

$$\partial \mathbf{N}_{1_BUTANOL, Vessel} / \partial t = (\mathbf{n}_{1_BUTANOL, reactants_input, source} - \mathbf{n}_{1_BUTANOL, products_output, source}) + (-ext_rxn0_{Vessel, liq0})$$

11. Unit Vessel species 1_BUTANOL mole quantity decomposition

$$\mathbf{N}_{1_BUTANOL, Vessel} = \mathbf{N}_{1_BUTANOL, Vessel, liq0}$$

12. Unit Jacketed_Cstr species n_BUTYL_ACETATE mole quantity decomposition

$$\mathbf{N}_{n_BUTYL_ACETATE, Jacketed_Cstr} = \mathbf{N}_{n_BUTYL_ACETATE, Vessel}$$

13. Unit Vessel species n_BUTYL_ACETATE balance

$$\partial N_{n_BUTYL_ACETATE, Vessel} / \partial t = (n_{n_BUTYL_ACETATE, reactants_input, source} - n_{n_BUTYL_ACETATE, products_output, source}) + ext_rxn0_{Vessel, liq0}$$

14. Unit Vessel species n_BUTYL_ACETATE mole quantity decomposition

$$N_{n_BUTYL_ACETATE, Vessel} = N_{n_BUTYL_ACETATE, Vessel, liq0}$$

15. Phase Vesselmatl_L0 total moles decomposition into individual species moles

$$N_{Vessel, liq0} = ((N_{ACETIC_ACID, Vessel, liq0} + N_{WATER, Vessel, liq0}) + N_{1_BUTANOL, Vessel, liq0}) + N_{n_BUTYL_ACETATE, Vessel, liq0}$$

16. Phase Vesselmatl_L0 sum of individual species mole fractions

$$((x_{ACETIC_ACID, Vessel, liq0} + x_{WATER, Vessel, liq0}) + x_{1_BUTANOL, Vessel, liq0}) + x_{n_BUTYL_ACETATE, Vessel, liq0} = 1$$

17. Phase Vesselmatl_L0 species ACETIC_ACID molar concentration

$$N_{ACETIC_ACID, Vessel, liq0} = c_{ACETIC_ACID, Vessel, liq0} * V_{Vessel, liq0}$$

18. Phase Vesselmatl_L0 species WATER mole fraction

$$N_{WATER, Vessel, liq0} = x_{WATER, Vessel, liq0} * N_{Vessel, liq0}$$

19. Phase Vesselmatl_L0 species WATER molar concentration

$$N_{WATER, Vessel, liq0} = c_{WATER, Vessel, liq0} * V_{Vessel, liq0}$$

20. Phase Vesselmatl_L0 species 1_BUTANOL mole fraction

$$N_{1_BUTANOL, Vessel, liq0} = x_{1_BUTANOL, Vessel, liq0} * N_{Vessel, liq0}$$

21. Phase Vesselmatl_L0 species 1_BUTANOL molar concentration

$$N_{1_BUTANOL, Vessel, liq0} = c_{1_BUTANOL, Vessel, liq0} * V_{Vessel, liq0}$$

22. Phase Vesselmatl_L0 species n_BUTYL_ACETATE mole fraction

$$N_{n_BUTYL_ACETATE, Vessel, liq0} = x_{n_BUTYL_ACETATE, Vessel, liq0} * N_{Vessel, liq0}$$

23. Phase Vesselmatl_L0 species n_BUTYL_ACETATE molar concentration

$$N_{n_BUTYL_ACETATE, Vessel, liq0} = c_{n_BUTYL_ACETATE, Vessel, liq0} * V_{Vessel, liq0}$$

24. Phase Jacketmatl_L0 total moles decomposition into individual species moles

$$N_{Jacket, liq0} = N_{WATER, Jacket, liq0}$$

25. Phase Jacketmatl_L0 sum of individual species mole fractions

$$x_{WATER, Jacket, liq0} = 1$$

26. Phase Jacketmatl_L0 species WATER molar concentration

$$N_{WATER, Jacket, liq0} = c_{WATER, Jacket, liq0} * V_{Jacket, liq0}$$

27. Unit Jacketed_Cstr reaction rxn0 extent decomposition

$$ext_rxn0_{Jacketed_Cstr} = ext_rxn0_{Vessel}$$

28. Unit Vessel reaction rxn0 extent decomposition

$$ext_rxn0_{Vessel} = ext_rxn0_{Vessel, liq0}$$

29. Flux reactants_input molar density

$$n_{tot, reactants_input, source} = rho_{n, reactants_input} * v_{reactants_input, source}$$

30. Phase reactants_input_phase molecular weight from PPM

$$\text{mw}_{\text{reactants_input}} = (((\mathbf{x}_{\text{ACETIC_ACID, reactants_input}} * 60.0526) + (\mathbf{x}_{\text{WATER, reactants_input}} * 18.0153)) + (\mathbf{x}_{\text{1_BUTANOL, reactants_input}} * 74.1228)) + (\mathbf{x}_{\text{n_BUTYL_ACETATE, reactants_input}} * 116.16)$$

31. Flux reactants_input sum of species mole fractions

$$((\mathbf{x}_{\text{ACETIC_ACID, reactants_input}} + \mathbf{x}_{\text{WATER, reactants_input}}) + \mathbf{x}_{\text{1_BUTANOL, reactants_input}}) + \mathbf{x}_{\text{n_BUTYL_ACETATE, reactants_input}} = 1$$

32. Phase reactants_input_phase density correlation from PPM

$$\rho_{\text{n, reactants_input}} = F(\mathbf{T}_{\text{reactants_input}}, \mathbf{x}_{\text{ACETIC_ACID, reactants_input}}, \mathbf{x}_{\text{WATER, reactants_input}}, \mathbf{x}_{\text{1_BUTANOL, reactants_input}}, \mathbf{x}_{\text{n_BUTYL_ACETATE, reactants_input}})$$

33. Flux reactants_input species ACETIC_ACID mole fraction

$$\mathbf{n}_{\text{ACETIC_ACID, reactants_input, source}} = \mathbf{x}_{\text{ACETIC_ACID, reactants_input}} * \mathbf{n}_{\text{tot, reactants_input, source}}$$

34. Flux reactants_input species WATER mole fraction

$$\mathbf{n}_{\text{WATER, reactants_input, source}} = \mathbf{x}_{\text{WATER, reactants_input}} * \mathbf{n}_{\text{tot, reactants_input, source}}$$

35. Flux reactants_input species 1_BUTANOL mole fraction

$$\mathbf{n}_{\text{1_BUTANOL, reactants_input, source}} = \mathbf{x}_{\text{1_BUTANOL, reactants_input}} * \mathbf{n}_{\text{tot, reactants_input, source}}$$

36. Flux reactants_input species n_BUTYL_ACETATE mole fraction

$$\mathbf{n}_{\text{n_BUTYL_ACETATE, reactants_input, source}} = \mathbf{x}_{\text{n_BUTYL_ACETATE, reactants_input}} * \mathbf{n}_{\text{tot, reactants_input, source}}$$

37. Flux products_output molar density

$$\mathbf{n}_{\text{tot, products_output, source}} = \rho_{\text{n, Vessel, liq0}} * \mathbf{v}_{\text{products_output, source}}$$

38. Flux products_output species ACETIC_ACID mole fraction

$$\mathbf{n}_{\text{ACETIC_ACID, products_output, source}} = \mathbf{x}_{\text{ACETIC_ACID, Vessel, liq0}} * \mathbf{n}_{\text{tot, products_output, source}}$$

39. Flux products_output species WATER mole fraction

$$\mathbf{n}_{\text{WATER, products_output, source}} = \mathbf{x}_{\text{WATER, Vessel, liq0}} * \mathbf{n}_{\text{tot, products_output, source}}$$

40. Flux products_output species 1_BUTANOL mole fraction

$$\mathbf{n}_{\text{1_BUTANOL, products_output, source}} = \mathbf{x}_{\text{1_BUTANOL, Vessel, liq0}} * \mathbf{n}_{\text{tot, products_output, source}}$$

41. Flux products_output species n_BUTYL_ACETATE mole fraction

$$\mathbf{n}_{\text{n_BUTYL_ACETATE, products_output, source}} = \mathbf{x}_{\text{n_BUTYL_ACETATE, Vessel, liq0}} * \mathbf{n}_{\text{tot, products_output, source}}$$

42. Flux coolant_inlet molar density

$$\mathbf{n}_{\text{tot, coolant_inlet, source}} = \rho_{\text{n, coolant_inlet}} * \mathbf{v}_{\text{coolant_inlet, source}}$$

43. Phase coolant_inlet_phase molecular weight from PPM

$$\text{mw}_{\text{coolant_inlet}} = 18.0153$$

44. Flux coolant_inlet sum of species mole fractions

$$\mathbf{x}_{\text{WATER, coolant_inlet}} = 1$$

45. Phase coolant_inlet_phase density correlation from PPM

$$\rho_{\text{n, coolant_inlet}} = F(\mathbf{T}_{\text{coolant_inlet}})$$

46. Flux coolant_inlet species WATER mole fraction

$$\mathbf{n}_{\text{WATER, coolant_inlet, source}} = \mathbf{x}_{\text{WATER, coolant_inlet}} * \mathbf{n}_{\text{tot, coolant_inlet, source}}$$

47. Flux coolant_outlet molar density

$$\mathbf{n}_{\text{tot, coolant_outlet, source}} = \rho_{\mathbf{n, Jacket, liq0}} * \mathbf{v}_{\text{coolant_outlet, source}}$$

48. Flux coolant_outlet species WATER mole fraction

$$\mathbf{n}_{\text{WATER, coolant_outlet, source}} = \mathbf{x}_{\text{WATER, Jacket, liq0}} * \mathbf{n}_{\text{tot, coolant_outlet, source}}$$

49. Unit Vessel energy balance

$$\partial \mathbf{U}_{\text{Vessel}} / \partial t = (\mathbf{e}_{\text{reactants_input, source}} - \mathbf{e}_{\text{products_output, source}}) - \mathbf{e}_{\text{q_exchange, source}}$$

50. Unit Jacket energy balance

$$\partial \mathbf{U}_{\text{Jacket}} / \partial t = (\mathbf{e}_{\text{coolant_inlet, source}} - \mathbf{e}_{\text{coolant_outlet, source}}) + \mathbf{e}_{\text{q_exchange, source}}$$

51. Flux reactants_input energy flux

$$\mathbf{e}_{\text{reactants_input, source}} = \mathbf{h}_{\text{reactants_input}} * \mathbf{n}_{\text{tot, reactants_input, source}}$$

52. Flux products_output energy flux

$$\mathbf{e}_{\text{products_output, source}} = \mathbf{h}_{\text{Vessel, liq0}} * \mathbf{n}_{\text{tot, products_output, source}}$$

53. Flux coolant_inlet energy flux

$$\mathbf{e}_{\text{coolant_inlet, source}} = \mathbf{h}_{\text{coolant_inlet}} * \mathbf{n}_{\text{tot, coolant_inlet, source}}$$

54. Flux coolant_outlet energy flux

$$\mathbf{e}_{\text{coolant_outlet, source}} = \mathbf{h}_{\text{Jacket, liq0}} * \mathbf{n}_{\text{tot, coolant_outlet, source}}$$

55. Flux q_exchange energy flux

$$\mathbf{e}_{\text{q_exchange, source}} = \mathbf{A}_{\text{q_exchange}} * (\mathbf{U}_{\text{o, q_exchange}} * (\mathbf{T}_{\text{Vessel}} - \mathbf{T}_{\text{Jacket}}))$$

56. Phase Vesselmatl_L0 reaction rxn0 forward rate

$$r_{\text{rxn0_forw}}_{\text{Vessel, liq0}} = (k_{\text{constforw}}_{\text{rxn0}} * (\mathbf{c}_{\text{ACETIC_ACID, Vessel, liq0}})^2) * (\mathbf{c}_{\text{1_BUTANOL, Vessel, liq0}})$$

57. Phase Vesselmatl_L0 reaction rxn0 reverse rate

$$r_{\text{rxn0_back}}_{\text{Vessel, liq0}} = (k_{\text{constback}}_{\text{rxn0}} * (\mathbf{c}_{\text{WATER, Vessel, liq0}})) * (\mathbf{c}_{\text{n_BUTYL_ACETATE, Vessel, liq0}})$$

58. Phase Vesselmatl_L0 reaction rxn0 extent

$$\text{ext_rxn0}_{\text{Vessel, liq0}} = (r_{\text{rxn0_forw}}_{\text{Vessel, liq0}} - r_{\text{rxn0_back}}_{\text{Vessel, liq0}}) * \mathbf{V}_{\text{Vessel, liq0}}$$

59. Phase Vesselmatl_L0 molar density

$$\mathbf{N}_{\text{Vessel, liq0}} = \rho_{\mathbf{n, Vessel, liq0}} * \mathbf{V}_{\text{Vessel, liq0}}$$

60. Phase Jacketmatl_L0 molar density

$$\mathbf{N}_{\text{Jacket, liq0}} = \rho_{\mathbf{n, Jacket, liq0}} * \mathbf{V}_{\text{Jacket, liq0}}$$

61. Unit Vessel extensive variable V_Vesselmatl decomposition

$$\mathbf{V}_{\text{Vessel}} = \mathbf{V}_{\text{Vessel, liq0}}$$

62. Unit Jacket extensive variable V_Jacketmatl decomposition

$$\mathbf{V}_{\text{Jacket}} = \mathbf{V}_{\text{Jacket, liq0}}$$

63. Phase Vesselmatl_L0 density correlation from PPM

$$\rho_{\mathbf{n, Vessel, liq0}} = F(\mathbf{T}_{\text{Vessel}}, \mathbf{x}_{\text{ACETIC_ACID, Vessel, liq0}}, \mathbf{x}_{\text{WATER, Vessel, liq0}}, \mathbf{x}_{\text{1_BUTANOL, Vessel, liq0}}, \mathbf{x}_{\text{n_BUTYL_ACETATE, Vessel, liq0}})$$

64. Phase Jacketmatl_L0 density correlation from PPM

$$\rho_{\mathbf{n, Jacket, liq0}} = F(\mathbf{T}_{\text{Jacket}})$$

65. Unit Vessel extensive variable Un_Vesselmatl decomposition

$$\mathbf{U}_{\text{Vessel}} = (\mathbf{N}_{\text{Vessel, liq0}} * \mathbf{h}_{\text{Vessel, liq0}}) - (\mathbf{P}_{\text{Vessel}} * \mathbf{V}_{\text{Vessel}})$$

66. Phase Vesselmatl_L0 enthalpy correlation from PPM

$$\mathbf{h}_{\text{Vessel, liq0}} = F(\mathbf{T}_{\text{Vessel}}, \mathbf{x}_{\text{ACETIC_ACID, Vessel, liq0}}, \mathbf{x}_{\text{WATER, Vessel, liq0}}, \mathbf{x}_{\text{I_BUTANOL, Vessel, liq0}}, \mathbf{x}_{\text{n_BUTYL_ACETATE, Vessel, liq0}})$$

67. Phase Vesselmatl_L0 heat capacity correlation from PPM

$$\mathbf{C}_{\text{p n, Vessel, liq0}} = F(\mathbf{T}_{\text{Vessel}}, \mathbf{x}_{\text{ACETIC_ACID, Vessel, liq0}}, \mathbf{x}_{\text{WATER, Vessel, liq0}}, \mathbf{x}_{\text{I_BUTANOL, Vessel, liq0}}, \mathbf{x}_{\text{n_BUTYL_ACETATE, Vessel, liq0}})$$

68. Unit Jacket extensive variable Un_Jacketmatl decomposition

$$\mathbf{U}_{\text{Jacket}} = (\mathbf{N}_{\text{Jacket, liq0}} * \mathbf{h}_{\text{Jacket, liq0}}) - (\mathbf{P}_{\text{Jacket}} * \mathbf{V}_{\text{Jacket}})$$

69. Phase Jacketmatl_L0 enthalpy correlation from PPM

$$\mathbf{h}_{\text{Jacket, liq0}} = F(\mathbf{T}_{\text{Jacket}})$$

70. Phase Jacketmatl_L0 heat capacity correlation from PPM

$$\mathbf{C}_{\text{p n, Jacket, liq0}} = F(\mathbf{T}_{\text{Jacket}})$$

71. Connection reactants_input phase reactants_input_phase enthalpy correlation from PPM

$$\mathbf{h}_{\text{reactants_input}} = F(\mathbf{T}_{\text{reactants_input}}, \mathbf{x}_{\text{ACETIC_ACID, reactants_input}}, \mathbf{x}_{\text{WATER, reactants_input}}, \mathbf{x}_{\text{I_BUTANOL, reactants_input}}, \mathbf{x}_{\text{n_BUTYL_ACETATE, reactants_input}})$$

72. Connection coolant_inlet phase coolant_inlet_phase enthalpy correlation from PPM

$$\mathbf{h}_{\text{coolant_inlet}} = F(\mathbf{T}_{\text{coolant_inlet}})$$

73. Phase Vesselmatl_L0 molecular weight from PPM

$$\mathbf{mw}_{\text{Vessel, liq0}} = (((\mathbf{x}_{\text{ACETIC_ACID, Vessel, liq0}} * 60.0526) + (\mathbf{x}_{\text{WATER, Vessel, liq0}} * 18.0153)) + (\mathbf{x}_{\text{I_BUTANOL, Vessel, liq0}} * 74.1228)) + (\mathbf{x}_{\text{n_BUTYL_ACETATE, Vessel, liq0}} * 116.16)$$

74. Phase Jacketmatl_L0 molecular weight from PPM

$$\mathbf{mw}_{\text{Jacket, liq0}} = 18.0153$$

Appendix E

2-D Spatially Distributed Tubular Reactor

Model Equations

1. Flux flow_z mole flux per area

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}, \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{n}_{\text{area, flow_z, source}}(r, z) = \rho_{\mathbf{n}, \text{Tube_reactor_rz, vap0}}(r, z) * \mathbf{v}_{\mathbf{z}, \text{flow_z}}(r, z)$$

2. Unit Tube_reactor_rz species o_XYLENE balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}|+, \mathbf{r}_{\max, \text{Tube_reactor}}|-); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}|+, \mathbf{z}_{\max, \text{Tube_reactor}}|-)$$
$$\partial \mathbf{c}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z) / \partial t = (((-\partial \mathbf{n}_{\text{area, o_XYLENE, flow_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) - (\partial \mathbf{n}_{\text{area, o_XYLENE, a_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) - ((1/r) * (\partial (\mathbf{n}_{\text{area, o_XYLENE, a_r, source}}(r, z) * r) / \partial \mathbf{r}_{\text{Tube_reactor}}))) + (-r_{\text{rxn0_forw_Tube_reactor_rzmatl_V0}}_{\text{Tube_reactor_rz, vap0}}(r, z))$$

3. Unit Tube_reactor_r1 boundary species o_XYLENE balance

$$\mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{n}_{\text{area, o_XYLENE, a_r}}(r_{\text{coord_min_Tube_reactor}}, z) = 0$$

4. Unit Tube_reactor_r2 boundary species o_XYLENE balance

$$\mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{n}_{\text{area, o_XYLENE, a_r}}(r_{\text{coord_max_Tube_reactor}}, z) = 0$$

5. Unit Tube_reactor_z1 boundary species o_XYLENE balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}|+, \mathbf{r}_{\max, \text{Tube_reactor}}|-)$$
$$\mathbf{n}_{\text{area, o_XYLENE, flow_z}}(r, z_{\text{coord_min_Tube_reactor}}) + \mathbf{n}_{\text{area, o_XYLENE, a_z}}(r, z_{\text{coord_min_Tube_reactor}}) = \mathbf{n}_{\text{area, o_XYLENE, reactants, source}}(r)$$

6. Unit Tube_reactor_z2 boundary species o_XYLENE balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}, \mathbf{r}_{\max, \text{Tube_reactor}})$$
$$\mathbf{n}_{\text{area, o_XYLENE, flow_z}}(r, z_{\text{coord_max_Tube_reactor}}) + \mathbf{n}_{\text{area, o_XYLENE, a_z}}(r, z_{\text{coord_max_Tube_reactor}}) = \mathbf{n}_{\text{area, o_XYLENE, products, source}}(r)$$

7. Flux a_z species o_XYLENE flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}, \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{n}_{\text{area, o_XYLENE, a_z, source}}(r, z) = (-\mathbf{D}_{\text{f, a_z}}(r, z) * (\partial \mathbf{c}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}}))$$

8. Flux a_r species o_XYLENE flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}, \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{n}_{\text{area, o_XYLENE, a_r, source}}(r, z) = (-\mathbf{D}_{\text{f, a_r}}(r, z) * (\partial \mathbf{c}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z) / \partial \mathbf{r}_{\text{Tube_reactor}}))$$

9. Unit Tube_reactor_rz species PHTHALIC_ANHYDRIDE balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}|+, \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\partial \mathbf{c}_{\text{PHTHALIC_ANHYDRIDE, Tube_reactor_rz, vap0}}(r, z) / \partial t = (-\partial \mathbf{n}_{\text{area, PHTHALIC_ANHYDRIDE, flow_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) + r_{\text{rxn0_forw_Tube_reactor_rzmatl_V0}}_{\text{Tube_reactor_rz, vap0}}(r, z)$$

10. Unit Tube_reactor_z1 boundary species PHTHALIC_ANHYDRIDE balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}})$$

$$\mathbf{n}_{\text{area, PHTHALIC_ANHYDRIDE, flow_z}}(r, z_{\text{coord_min_Tube_reactor}}) = \mathbf{n}_{\text{area, PHTHALIC_ANHYDRIDE, reactants, source}}(r)$$

11. Unit Tube_reactor_z2 boundary species PHTHALIC_ANHYDRIDE balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}})$$

$$\mathbf{n}_{\text{area, PHTHALIC_ANHYDRIDE, flow_z}}(r, z_{\text{coord_max_Tube_reactor}}) = \mathbf{n}_{\text{area, PHTHALIC_ANHYDRIDE, products, source}}(r)$$

12. Unit Tube_reactor_rz species OXYGEN balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}|+, \mathbf{r}_{\max, \text{Tube_reactor}}|-); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}|+, \mathbf{z}_{\max, \text{Tube_reactor}}|-)$$

$$\partial \mathbf{c}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z) / \partial t = (((-\partial \mathbf{n}_{\text{area, OXYGEN, flow_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) - (\partial \mathbf{n}_{\text{area, OXYGEN, b_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) - ((1/r) * (\partial (\mathbf{n}_{\text{area, OXYGEN, b_r, source}}(r, z) * r) / \partial r_{\text{Tube_reactor}}))) + (-3 * r_{\text{rxn0_forw_Tube_reactor_rzmatl_V0}}_{\text{Tube_reactor_rz, vap0}}(r, z))$$

13. Unit Tube_reactor_r1 boundary species OXYGEN balance

$$\mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{n}_{\text{area, OXYGEN, b_r}}(r_{\text{coord_min_Tube_reactor}}, z) = 0$$

14. Unit Tube_reactor_r2 boundary species OXYGEN balance

$$\mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{n}_{\text{area, OXYGEN, b_r}}(r_{\text{coord_max_Tube_reactor}}, z) = 0$$

15. Unit Tube_reactor_z1 boundary species OXYGEN balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}|+, \mathbf{r}_{\max, \text{Tube_reactor}}|-)$$

$$\mathbf{n}_{\text{area, OXYGEN, flow_z}}(r, z_{\text{coord_min_Tube_reactor}}) + \mathbf{n}_{\text{area, OXYGEN, b_z}}(r, z_{\text{coord_min_Tube_reactor}}) = \mathbf{n}_{\text{area, OXYGEN, reactants, source}}(r)$$

16. Unit Tube_reactor_z2 boundary species OXYGEN balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}})$$

$$\mathbf{n}_{\text{area, OXYGEN, flow_z}}(r, z_{\text{coord_max_Tube_reactor}}) + \mathbf{n}_{\text{area, OXYGEN, b_z}}(r, z_{\text{coord_max_Tube_reactor}}) = \mathbf{n}_{\text{area, OXYGEN, products, source}}(r)$$

17. Flux b_z species OXYGEN flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{n}_{\text{area, OXYGEN, b_z, source}}(r, z) = (-\mathbf{D}_{f, b_z}(r, z) * (\partial \mathbf{c}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}}))$$

18. Flux b_r species OXYGEN flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{n}_{\text{area, OXYGEN, b_r, source}}(r, z) = (-\mathbf{D}_{f, b_r}(r, z) * (\partial \mathbf{c}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z) / \partial r_{\text{Tube_reactor}}))$$

19. Unit Tube_reactor_rz species WATER balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}|+, \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\partial \mathbf{c}_{\text{WATER, Tube_reactor_rz, vap0}}(r, z) / \partial t = (-\partial \mathbf{n}_{\text{area, WATER, flow_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) + (3 * r_{\text{rxn0_forw_Tube_reactor_rzmatl_V0}}_{\text{Tube_reactor_rz, vap0}}(r, z))$$

20. Unit Tube_reactor_z1 boundary species WATER balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}})$$

$$\mathbf{n}_{\text{area, WATER, flow_z}}(r, z_{\text{coord_min_Tube_reactor}}) = \mathbf{n}_{\text{area, WATER, reactants, source}}(r)$$

21. Unit Tube_reactor_z2 boundary species WATER balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \mathbf{r}_{\max, \text{Tube_reactor}})$$
$$\mathbf{n}_{\text{area, WATER, flow_z}}(r, z_{\text{coord_max_Tube_reactor}}) = \mathbf{n}_{\text{area, WATER, products, source}}(r)$$

22. Unit Jacket species WATER balance

$$\partial \mathbf{N}_{\text{WATER, Jacket}} / \partial t = \mathbf{n}_{\text{WATER, cool_in, source}} - \mathbf{n}_{\text{WATER, cool_out, source}}$$

23. Unit Jacket species WATER mole quantity decomposition

$$\mathbf{N}_{\text{WATER, Jacket}} = \mathbf{N}_{\text{WATER, Jacket, liq0}}$$

24. Phase Tube_reactor_rzmatl_V0 sum of individual species mole fractions

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$((\mathbf{x}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z) + \mathbf{x}_{\text{PHTHALIC_ANHYDRIDE, Tube_reactor_rz, vap0}}(r, z)) + \mathbf{x}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z)) + \mathbf{x}_{\text{WATER, Tube_reactor_rz, vap0}}(r, z) = 1$$

25. Phase Tube_reactor_rzmatl_V0 species o_XYLENE molar concentration

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{c}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z) = \mathbf{x}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z) * \mathbf{rho}_{\mathbf{n}, \text{Tube_reactor_rz, vap0}}(r, z)$$

26. Phase Tube_reactor_rzmatl_V0 species PHTHALIC_ANHYDRIDE molar concentration

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{c}_{\text{PHTHALIC_ANHYDRIDE, Tube_reactor_rz, vap0}}(r, z) = \mathbf{x}_{\text{PHTHALIC_ANHYDRIDE, Tube_reactor_rz, vap0}}(r, z) * \mathbf{rho}_{\mathbf{n}, \text{Tube_reactor_rz, vap0}}(r, z)$$

27. Phase Tube_reactor_rzmatl_V0 species OXYGEN molar concentration

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{c}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z) = \mathbf{x}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z) * \mathbf{rho}_{\mathbf{n}, \text{Tube_reactor_rz, vap0}}(r, z)$$

28. Phase Tube_reactor_rzmatl_V0 species WATER molar concentration

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{c}_{\text{WATER, Tube_reactor_rz, vap0}}(r, z) = \mathbf{x}_{\text{WATER, Tube_reactor_rz, vap0}}(r, z) * \mathbf{rho}_{\mathbf{n}, \text{Tube_reactor_rz, vap0}}(r, z)$$

29. Phase Jacketmatl_L0 total moles decomposition into individual species moles

$$\mathbf{N}_{\text{Jacket, liq0}} = \mathbf{N}_{\text{WATER, Jacket, liq0}}$$

30. Phase Jacketmatl_L0 sum of individual species mole fractions

$$\mathbf{x}_{\text{WATER, Jacket, liq0}} = 1$$

31. Phase Jacketmatl_L0 species WATER molar concentration

$$\mathbf{N}_{\text{WATER, Jacket, liq0}} = \mathbf{c}_{\text{WATER, Jacket, liq0}} * \mathbf{V}_{\text{Jacket, liq0}}$$

32. Phase reactants_phase molecular weight from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \mathbf{r}_{\max, \text{Tube_reactor}})$$
$$\mathbf{mw}_{\text{reactants}}(r) = (((\mathbf{x}_{\text{o_XYLENE, reactants}}(r) * 106.167) + (\mathbf{x}_{\text{PHTHALIC_ANHYDRIDE, reactants}}(r) * 148.118)) + (\mathbf{x}_{\text{OXYGEN, reactants}}(r) * 31.9988)) + (\mathbf{x}_{\text{WATER, reactants}}(r) * 18.0153))$$

33. Flux reactants sum of species mole fractions

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \mathbf{r}_{\max, \text{Tube_reactor}})$$
$$((\mathbf{x}_{\text{o_XYLENE, reactants}}(r) + \mathbf{x}_{\text{PHTHALIC_ANHYDRIDE, reactants}}(r)) + \mathbf{x}_{\text{OXYGEN, reactants}}(r)) + \mathbf{x}_{\text{WATER, reactants}}(r) = 1$$

34. Flux reactants species o_XYLENE mole fraction

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \mathbf{r}_{\max, \text{Tube_reactor}})$$

$$\mathbf{n}_{\text{area, o_XYLENE, reactants, source}}(r) = \mathbf{x}_{\text{o_XYLENE, reactants}}(r) * \mathbf{n}_{\text{area, reactants, source}}(r)$$

35. Flux reactants species PHTHALIC_ANHYDRIDE mole fraction

$$\mathbf{r} := (\mathbf{r}_{\text{min, Tube_reactor}}, \mathbf{r}_{\text{max, Tube_reactor}})$$

$$\mathbf{n}_{\text{area, PHTHALIC_ANHYDRIDE, reactants, source}}(r) = \mathbf{x}_{\text{PHTHALIC_ANHYDRIDE, reactants}}(r) * \mathbf{n}_{\text{area, reactants, source}}(r)$$

36. Flux reactants species OXYGEN mole fraction

$$\mathbf{r} := (\mathbf{r}_{\text{min, Tube_reactor}}, \mathbf{r}_{\text{max, Tube_reactor}})$$

$$\mathbf{n}_{\text{area, OXYGEN, reactants, source}}(r) = \mathbf{x}_{\text{OXYGEN, reactants}}(r) * \mathbf{n}_{\text{area, reactants, source}}(r)$$

37. Flux reactants species WATER mole fraction

$$\mathbf{r} := (\mathbf{r}_{\text{min, Tube_reactor}}, \mathbf{r}_{\text{max, Tube_reactor}})$$

$$\mathbf{n}_{\text{area, WATER, reactants, source}}(r) = \mathbf{x}_{\text{WATER, reactants}}(r) * \mathbf{n}_{\text{area, reactants, source}}(r)$$

38. Flux products total mole flux decomposition into individual species mole fluxes

$$\mathbf{r} := (\mathbf{r}_{\text{min, Tube_reactor}}, \mathbf{r}_{\text{max, Tube_reactor}})$$

$$\mathbf{n}_{\text{area, products, source}}(r) = ((\mathbf{n}_{\text{area, o_XYLENE, products, source}}(r) + \mathbf{n}_{\text{area, PHTHALIC_ANHYDRIDE, products, source}}(r)) + \mathbf{n}_{\text{area, OXYGEN, products, source}}(r)) + \mathbf{n}_{\text{area, WATER, products, source}}(r)$$

39. Flux flow_z differential species o_XYLENE mole flux

$$\mathbf{r} := (\mathbf{r}_{\text{min, Tube_reactor}}, \mathbf{r}_{\text{max, Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\text{min, Tube_reactor}}, \mathbf{z}_{\text{max, Tube_reactor}})$$

$$\mathbf{n}_{\text{area, o_XYLENE, flow_z, source}}(r, z) = \mathbf{c}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z) * \mathbf{v}_{\text{z, flow_z}}(r, z)$$

40. Flux flow_z differential species PHTHALIC_ANHYDRIDE mole flux

$$\mathbf{r} := (\mathbf{r}_{\text{min, Tube_reactor}}, \mathbf{r}_{\text{max, Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\text{min, Tube_reactor}}, \mathbf{z}_{\text{max, Tube_reactor}})$$

$$\mathbf{n}_{\text{area, PHTHALIC_ANHYDRIDE, flow_z, source}}(r, z) = \mathbf{c}_{\text{PHTHALIC_ANHYDRIDE, Tube_reactor_rz, vap0}}(r, z) * \mathbf{v}_{\text{z, flow_z}}(r, z)$$

41. Flux flow_z differential species OXYGEN mole flux

$$\mathbf{r} := (\mathbf{r}_{\text{min, Tube_reactor}}, \mathbf{r}_{\text{max, Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\text{min, Tube_reactor}}, \mathbf{z}_{\text{max, Tube_reactor}})$$

$$\mathbf{n}_{\text{area, OXYGEN, flow_z, source}}(r, z) = \mathbf{c}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z) * \mathbf{v}_{\text{z, flow_z}}(r, z)$$

42. Flux flow_z differential species WATER mole flux

$$\mathbf{r} := (\mathbf{r}_{\text{min, Tube_reactor}}, \mathbf{r}_{\text{max, Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\text{min, Tube_reactor}}, \mathbf{z}_{\text{max, Tube_reactor}})$$

$$\mathbf{n}_{\text{area, WATER, flow_z, source}}(r, z) = \mathbf{c}_{\text{WATER, Tube_reactor_rz, vap0}}(r, z) * \mathbf{v}_{\text{z, flow_z}}(r, z)$$

43. Phase cool_in phase molecular weight from PPM

$$\mathbf{mw}_{\text{cool_in}} = 18.0153$$

44. Flux cool_in sum of species mole fractions

$$\mathbf{x}_{\text{WATER, cool_in}} = 1$$

45. Flux cool_in species WATER mole fraction

$$\mathbf{n}_{\text{WATER, cool_in, source}} = \mathbf{x}_{\text{WATER, cool_in}} * \mathbf{n}_{\text{tot, cool_in, source}}$$

46. Flux cool_out species WATER mole fraction

$$\mathbf{n}_{\text{WATER, cool_out, source}} = \mathbf{x}_{\text{WATER, Jacket, liq0}} * \mathbf{n}_{\text{tot, cool_out, source}}$$

47. Unit Tube_reactor_rz energy balance

$$\mathbf{r} := (\mathbf{r}_{\text{min, Tube_reactor}}|+, \mathbf{r}_{\text{max, Tube_reactor}}|-); \mathbf{z} := (\mathbf{z}_{\text{min, Tube_reactor}}|+, \mathbf{z}_{\text{max, Tube_reactor}}|-)$$

$$\mathbf{rho}_{\text{n, Tube_reactor_rz, vap0}}(r, z) * \partial \mathbf{u}_{\text{Tube_reactor_rz, vap0}}(r, z) / \partial t = ((((((-(\partial \mathbf{e}_{\text{area, flow_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) - (\partial \mathbf{e}_{\text{area, a_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) - (\partial \mathbf{e}_{\text{area, b_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) - (\partial \mathbf{e}_{\text{area, q_z, source}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}})) - ((1/r) * (\partial (\mathbf{e}_{\text{area, a_r, source}}(r, z) * \mathbf{r}) / \partial \mathbf{r}_{\text{Tube_reactor}})) - ((1/r) * (\partial (\mathbf{e}_{\text{area, b_r, source}}(r, z) * \mathbf{r}) / \partial \mathbf{r}_{\text{Tube_reactor}})) - ((1/r) * (\partial (\mathbf{e}_{\text{area, q_r, source}}(r, z) * \mathbf{r}) / \partial \mathbf{r}_{\text{Tube_reactor}})) - ((1/r) * (\partial (\mathbf{e}_{\text{area, q_r, source}}(r, z) * \mathbf{r}) / \partial \mathbf{r}_{\text{Tube_reactor}}))$$

48. Unit Tube_reactor_r1 boundary energy balance

$$\mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$(\mathbf{e}_{\text{area, a_r}}(r_coord_min_Tube_reactor, z) + \mathbf{e}_{\text{area, b_r}}(r_coord_min_Tube_reactor, z)) + \mathbf{e}_{\text{area, q_r}}(r_coord_min_Tube_reactor, z) = 0$$

49. Unit Tube_reactor_r2 boundary energy balance

$$\mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$(\mathbf{e}_{\text{area, a_r}}(r_coord_max_Tube_reactor, z) + \mathbf{e}_{\text{area, b_r}}(r_coord_max_Tube_reactor, z)) + \mathbf{e}_{\text{area, q_r}}(r_coord_max_Tube_reactor, z) = \mathbf{e}_{\text{area, q, source}}(z)$$

50. Unit Tube_reactor_z1 boundary energy balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}^{|+}, \ \mathbf{r}_{\max, \text{Tube_reactor}}^{|-})$$
$$((\mathbf{e}_{\text{area, flow_z}}(r, z_coord_min_Tube_reactor) + \mathbf{e}_{\text{area, a_z}}(r, z_coord_min_Tube_reactor)) + \mathbf{e}_{\text{area, b_z}}(r, z_coord_min_Tube_reactor)) + \mathbf{e}_{\text{area, q_z}}(r, z_coord_min_Tube_reactor) = \mathbf{e}_{\text{area, reactants, source}}(r)$$

51. Unit Tube_reactor_z2 boundary energy balance

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}})$$
$$((\mathbf{e}_{\text{area, flow_z}}(r, z_coord_max_Tube_reactor) + \mathbf{e}_{\text{area, a_z}}(r, z_coord_max_Tube_reactor)) + \mathbf{e}_{\text{area, b_z}}(r, z_coord_max_Tube_reactor)) + \mathbf{e}_{\text{area, q_z}}(r, z_coord_max_Tube_reactor) = \mathbf{e}_{\text{area, products, source}}(r)$$

52. Unit Jacket energy balance

$$\partial \mathbf{U}_{\text{Jacket}} / \partial t = (\mathbf{e}_{\text{cool_in, source}} - \mathbf{e}_{\text{cool_out, source}}) + \mathbf{e}_{\text{q, source}}$$

53. Flux reactants energy flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, reactants, source}}(r) = \mathbf{h}_{\text{reactants}}(r) * \mathbf{n}_{\text{area, reactants, source}}(r)$$

54. Flux products energy flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, products, source}}(r) = \mathbf{h}_{\text{products, source}}(r) * \mathbf{n}_{\text{area, products, source}}(r)$$

55. Flux flow_z energy flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, flow_z, source}}(r, z) = \mathbf{h}_{\text{Tube_reactor_rz, vap0}}(r, z) * \mathbf{n}_{\text{area, flow_z, source}}(r, z)$$

56. Flux a_z energy flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, a_z, source}}(r, z) = \mathbf{h}_{\text{a_z, source}}(r, z) * \mathbf{n}_{\text{area, o_XYLENE, a_z, source}}(r, z)$$

57. Flux b_z energy flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, b_z, source}}(r, z) = \mathbf{h}_{\text{b_z, source}}(r, z) * \mathbf{n}_{\text{area, OXYGEN, b_z, source}}(r, z)$$

58. Flux q_z energy flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, q_z, source}}(r, z) = (-\mathbf{k}_{\text{r, q_z}}(r, z) * (\partial \mathbf{T}_{\text{Tube_reactor_rz}}(r, z) / \partial \mathbf{z}_{\text{Tube_reactor}}))$$

59. Flux a_r energy flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, a_r, source}}(r, z) = \mathbf{h}_{\text{a_r, source}}(r, z) * \mathbf{n}_{\text{area, o_XYLENE, a_r, source}}(r, z)$$

60. Flux b_r energy flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, b_r, source}}(r, z) = \mathbf{h}_{\text{b_r, source}}(r, z) * \mathbf{n}_{\text{area, OXYGEN, b_r, source}}(r, z)$$

61. Flux q_r energy flux

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, q_r, source}}(r, z) = (-(\mathbf{k}_{\text{q_r}}(r, z) * (\partial \mathbf{T}_{\text{Tube_reactor_rz}}(r, z) / \partial \mathbf{r}_{\text{Tube_reactor}})))$$

62. Flux cool_in energy flux

$$\mathbf{e}_{\text{cool_in, source}} = \mathbf{h}_{\text{cool_in}} * \mathbf{n}_{\text{tot, cool_in, source}}$$

63. Flux cool_out energy flux

$$\mathbf{e}_{\text{cool_out, source}} = \mathbf{h}_{\text{jacket, liq0}} * \mathbf{n}_{\text{tot, cool_out, source}}$$

64. Flux q energy flux

$$\mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{e}_{\text{area, q, source}}(z) = \mathbf{U}_{\text{o, q}}(z) * (\mathbf{T}_{\text{Tube_reactor_rz}}(r_{\text{coord_max_Tube_reactor}}, z) - \mathbf{T}_{\text{jacket}})$$

65. Flux q flux integrated

$$\mathbf{e}_{\text{q, source}} = (\text{INTEGRAL}((\mathbf{e}_{\text{area, q, source}}(z) * \mathbf{r}_{\max, \text{Tube_reactor}}), \text{dz}_{\text{Tube_reactor}})) * (\theta_{\max, \text{Tube_reactor}} - \theta_{\min, \text{Tube_reactor}})$$

66. Phase Tube_reactor_rzmatl_V0 species o_XYLENE partial pressure

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{p}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z) = \mathbf{x}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z) * \mathbf{P}_{\text{Tube_reactor_rz}}(r, z)$$

67. Phase Tube_reactor_rzmatl_V0 species PHTHALIC_ANHYDRIDE partial pressure

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{p}_{\text{PHTHALIC_ANHYDRIDE, Tube_reactor_rz, vap0}}(r, z) = \mathbf{x}_{\text{PHTHALIC_ANHYDRIDE, Tube_reactor_rz, vap0}}(r, z) * \mathbf{P}_{\text{Tube_reactor_rz}}(r, z)$$

68. Phase Tube_reactor_rzmatl_V0 species OXYGEN partial pressure

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{p}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z) = \mathbf{x}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z) * \mathbf{P}_{\text{Tube_reactor_rz}}(r, z)$$

69. Phase Tube_reactor_rzmatl_V0 species WATER partial pressure

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{p}_{\text{WATER, Tube_reactor_rz, vap0}}(r, z) = \mathbf{x}_{\text{WATER, Tube_reactor_rz, vap0}}(r, z) * \mathbf{P}_{\text{Tube_reactor_rz}}(r, z)$$

70. Phase Tube_reactor_rzmatl_V0 reaction rxn0 forward rate

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$
$$\mathbf{r}_{\text{rxn0_forw_Tube_reactor_rzmatl_V0, Tube_reactor_rz, vap0}}(r, z) = ((\mathbf{A}_{\text{rxn0_forw}} * \exp((-E_{\text{rxn0_forw}}) / (\mathbf{Rgas} * \mathbf{T}_{\text{Tube_reactor_rz}}(r, z)))) * (\mathbf{p}_{\text{o_XYLENE, Tube_reactor_rz, vap0}}(r, z))) * (\mathbf{p}_{\text{OXYGEN, Tube_reactor_rz, vap0}}(r, z)))$$

71. Phase Jacketmatl_L0 molar density

$$\mathbf{N}_{\text{jacket, liq0}} = \rho_{\text{n, Jacket, liq0}} * \mathbf{V}_{\text{jacket, liq0}}$$

72. Unit Jacket extensive variable V_Jacketmatl decomposition

$$\mathbf{V}_{\text{jacket}} = \mathbf{V}_{\text{jacket, liq0}}$$

73. Phase Tube_reactor_rzmatl_V0 density correlation from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\rho_{n, \text{Tube_reactor_rz, vap0}}(r, z) = F(\mathbf{T}_{\text{Tube_reactor_rz}}(r, z), \mathbf{P}_{\text{Tube_reactor_rz}}(r, z))$$

74. Phase Jacketmatl_L0 density correlation from PPM

$$\rho_{n, \text{Jacket, liq0}} = F(\mathbf{T}_{\text{Jacket}})$$

75. Phase Tube_reactor_rzmatl_V0 species o_XYLENE fugacity coefficient correlation from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{fc}_{o_XYLENE, \text{Tube_reactor_rz, vap0}}(r, z) = 1$$

76. Phase Tube_reactor_rzmatl_V0 species PHTHALIC_ANHYDRIDE fugacity coefficient correlation from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{fc}_{\text{PHTHALIC_ANHYDRIDE}, \text{Tube_reactor_rz, vap0}}(r, z) = 1$$

77. Phase Tube_reactor_rzmatl_V0 species OXYGEN fugacity coefficient correlation from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{fc}_{\text{OXYGEN}, \text{Tube_reactor_rz, vap0}}(r, z) = 1$$

78. Phase Tube_reactor_rzmatl_V0 species WATER fugacity coefficient correlation from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{fc}_{\text{WATER}, \text{Tube_reactor_rz, vap0}}(r, z) = 1$$

79. Phase Tube_reactor_rzmatl_V0 internal energy correlation from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{u}_{\text{Tube_reactor_rz, vap0}}(r, z) = F(\mathbf{x}_{o_XYLENE, \text{Tube_reactor_rz, vap0}}(r, z), \mathbf{T}_{\text{Tube_reactor_rz}}(r, z), \mathbf{x}_{\text{PHTHALIC_ANHYDRIDE}, \text{Tube_reactor_rz, vap0}}(r, z), \mathbf{x}_{\text{OXYGEN}, \text{Tube_reactor_rz, vap0}}(r, z), \mathbf{x}_{\text{WATER}, \text{Tube_reactor_rz, vap0}}(r, z))$$

80. Phase Tube_reactor_rzmatl_V0 enthalpy correlation from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{h}_{\text{Tube_reactor_rz, vap0}}(r, z) = F(\mathbf{x}_{o_XYLENE, \text{Tube_reactor_rz, vap0}}(r, z), \mathbf{T}_{\text{Tube_reactor_rz}}(r, z), \mathbf{x}_{\text{PHTHALIC_ANHYDRIDE}, \text{Tube_reactor_rz, vap0}}(r, z), \mathbf{x}_{\text{OXYGEN}, \text{Tube_reactor_rz, vap0}}(r, z), \mathbf{x}_{\text{WATER}, \text{Tube_reactor_rz, vap0}}(r, z))$$

81. Phase Tube_reactor_rzmatl_V0 heat capacity correlation from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}}); \ \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}} \ \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{C}_{p, n, \text{Tube_reactor_rz, vap0}}(r, z) = F(\mathbf{x}_{o_XYLENE, \text{Tube_reactor_rz, vap0}}(r, z), \mathbf{T}_{\text{Tube_reactor_rz}}(r, z), \mathbf{x}_{\text{PHTHALIC_ANHYDRIDE}, \text{Tube_reactor_rz, vap0}}(r, z), \mathbf{x}_{\text{OXYGEN}, \text{Tube_reactor_rz, vap0}}(r, z), \mathbf{x}_{\text{WATER}, \text{Tube_reactor_rz, vap0}}(r, z))$$

82. Unit Jacket extensive variable Un_Jacketmatl decomposition

$$\mathbf{U}_{\text{Jacket}} = (\mathbf{N}_{\text{Jacket, liq0}} * \mathbf{h}_{\text{Jacket, liq0}}) - (\mathbf{P}_{\text{Jacket}} * \mathbf{V}_{\text{Jacket}})$$

83. Phase Jacketmatl_L0 enthalpy correlation from PPM

$$\mathbf{h}_{\text{Jacket, liq0}} = F(\mathbf{T}_{\text{Jacket}})$$

84. Phase Jacketmatl_L0 heat capacity correlation from PPM

$$\mathbf{C}_{p, n, \text{Jacket, liq0}} = F(\mathbf{T}_{\text{Jacket}})$$

85. Connection reactants phase reactants_phase enthalpy correlation from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}} \ \mathbf{r}_{\max, \text{Tube_reactor}})$$

$$\mathbf{h}_{\text{reactants}}(r) = F(\mathbf{x}_{o_XYLENE, \text{reactants}}(r), \mathbf{x}_{\text{PHTHALIC_ANHYDRIDE}, \text{reactants}}(r), \mathbf{x}_{\text{OXYGEN}, \text{reactants}}(r), \mathbf{x}_{\text{WATER}, \text{reactants}}(r), \mathbf{T}_{\text{reactants}}(r))$$

86. Flux a_z species o_XYLENE enthalpy from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}, \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{h}_{\mathbf{a}_z, \text{source}}(r, z) = 19080000 + ((74396 * (\mathbf{T}_{\text{Tube_reactor_rz}}(r, z) - 298.15)) + (131.755 * ((\mathbf{T}_{\text{Tube_reactor_rz}}(r, z))^2 - 88893.422)))$$

87. Flux \mathbf{b}_z species OXYGEN enthalpy from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}, \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{h}_{\mathbf{b}_z, \text{source}}(r, z) = 0 + ((26311.5 * (\mathbf{T}_{\text{Tube_reactor_rz}}(r, z) - 298.15)) + (4.7675 * ((\mathbf{T}_{\text{Tube_reactor_rz}}(r, z))^3 - 88893.422)))$$

88. Flux \mathbf{a}_r species o_XYLENE enthalpy from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}, \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{h}_{\mathbf{a}_r, \text{source}}(r, z) = 19080000 + ((74396 * (\mathbf{T}_{\text{Tube_reactor_rz}}(r, z) - 298.15)) + (131.755 * ((\mathbf{T}_{\text{Tube_reactor_rz}}(r, z))^2 - 88893.422)))$$

89. Flux \mathbf{b}_r species OXYGEN enthalpy from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}, \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{h}_{\mathbf{b}_r, \text{source}}(r, z) = 0 + ((26311.5 * (\mathbf{T}_{\text{Tube_reactor_rz}}(r, z) - 298.15)) + (4.7675 * ((\mathbf{T}_{\text{Tube_reactor_rz}}(r, z))^3 - 88893.422)))$$

90. Connection cool_in phase cool_in_phase enthalpy correlation from PPM

$$\mathbf{h}_{\text{cool_in}} = F(\mathbf{T}_{\text{cool_in}})$$

91. Phase Tube_reactor_rzmatl_V0 molecular weight from PPM

$$\mathbf{r} := (\mathbf{r}_{\min, \text{Tube_reactor}}, \mathbf{r}_{\max, \text{Tube_reactor}}); \mathbf{z} := (\mathbf{z}_{\min, \text{Tube_reactor}}, \mathbf{z}_{\max, \text{Tube_reactor}})$$

$$\mathbf{mw}_{\text{Tube_reactor_rz, vap0}}(r, z) = (((\mathbf{x}_{\text{o}_XYLENE, \text{Tube_reactor_rz, vap0}}(r, z) * 106.167) + (\mathbf{x}_{\text{PHTHALIC_ANHYDRIDE, \text{Tube_reactor_rz, vap0}}}(r, z) * 148.118)) + (\mathbf{x}_{\text{OXYGEN, \text{Tube_reactor_rz, vap0}}}(r, z) * 31.9988)) + (\mathbf{x}_{\text{WATER, \text{Tube_reactor_rz, vap0}}}(r, z) * 18.0153))$$

92. Phase Jacketmatl_L0 molecular weight from PPM

$$\mathbf{mw}_{\text{jacket, liq0}} = 18.0153$$

93. user_equation

$$\mathbf{n}_{\text{tot, cool_in, source}} = \mathbf{n}_{\text{tot, cool_out, source}}$$