DEVELOPMENT OF COMPUTER-AIDED CHEMICAL PROCESS FLOWSHEET DRAWING

ROSCHONG SAKDUMRONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF SCIENCE
(TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2000

ISBN 974-663-835-1

Thesis

entitled

DEVELOPMENT OF COMPUTER-AIDED CHEMICAL PROCESS FLOWSHEET DRAWING

Roschong Sakdumrong
.....
Miss Roschong Sakdumrong
Candidate

Suttinant Nantachit
.....
Lect.Suttinant Nantachit, M.S.
Major-advisor

R. Wanchai
.....
Assoc. Prof. Wanchai Rivepiboon, Ph.D.
Co-advisor

.....
Lect.Chumchok Namsrisakulrat, M.Eng.
Co-advisor

.....
Prof.Liangchai Limlomwongse, Ph.D.
Dean
Faculty of Graduate Studies

.....
Lect.Thanakorn Uao-On, D.Engr.
Chairman
Master of Science Programme in Technology of Information System Management
Faculty of Engineering

Thesis

entitled

DEVELOPMENT OF COMPUTER-AIDED CHEMICAL PROCESS FLOWSHEET DRAWING

was submitted to the Faculty of Graduate Studies, Mahidol University for the degree

of Master of Science (Technology of Information System Management)

on

March 31, 2000

Roschong Sakdumrong
Miss Roschong Sakdumrong
Candidate

Suttinant Nantachit
Lect.Suttinant Nantachit, M.S.
Chairman

R. Wanchai
Assoc. Prof. Wanchai Rivepiboon, Ph.D.
Member

Kasem Kulp.
Asst.Prof.Kasem Kulpradit, M.Sc.
Member

Lect.Chumchok Namsrisakulrat, M.Eng.
Member

Liangchai Limlomwongse,
Ph.D.
Dean
Faculty of Graduate Studies
Mahidol University

Lect.Thanakorn Uao-On, D.Engr.
Dean
Faculty of Engineering
Mahidol University

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and deep appreciation to Mr.Suttinant Nantachit my principal supervisor for his guidance, encouragement throughout their great assistance in programming, and solving technical problems. He was never lacking in kindness and support.

I would like to thank Dr.Wanchai Rivepiboon for his constructive comments, support with respect to the object-oriented technology, supervision and encouragement. I am equally grateful to Mr.Chumchok Namsrisakulrat my associate supervisor for his helpful guidance constructive comments, and supervision.

Thanks are also extended to all my friends and IT.MU.'s officers for all of their support and friendship.

Finally, I wish to express my deepest thanks to my grandmother, my aunt, brother and sister for their everlasting supports and enthusiasm throughout the whole life.

Roschong Sakdumrong

4037675 EGTI/M : MAJOR : TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT; M.SC. (TECHNOLOGY OF INFORMATION SYSTEM MANAGEMENT)

KEY WORDS : OBJECT - ORIENTED TECHNOLOGY , ANALYSIS AND DESIGN / FLOWSHEETS / CHEMICAL SYMBOLS / CHEMICAL PROCESSES

ROSCHONG SAKDUMRONG: DEVELOPMENT OF COMPUTER-AIDED CHEMICAL PROCESS FLOWSHEET DRAWING. THESIS ADVISORS: SUTTINANT NANTACHIT, M.S., WANCHAI RIVEPIBOON, Ph.D., CHUMCHOK NAMSRISAKULRAT, M.Eng. 116 p. ISBN 974-663-835-1

In the chemical process design, the chemical engineer designs the flowsheet that illustrates the sequence of process units and the overview of the chemical process. However, the creation and modification of the flowsheet tends to be difficult and time consuming. Worse yet, it seems to be possible that errors can occur in the design step of the complex flowsheet.

Hence, the purpose of this study is to design and develop a suitable software called ChemProc for drawing the chemical process flowsheet. With this software the user can draw the flowsheet using a symbol in the application or using theirs own symbol. The line symbol and the equipment symbol drawn on the flowsheet can be connected to ease flowsheet modification. In addition, this application supports the design of a complete flowsheet and manages the properties of each symbol on the flowsheet. Furthermore, each symbol contains information about its own technical specification. Users can edit the technical specification and view or print output reports. All symbols on the flowsheet are checked for the correct amount of input and output to make the flowsheet complete.

4037675 EGTI/M: สาขาวิชา : เทคโนโลยีการจัดการระบบสารสนเทศ; วท.ม. (เทคโนโลยีการจัดการระบบสารสนเทศ)

รสชง ศักดิ์คำรง : การพัฒนาโปรแกรมคอมพิวเตอร์ช่วยสร้างแผนภาพของกระบวนการทางเคมี (DEVELOPMENT OF COMPUTER-AIDED CHEMICAL PROCESS FLOWSHEET DRAWING) คณะกรรมการควบคุมวิทยานิพนธ์: สุทธินันท์ นันทจิต, M.S., วันชัย ริ้วไพบูลย์, Ph.D., ชุมโชค นำศรีสกุลรัตน์, M.Eng. 116. ISBN 974-663-835-1

การออกแบบกระบวนการทางเคมี นักวิศวกรเคมีจะเขียนแผนภาพ (Flowsheet) กระบวนการทางเคมี เพื่อแสดงถึงภาพรวมของกระบวนการ โดยปกติแล้วการออกแบบและการวาด แผนภาพทางเคมีจะเป็นงานที่ใช้เวลามาก และการแก้ไขสามารถทำได้ยาก โดยเฉพาะอย่างยิ่งแผนภาพที่มีความซับซ้อนมาก มักมีความจำเป็นในการแก้ไขการออกแบบให้มีความสมบูรณ์ ถูกต้อง ชัดเจนมากขึ้น

ดังนั้นจึงมีการออกแบบและพัฒนาเครื่องมือที่จะช่วยในการวาดแผนภาพของกระบวนการเคมีขึ้น ซึ่งผู้ใช้สามารถออกแบบแผนภาพได้ โดยการนำสัญลักษณ์ทางเคมีที่ได้สร้างไว้ในโปรแกรมประยุกต์ หรือนำสัญลักษณ์ที่ผู้ใช้สร้างขึ้นเองมาวาดลงบนแผนภาพ นอกจากนี้เพื่อให้การออกแบบแผนภาพสามารถทำได้โดยสมบูรณ์ และมีข้อมูลของคุณลักษณะของสัญลักษณ์ต่างๆ บรรจุอยู่ด้วย ซอฟแวร์ประยุกต์ที่สร้างขึ้นจึงมีการทำงานที่ช่วยให้ผู้ใช้สามารถใส่ข้อมูลการกำหนดรายละเอียดทางเทคนิคของสัญลักษณ์แต่ละสัญลักษณ์ได้ และสามารถเรียกดูได้โดยสะดวก ยิ่งไปกว่านั้น สัญลักษณ์ต่างๆ ที่ผู้ใช้วาดลงบนแผนภาพจะมีการตรวจสอบจำนวนของวัสดุที่เข้าและออกจากอุปกรณ์ได้เองโดยอัตโนมัติ เพื่อช่วยให้ผู้ใช้สามารถเห็นถึงความผิดพลาดที่เกิดขึ้นได้

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

CHAPTER I

INTRODUCTION

1.1 Background and Statement of Problem

The purpose of chemical engineering is to design and to create new material. But the step of design is often an enormously complex problem. Because the chemical engineering design is the process that requires the engineering principles and theories combined with a practical realization by industrial. In the process design a flowsheet is the important element. The flowsheet is draw in the early stages of process design by generate idea and then translate them into the flowsheet. In the flowsheet, symbols are used to represent equipment or process unit and lines to represent input and output material. The flowsheet has many benefits such as showing the sequence of process unit, equipment, flow of materials and detailed information on flow for producing new materials or for upgrading the existing materials.

In the chemical process flowsheet, the flowsheet is usually complex and large size. The creation and modification of the flowsheet is difficult and times consuming. In addition, it is possible that the number of input and output of each equipment or process unit was wrong and causes the incomplete flowsheet. Computer should be used to create the flowsheet instead of manual drawing. The software developed can help the chemical engineer to draw the complex flowsheet and support flexible design flowsheet for ease to modify. During creation, the software can verify the number of input and output material of each equipment and process unit for

making the completed flowsheet. In addition, the software also supports the detailed information of each equipment, process unit, and material.

1.2 Objective

To design and develop the software using object technologies to create drawing for the chemical process flowsheet.

1.3 Scope of work

The software developed to draw chemical process will have the following functions:

1. Create line and symbol.
2. Connect line to symbol.
3. Enter data specification in line and symbol.
4. Automatic checker of input and output.
5. General output image and report.

1.4 Expected results

Software for drawing the chemical process flowsheet has the following benefits:

1. Save time in drawing chemical process flowsheet.
2. Easy to create and modify the flowsheet.
3. Reduce human error by software checking.
4. Technical specification included in each symbol.

CHAPTER II

LITERATURE REVIEW

The content of this chapter is the literatures and other application that related work. The topic covers the chemical process, the flowsheet and characteristics of the object-oriented approach.

2.1 The chemical process

The development of a commercial product from a laboratory chemical involves the talents and efforts of many people. At the laboratory stage, the market potential of the product is estimated. The physical and chemical properties of the chemicals involved in the process are then determined, and the conditions under which the product can be produced in the laboratory are explored. But a long and complex effort still remains. There are times when an engineer is called upon to devise a plausible process long before all of the development effort has taken place. Perhaps a decision whether to invest in the development effort is required. If a tentative process can be suggested, the cost and difficulty of the development of the process can be estimated. Also, a rough estimate of the cost of the product can be calculated, and the size of the market can be forecast. Like most economic forecasts, these "guesstimates" are subject to an enormous amount of uncertainty. Accordingly, many technically and economically sound processes and products have been shipwrecked on the rocks of poor marketing forecast (1).

Process development

Process development covers not only developing a manufacturing process for an entirely new product but also a new process or route for an existing product. The push for the latter may originate for one or more of the following reasons: availability of new technology, change in the availability and/or cost of new materials. Process development for a new product depends on things such as the scale on which it is to be manufactured, the by-products formed and their removal/recovery, and required purity. Data will be acquired during this development stage using semi-technical plant, which will be invaluable in the design of the actual manufacturing plant. If the plant is to be a very large capacity, continuously operating one, then a pilot plant will first be built and operated to test out the process and acquire more data. These semi-technical or pilot plants will also supply increased quantities, which will be required for testing, or customer evaluation. A major part of the process development activity for a new plant is to minimize, or ideally prevent by designing out, waste production and hence possible pollution. The economic and environmental advantages of this are obvious (2).

Process synthesis

The development of a process scheme involves coming up with that configuration of processing steps that efficiently and safely produces the desired product. An enormous amount of art, skill, intuition, and innovation goes into developing the processing scheme. Literally millions of designs are possible. Somehow the chemical engineer must sort through the myriad of alternatives to come up with a good process (1).

The chemical engineer should always be willing to consider completely new designs. Process design refers to the actual design of the equipment, facilities necessary for carrying out the process and design new materials. One of the most important items that we develop during a design is a process flowsheet.

2.2 Flowsheet

George describes the flowsheets in the following manner (3):

Flowsheets allow a great deal of information to be collected and examined in a small space. There are a great many types of flowsheets used: some show simple flows, others show material balances and some are so complete. The flowsheets have been reduced to maximum simplicity and are designed to show principles rather than details.

Flowsheets come in many types and are used for many purposes. Simple blocks often show only material and energy flows and operating conditions. More elaborate ones show everything in a process and become extremely complicated. Specialized ones show such details as fire lines, instruments and control systems, air lines, drains, etc. Most coordinate the sequence of unit operations and unit processes. They indicate the point of entrance for raw material and any necessary energy, likewise the points of removal of the product and the by-produces. Many flowsheets are made to illustrate proposed alterations as a design proceeds. No other description of a chemical process is as concise or as revealing regarding equipment, operating details, and general reactions as that presented by a skillfully drawn flowsheet, which should include data covering not only materials, but labor and utilities as well.

An overview of the concept is provided by Douglas (4):

The flowsheet shows the major pieces of equipment, and usually each piece of equipment is given a special number or name. Normally each stream on the flowsheet is also lettered or numbered, and a stream table that contains theses or numbers often appears at the bottom of the flowsheet. The stream table contains the flows of each component in every stream as well as the stream temperatures and pressures. In some case, enthalpies, densities, and other information for each stream are included in the stream table.

The process flowsheet of *flow sheeting software* defines the reactors, unit operations, and utilities needed, along with the stream of material and energy into and out of the process and between all the process units.

The chemical process flowsheet are given in the Appendix A.

Equipment

Equipment is emphasized in conjunction with descriptions of the various processes and with flowsheets representing these processes. Any chemical engineer should start early to become familiar with industrial equipment such as pumps, filter presses, distillation towers, nitrators and sulfonators (3).

What do chemical engineers need to use flowsheet?

When the chemical engineer are presented with a description of a process like this one and are then asked to determine something about the process, it is essential to organize the given information is a way that is convenient for subsequent calculations. The best way to do this is to draw a flowsheet of the process (5).

The chemical engineer uses flowsheet to show the sequence of equipment and unit operations in the overall process, to simplify visualization of the

manufacturing procedures, and to indicate the quantities of materials and energy transfer (6).

2.3 Other types of computer-aided design program

There are computer-aided design packages that are based on an equation-solving approach. That is, they attempt to solve all the equations describing the flowsheet simultaneously. Numerous computer-aided design programs are available commercially including flowtran, process synthesis, flow-sheeting programs and ChemCad are those most specifically intended for design use.

FLOWTRAN is a computer-aided design (CAD) program that available commercially. General structure of *flowtran* has essentially the following components:

1. An executive system. The executive system reads the input data, controls the order in which the equipment subroutines are calculated, prints the results of the calculations, etc.
2. A physical properties data bank. The physical properties data bank contains data for 180 chemical compounds, including molecular weights, critical constants, heat capacities, acentric factors, etc.
3. A thermodynamics properties package. The thermodynamics package uses the pure component properties in data bank to calculate gas and liquid densities and enthalpies, as well as vapor-liquid equilibrium relationships, for the process streams in the plant.
4. A collection of design (and cost) subroutines for a variety of process unit.

The equipment design subroutines use the information on the stream properties to calculate equipment sizes (and costs) for the various process units, i.e., furnaces, heat exchangers, compressors, distillation columns, etc.

PROCESS SYNTHESIS involves generating a flowsheet for a process to produce a particular product or slate of products from specified raw materials. The process flowsheet identifies the chemical reactors and unit operations required and their sequence, the material and energy streams in the process, those streams to be recycled, and some characteristics of the equipment. This flowsheet should be a reasonable approximation of the best economic flowsheet for the product slate. *Process synthesis* software, a type of expert system program.

FLOW-SHEETING as used in computer-aided design, means performing on a specified flow sheet the calculations necessary to simulate the behavior of the process or to design the equipment and to determine values for key operating conditions. These calculations include mass and energy balances, process-equipment parameters, and cost estimation for the equipment and plant as well as an economic analysis of the process.

Chemical process *flow-sheeting* program include property data and prediction methods for pure components and mixtures. An objective of *flow-sheeting* is specified, typically the desired annual production rate of the principal products, as is the process flow sheet.

CHEMCAD is the software created by Chempute Software (7) that can be used to design or simulate almost any steady state process plant having recycled

streams and control loops. Models for virtually all standard unit operations simulate the behavior of a process under varying feed rates, product rates, temperatures, pressures and compositions. The program contains the 1500 component DIPPR database with a separate user database, which can be defined or modified by the user. Components are selected dynamically by ID number, formula, synonym or class. The program also includes property prediction routines for components not included in the database.

The flowsheet representing the plant layout is input graphically, using a mouse, with different unit operations being represented by user-modifiable icons. Stream numbering is done automatically. Data describing each feed stream and unit operation is entered via pulldown menu options, available from the flowsheet graphics input screen. Automatic error checking prevents overspecification or missing data entries.

The simulation can be halted during calculation, for viewing of intermediate results. The input data may then be modified and calculations resumed at the point where they were originally interrupted. The interactive interface also permits individual unit operations to be run separately to the complete flowsheet for quick "what-if" studies.

Results output by the program include a full heat and material balance, thermodynamic and physical properties of all streams, component flowrates, stream temperatures, pressures and vapour-fractions, and process equipment parameters. The user may view the results graphically or may obtain a printout of detailed or summary results.

2.4 Object Technologies

A discussion of object technologies is presented by Pressman (8):

The term "Object Technologies" (9) is often used to encompass all aspects of an object-oriented view and includes analysis, design and testing methods: programming languages; tools; databases; and the applications that are created using as object-oriented approach. Object technologies lead to reuse, and reuse (of program component) leads to faster software development and higher-quality program. Object-oriented software is easier to adapt, to scale and to maintain because its structure is inherently decoupled. This leads to fewer side effects when changes have to be made and less frustration for the software engineer.

Objects and Classes

Objects are categorized into classes and class hierarchies. A class is an object-oriented concept that encapsulates the data and procedural abstractions that are required to describe the content and behavior of some real world entity. By definition, all objects that exist within a class inherit its attributes and the operations that are available to manipulate the attributes. A superclass is a collection of classes, and a subclass is an instance of a class. These definitions imply the existence of a class hierarchy in which the attributes and operations of the superclass are inherited by subclasses. At each level of the class hierarchy, new attributes and operations may be added to those that have been inherited from higher levels in the hierarchy.

Attribute and Operation

Attributes are attached to classes and objects, and that they describe the class or object in some way. An object encapsulates data and the algorithms that

process the data. These algorithms are called operations and can be viewed as modules in a conventional sense. Each of the operations that is encapsulated by an object provides a representation of one of the behaviors of the object.

Three important concept differentiate the object-oriented approach from conventional software engineering (8):

1. Encapsulation packages data and the operations that manipulate the data into a single named object. There are the most important benefits of encapsulate data and operations:
   - The internal implementation details of data and procedures are hidden from the outside world. This reduces the propagation of side effects when changes occur.
   - Data structures and the operations that manipulate them are merged in a single named entity.
   - Interfaces among encapsulated objects are simplified. An object that sends a message need not be concerned with the details of internal data structures in the receiving object.
2. Inheritance enables the attributes and operation of a class to be inherited by all subclasses and the objects that are instantiated from them.
3. Polymorphism enables a number of different operation to have the same name, reducing the number of lines of code required to implement a system and facilitating changes when they are made.

The object-oriented paradigm

When a class is built, the class is looked up in a library. Because object-oriented software engineering emphasizes reuse. If a class cannot be found in the library, object-oriented analysis, object-oriented design, object-oriented programming, and object-oriented testing are applied to create the class and the objects derived from the class.

Object-oriented Analysis

The objective of object-oriented analysis is to develop a series of models that describe computer software as it works to satisfy a set of defined requirement. Object-oriented analysis methods enable a software engineer to model a problem by representing objects, attributes, and operations the primary modeling components. The intent of object-oriented analysis is to fine all classes, the relationships and behavior associated with all classes that are relevant to the problem to be solved. To accomplish this, four basic principles are applied:

1. Classes must be identified.
2. Object-to-object relationships should be represented.
3. A class hierarchy must be specified.
4. Object behavior must be modeled.

Tasks 1 through 4 are reapplied iteratively until the model is complete.

1. Identifying Classes

Class-responsibility-collaborator (CRC) modeling (10) provides a simple means for identifying and organizing the classes that are relevant to system or product

requirements. The CRC model may make use of actual or virtual index cards (Figure 2.1).

The class can be identified by examining the problem statement or by performing a grammatical parse on the processing narrative for the system to be built. Classes are determined by underlining each noun or noun clause and entering it in a simple table. Synonyms should be noted. Responsibilities are the attributes and operations that are relevant for the class. Collaborators are those classes that are required to provide a class with the information needed to complete a responsibility.

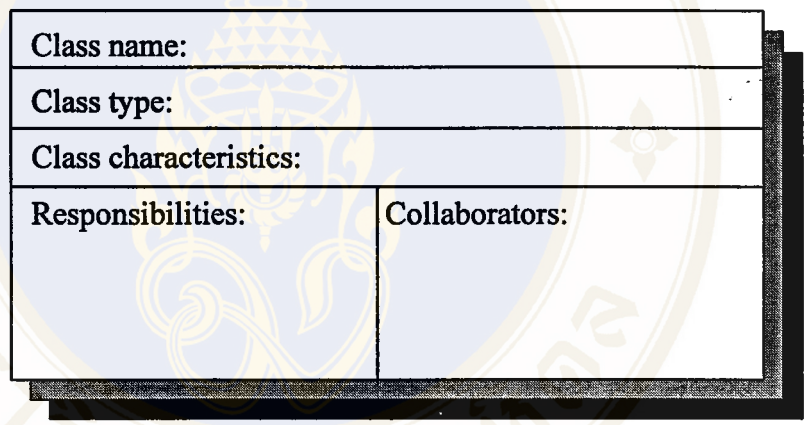


Figure 2.1 A CRC model index card

Specify class hierarchy

A class hierarchy shows the existence of classes and a relationship among classes where in one-class shares the structure or behavior defined in one (single inheritance) or more (multiple inheritance) other classes. Inheritance defines a *kind of* hierarchy among classes in which a subclass inherits from one or more superclass. Coad and Yourdon suggest that a generalization-specialization (gen-spec) structure (Figure 2.2) should be derived for the identified classes.

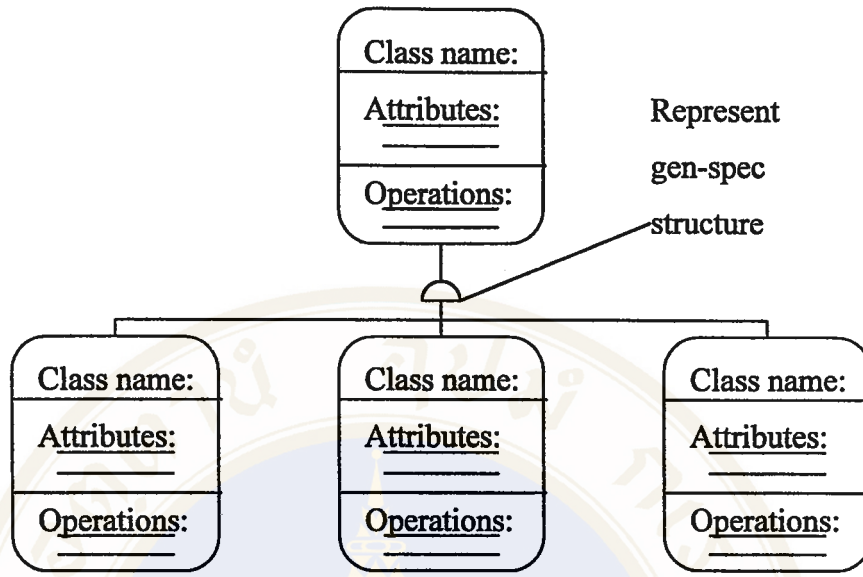


Figure 2.2 Gen-spec structure notation

In other cases, the has_relationship denotes a whole-part relationship between two classes, where the whole class is at one end of the relationship and the part class is at the other end. The whole class contains or owns its part object. This relationship is also known as an aggregation relationship. These aggregate objects can be defined using the notation represented in Figure 2.3 (11).

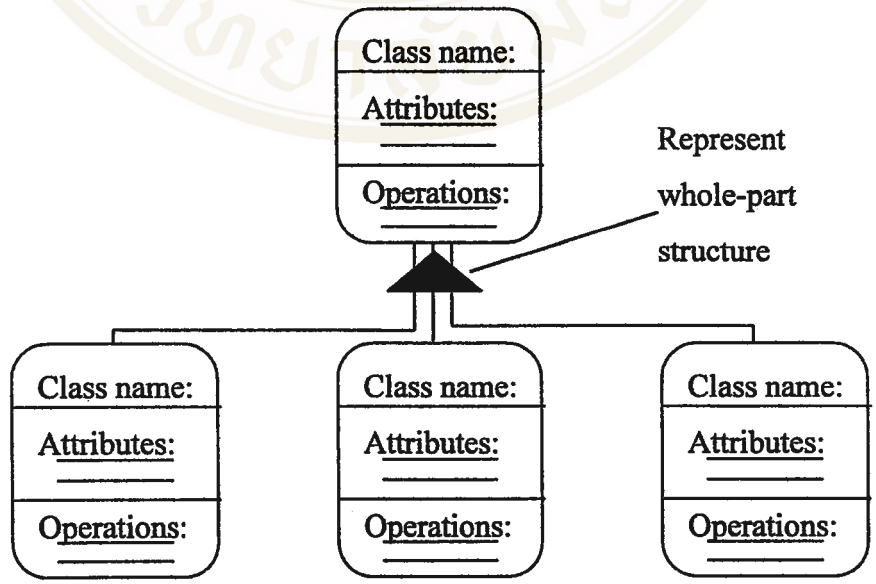


Figure 2.3 Whole-part structure notation

Object relationship model

The first step in establishing relationships is to understand the responsibilities for each class. The CRC model index card contains a list of responsibilities. The next step is to define those collaborator classes that help in achieving each responsibility. This establishes the connection between classes. The relationships can be derived by examining the verbs or verb phrases in the grammatical parse. The verbs are isolated to indicate physical location or placement (next to, part of, contained in), communications (transmits to, acquires from), ownership (incorporated by, is composed of), and satisfaction of a condition (manages, coordinates, controls).

Object behavior model

The object-behavior model represents dynamic element of the object-oriented analysis model. Therefore we must represent the behavior of the system as a function of specific events and time. The object behavior model indicates how an object-oriented system will respond to external events or stimuli. An event occurs whenever an object-oriented system and an actor (recall that an actor can be a person, a device, or even an external system) exchange information. An event is Boolean that is not the information that has been exchanged. The one type of behavioral representation for object-oriented analysis considers a state representation for the overall product or system. This representation encompasses a simple event trace model that indicates how events cause transitions from object to object that depicts the processing behavior of each object. Once a complete event trace has been developed, all of the events that cause transitions between system objects can be collated into a set

of input events and output events. This can be represented using an event flow diagram (12).

Object-oriented Design

The first step in the construction activities for object-oriented systems is object-oriented design. Object-oriented design transform the analysis model created using object-oriented analysis into a design model that serves as a blueprint for software construction. Object-oriented design enables a software engineer to indicate the objects that are derived from each class and how these objects interrelate with one another. In addition, object-oriented design depicts how the relationships among objects are achieved, how behavior is to be implement, and how communication is to be implemented between objects. For object-oriented systems, the object-oriented design step can be defined as a design pyramid. As shown in figure 2.4, the four layers of the object-oriented design pyramid are (8):

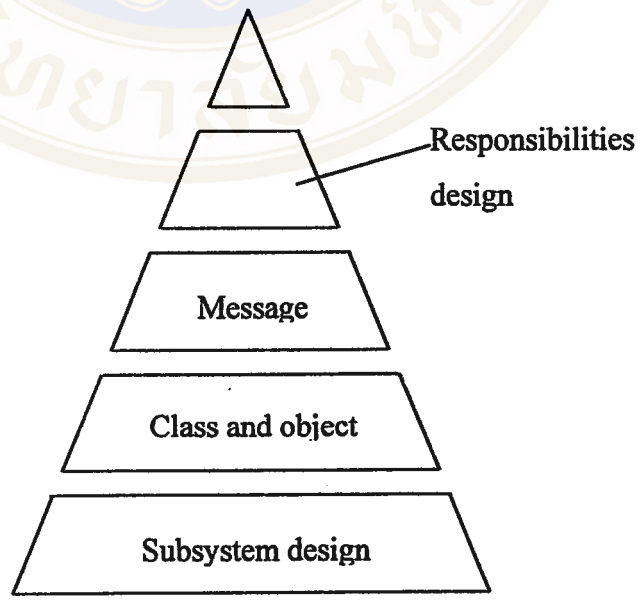


Figure 2.4 The object-oriented design pyramid

1. *The subsystem layer.* Contains a representation of each of the subsystems that enable the software to achieve its customer-defined requirements and to implement the technical infrastructure that supports customer requirements., the following subsystem design component were defined:
    - Problem domain: the subsystems that are responsible for implement customer requirements directly.
    - Human interaction: the subsystems that implement the user interface.
    - Task management: the subsystems that are responsible for controlling and coordinating concurrent tasks that may be packaged within a subsystem or among different subsystems.
    - Data management: the subsystem that is responsible for the storage and retrieval of object.

    As subsystems are designed, they should conform to the following design criteria:
    - The subsystem should have a well-defined interface through which all communication with the rest of the system occurs.
    - With the exception of a small number of "communication classes," the classes within a subsystem should collaborate only with other classes within the subsystem.
    - The number of subsystems should be kept small.
    - Subsystems can be partitioned internally to help reduce complexity.
2. *The class and object layer.* Contains the class hierarchies that enable the system to be created using generalizations and increasingly more targeted

specialization. This layer also contains design representations of each object.

3. *The message layer.* Contains the details that enable each object to communicate with its collaborators. This layer establishes the external and internal interfaces for the system.
4. *The responsibilities layer.* Contains the data structure and algorithmic design for all attributes and operations for each object.

Object-oriented Programming

Object-oriented programming is a method for structuring programs as hierarchically organized classes describing the data and operations of objects that may interact with other objects (13).

Object-oriented programming extends the design model into the executable domain. An object-oriented programming language is used to translate the classes, attributes, operations and messages into a form that can be executed by a machine (8).

Object-oriented Testing

The objective of object-oriented testing is to find the greatest possible number of errors with a manageable amount of effort applied over a realistic time span. The view of testing broadens to include the review of both the analysis and design model. In addition, the focus of testing moves away from the procedural component and toward the class (8).

Testing object-oriented analysis and object-oriented design models

The definition of testing must be broadened to include error discovery techniques applied to object-oriented analysis and object-oriented design models. Because of the evolutionary nature of the object-oriented software engineering paradigm, these models begin as relatively informal representations of system requirements and evolve into detailed models of classes, class connections and relationships, system design and allocation, and object design. At each stage, the models can be testing in an attempt to uncover errors prior to their propagation to the next iteration.

Unit and integration testing

The strategy for unit and integration testing must change significantly. Object-oriented testing process, we begin with unit testing, then progress toward integration testing, and culminate with validation and system testing. In conventional applications, unit testing focuses on the smallest compilable program unit (the subprogram). Once each of these units has been testing individually, it is integrated into a program structure while a series of regression tests are run to uncover errors due to interfacing the modules and side effects that are caused by the addition of new units. Finally, the system as a whole is tested to ensure that errors in requirements are uncovered.

Fault-based testing

The object of fault-based testing within an Object-oriented system is to design tests that have a high likelihood of uncovering plausible faults. Because the product or system must conform to customer requirements, the preliminary planning

required to perform fault-based testing begins with the analysis model. The tester looks for plausible faults. To determine whether these faults exist, test cases are designed to exercise the design or code.

Scenario-based testing

Scenario-based testing concentrates on what the user does, not what the product does. It means capturing the tasks that the user has to perform, then applying them and their variants as tests.

Black-box testing

Black-box testing, focuses on the functional requirements of the software. That is, black-box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black-box testing attempts to find errors in the following categories:

1. Incorrect or missing functions and Interface errors
2. Errors in data structures or external data base access
3. Performance errors
4. Initialization and termination errors

CHAPTER III

MATERIALS AND METHODS

3.1 Materials

Hardware: Personal computer,

CPU Celeron 333 or upper,

RAM 64 MB or upper,

Harddisk 4.3 GB or upper,

Keyboard 104 key,

Mouse,

Printer

Software: Operating system: Windows98

Application development: C++ Builder 4.0

Database: Paradox

Help authoring tool: HelpScribble

Image authoring tool: Image Editor

3.2 Methods

3.2.1 Study the chemical process flowsheet, the object technologies and the C++Builder.

3.2.2 Analysis

Analysis is the first technical activity in object-oriented technology using the object-oriented analysis (OOA) of Pressman, Roger S. That is, the software architecture is analyzed in an effort to model a problem by representing object, attributes, and operations as the primary modeling components and class hierarchies as the relationship between any two classes that are connected. The OOA can derive in the following step (8):

3.2.2.1 Identify classes

The technique for identifying classes is to perform a grammatical parse on the processing narrative for the system. All nouns become potential objects. The potential object can be external entities, things, occurrences or events, roles, organizational units, places or structures.

3.2.2.2 Create object-to-object relationships

The object-relationship is considered using a grammatical parse. Verbs are isolated that indicate physical location or placement, communications, ownership, and satisfaction of a condition. These provide an indication of a relationship. The object-relationship model is represented by Coad and Youndon (11) notation.

3.2.2.3 Create object behavior model

The object behavior model indicates the behavior of individual objects and the overall behavior of the object-oriented system. Using a grammatical parse, verbs become candidate operation. Each operation that is chosen for a class exhibits a behavior of the class. The behavioral representation is a representation of how events cause flow from one object to another called an event trace. An event flow diagram

(12) is used to represent all of the events that cause transitions between object can be collated into a set of input events and output events.

3.2.3 Design

Object-oriented design transforms the analysis model created using object-oriented analysis into a design model for software construction. The tasks of the object-oriented design by Pressman (8) are:

3.2.3.1 Subsystem design

The analysis model is partitioned into subsystem for layout a schedule. In addition to, the collaborations that exist between the subsystems are also defined.

3.2.3.2 Object design

Object design contains a description of each class, which contains a set of attributes and operations using the processing narrative for the problem. In addition to, all database are design for the class needed to store and retrieve the data.

3.2.3.3 User Interface design

In this step, the external interfaces are established. That is, the interactive screen that user will use to work with the system are created such as menus, symbol palette, drawing area.

3.2.3.4 Procedural design

Algorithms and data structures are designed. An algorithm is created to implement the specification for each operation in English statements. Data structures are derived from the attributes that appropriate for algorithms.

3.2.4 Implementation

Object-oriented programming extends the design model into a C++builder using target from object-oriented design.

3.2.4.1 Develop user interface

All screens on the human interface design are created on C++builder form.

3.2.4.2 Coding

In this step, each operation is coded using designed algorithms that relevant data structure of the attribute. And the system will be programmed in C++builder

3.2.5 Testing

All objected-oriented models should be tested for correctness, completeness and consistency. Testing software uses the following step:

3.2.5.1 Class testing

In this step, each classes are tested that is the operations encapsulated by this class and the state behavior of this class are also tested. And the values of its attributes are examined if errors exist.

3.2.5.2 Integration testing

The dependent classes are tested and the independent classes that used by the dependent classes are tested continues until the entire system.

3.2.5.3 Validation testing

In this step, using black-box testing that attempts to find errors in the requirements specification and using example of flowsheet in Appendix A for testing.

3.2.6 Evaluation

In this step, the software is evaluated the expected benefits to ensure that the flowsheet are drawn quickly and completely

3.2.7 Develop user manual for describing step of using program

CHAPTER IV

RESULTS

In this chapter, to illustrate the results of working step that consists of the results of the analysis step and the design step.

4.1 The results of the analysis step

The grammatical parse on the processing narrative of the software (called ChemProc) creation for drawing the flowsheet can be stated in the following paragraphs. To illustrate, underlining the first occurrence of all nouns and italicizing the first occurrence of all verbs isolates all nouns and verbs in the narrative that shown in the following:

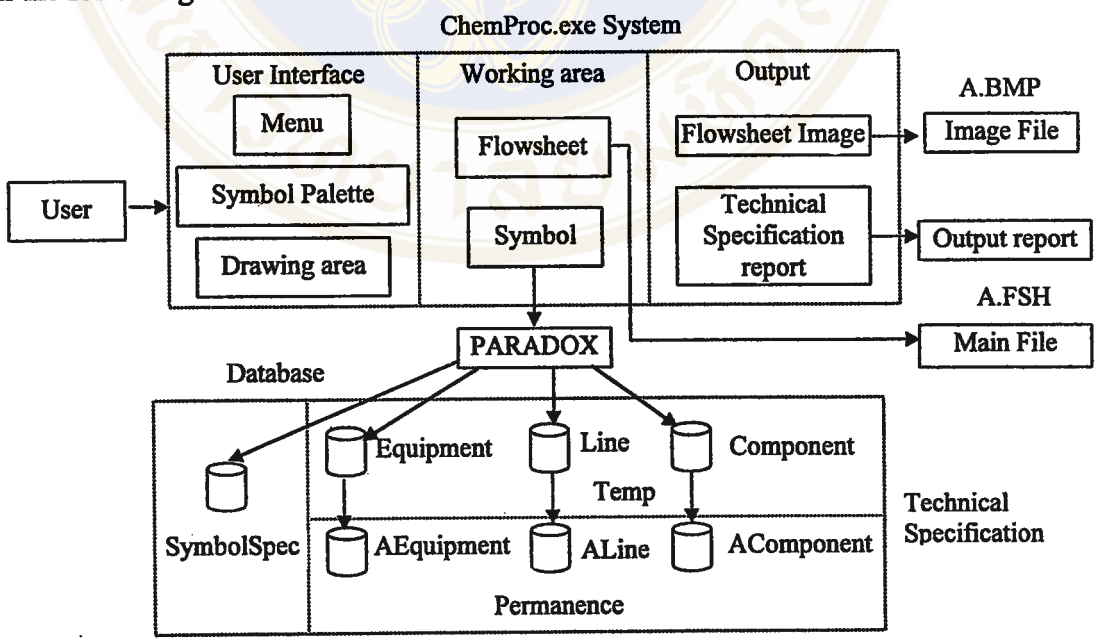


Figure 4.1 Software architecture of the ChemProc System

ChemProc is the software that supports to *draw* the flowsheet. As shown in Figure 4.1, the user must *choose* a symbol through the symbol palette for *placed* into

the flowsheet and *input* detailed information of each symbol to the symbol table. The symbol palette consists of a line symbol palette and an equipment symbol palette. During creation, each symbol can be *changed color*, *cut*, *copied*, *pasted*, *deleted*, *moved*, *grouped* with any symbol, *ungrouped* to separate symbol, *added* new equipment symbol and *modified size* for making the completed flowsheet. Each line symbol is assigned a source and destination equipment symbol by *connected* to any equipment symbol. After the connection, the input and output number of all symbol on the flowsheet must be *verified*. In addition, the user can *manipulate* the created flowsheet file trough the menu.

4.1.1 Identify classes

Extracting the nouns or noun phrases in the processing narrative, the potential objects can be proposed and considered for inclusion in the analysis model:

Potential Object / Class

Computer-Aided..	- Irrelevant. Rejected
Software	- Irrelevant. Rejected
Flowsheet	- The main class of the system. Kept
User	- Irrelevant. Rejected
Symbol	- Useful class. Kept
Symbol palette	- Allows the user to select the symbol for draw. Kept
Detailed information	- Attributes. Rejected
Symbol table	- A class which is needed to edit the specification. Kept
Line palette	- A type of the symbol palette. Kept
Equipment palette	- A type of the symbol palette. Kept

Line - A type of the symbol. Kept

Equipment - A type of the equipment. Kept

Input number - Attribute. Rejected

Output number - Attribute. Rejected

Flowsheet File - Attributes. Rejected

Menu - Allow the user to manipulate the flowsheet. Kept

The above list some of the rejected potential objects will become attributes for those objects that were accepted. And this list is not all classes in the software that additional classes would have to be added to complete the model.

Added classes

Label - The class that needed to manage the caption of each symbol.

Group - A part of flowsheet that point to the grouped line and equipment.

Drawing area - Allows the user to communicate with the flowsheet

Data Dictionary

A data dictionary stores descriptions of each class. These descriptions are intended to be used to understand the system.

Class Name	**Description**
Flowsheet	A unit that would maintain all symbols placed by the user.
Symbol	A symbol manipulates drawing that it is designed to do.
Symbol Table (Alias Table)	The user inputs equipment specification for equipment symbol and input material properties for line symbol through this unit to the equipment and line symbol.

Class Name	Description
Symbol palette	The user selects the symbol image for draw on drawing screen.
Line	This unit which would enable the user to draw the chose line symbol on the palette.
Line palette	The user selects the line type for draw on drawing screen through this unit.
Equipment	This unit which would enable the user to draw the chose equipment symbol on the palette.
Equipment palette	The user selects the equipment type for draw on drawing screen through this unit.
Menu	A unit manages a certain activity that it is designed to do.
Label	This unit is the symbol name label that allows the user to move the symbol name to any position.
Group	A unit that would manages a symbol group on the flowsheet.
Drawing area	A unit which would enable a user to draw the flowsheet.

4.1.2 Create object-to-object relationship

All statement in the process narrative are extracted and considered for establishing the classes relationship by deriving the verbs and verb phrases in the statement.

Identification of classes relationships

ChemProc is uses to draw the flowsheet. - Irrelevant. Ignored

A symbol is chose through the symbol palette. - Irrelevant. Ignored

The symbol is placed into the flowsheet.	- Relationship between symbol and flowsheet. Kept
The user must input detailed information to the table.	- Implied relationship between table and line and between table and equipment. Kept
The line palette and the equipment palette are the symbol palette.	- Relationship between symbol palette and line palette and equipment palette. Kept
The user can be change color, cut, copy, paste, delete, move, group with any symbol, ungroup to separate symbol, add new equipment symbol and modify size the symbol.	- Operation. Ignored
The line symbol is connected to any equipment symbol.	- Operation. Ignored
The input and output number of all symbol on the flowsheet must be verified.	- Operation. Ignored
The user can manipulate the created flowsheet file trough the menu.	- Implied relationship between flowsheet and menu. Kept
Implied relationships	
Label is a part of line.	- Relationship between line and Label. Kept
Label is a part of equipment.	- Relationship between equipment and label. Kept

Group is a part of flowsheet.	- Relationship between group and flowsheet. Kept
Drawing area configures the flowsheet.	- Relationship between drawing area and the flowsheet. Kept
Drawing area monitors the symbol palette.	- Relationship between drawing area and symbol palette. Kept
Images of the symbol image are draw on drawing area.	- Relationship between drawing area and line and equipment. Kept
Line is type of the symbol.	- Relationship between symbol and line. Kept
Equipment is type of the symbol.	- Relationship between symbol and equipment. Kept
The user can new and close the drawing area trough the menu.	- Relationship between menu and drawing area. Kept
The user can show the symbol palette trough the menu.	- Relationship between menu and symbol palette. Kept

Specification of the semantics of the relations

In this step, objects are connected to other objects using named relationships and an overall network of relationship is established.

1. Flowsheet uses symbols.

   Name: Flowsheet manages symbols.

   Multiplicity: Flowsheet: 1 Symbol: n

2. Line is a symbol.

   Name: (irrelevant)

   Multiplicity: (irrelevant)

3. Equipment is a symbol.

   Name: (irrelevant)

   Multiplicity: (irrelevant)

4. Line uses a table.

   Name: The detailed information of Line is inputted via table.

   Multiplicity: Line: 1 Table: 1

5. Equipment uses a table.

   Name: The detailed information of Equipment is inputted via table.

   Multiplicity: Equipment: 1 Table: 1

6. Line uses a label

   Name: Line label is illustrated using a label.

   Multiplicity: Line: 1 Label: 1

7. Equipment uses a label.

   Name: Equipment label is illustrated using a label.

   Multiplicity: Equipment: 1 Label: 1

8. Line uses a drawing area.

   Name: Line is draw on a drawing area.

   Multiplicity: Line: n Drawing area: 1

9. Equipment uses a drawing area.

   Name: Equipment is draw on a drawing area.

   Multiplicity: Equipment: n Drawing area: 1

10. Flowsheet uses a drawing area.

    Name: Flowsheet is communicated via a drawing area.

    Multiplicity: Flowsheet: 1 Drawing area: 1

11. Drawing area uses a symbol palette.

    Name: Drawing area monitors a symbol palette.

    Multiplicity: Drawing area: 1 Symbol palette: 1

12. Line palette is a symbol palette.

    Name: (irrelevant)

    Multiplicity: (irrelevant)

13. Equipment palette is a symbol palette.

    Name: (irrelevant)

    Multiplicity: (irrelevant)

14. Flowsheet has group.

    Name: Flowsheet has group of symbols.

    Multiplicity: Flowsheet: 1 Group: n

15. Flowsheet uses menu.

    Name: Flowsheet is configured by menu.

    Multiplicity: Flowsheet: 1 Menu: 1

16. Symbol palette uses menu.

    Name: Symbol palette is showed by menu

    Multiplicity: Symbol palette: 1 Menu: 1

17. Drawing area uses menu.

    Name: Drawing area is showed and closed by menu

    Multiplicity: Drawing area: 1 Menu: 1

Specify structures and hierarchies

A generalization-specialization structure and a whole-part structure by Coad and Yourdon (11) are used for define the structure of the class and hierarchies and using the notation represented in Figure 4.2. A whole-part structure also has cardinality, as represented by one-to-many or many-to-many. The double-headed arrows represent communication paths between objects contained within the subject references. The detailed structures are shown in Appendix B.

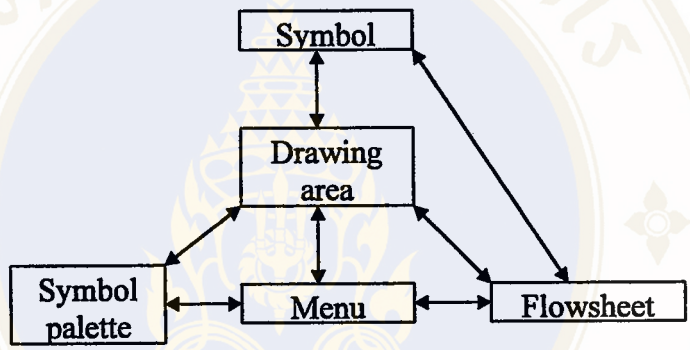


Figure 4.2 Preliminary structure analysis with subject references

4.1.3 Create object behavior model

In this step, all events have been identified and allocated to the object involved. The event flow diagram is developed to represent all events that flow into and out of an object as shown in Figure 4.3 – 4.8.

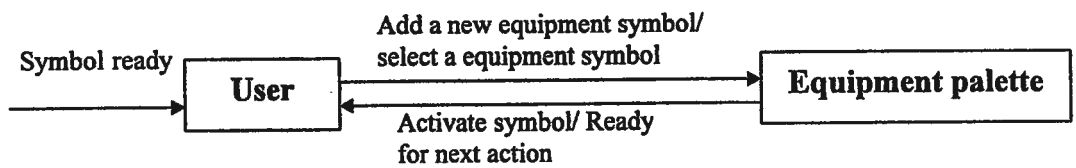


Figure 4.3 A partial event flow diagram for adds and selects a new equipment symbol

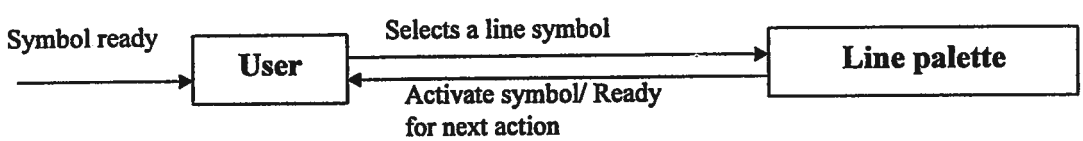


Figure 4.4 A partial event flow diagram for selects a new line symbol

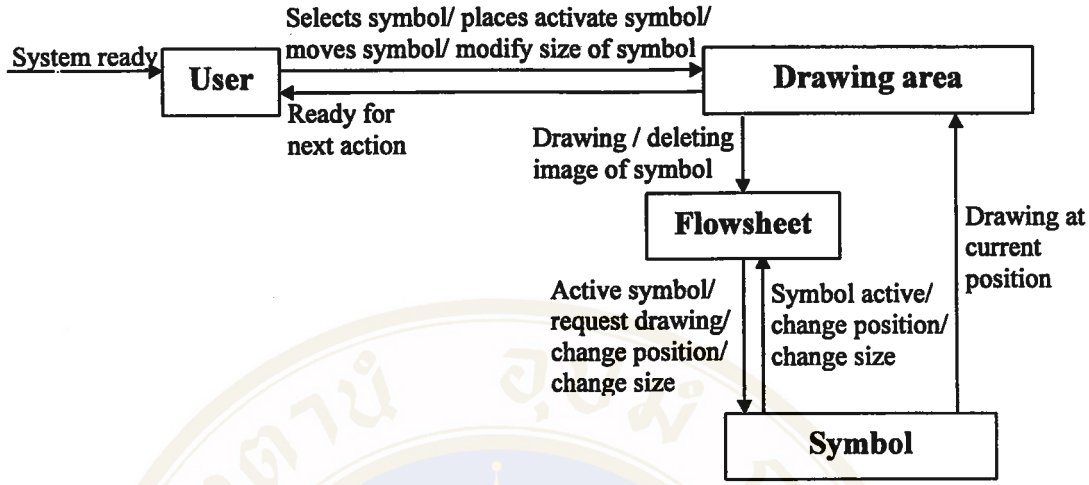


Figure 4.5 A partial event flow diagram for creates symbol and manipulates symbol on flowsheet

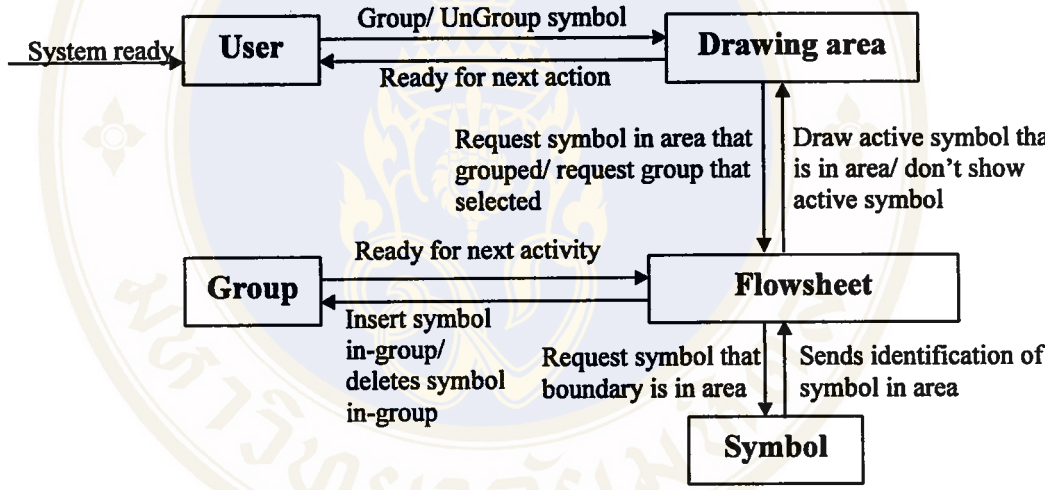


Figure 4.6 A partial event flow diagram for groups and ungroups symbol on flowsheet

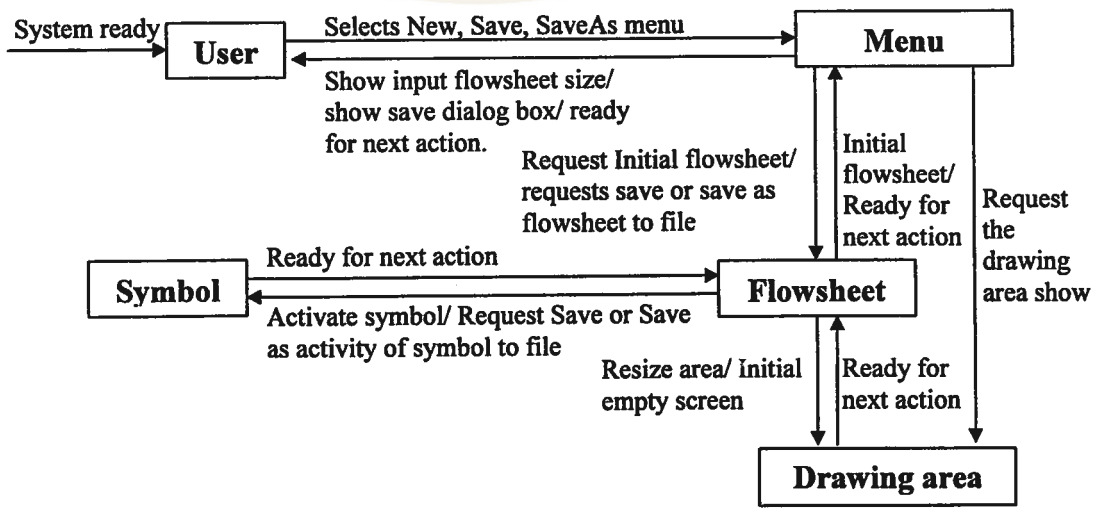


Figure 4.7 A partial event flow diagram for uses new, save, and save as menu

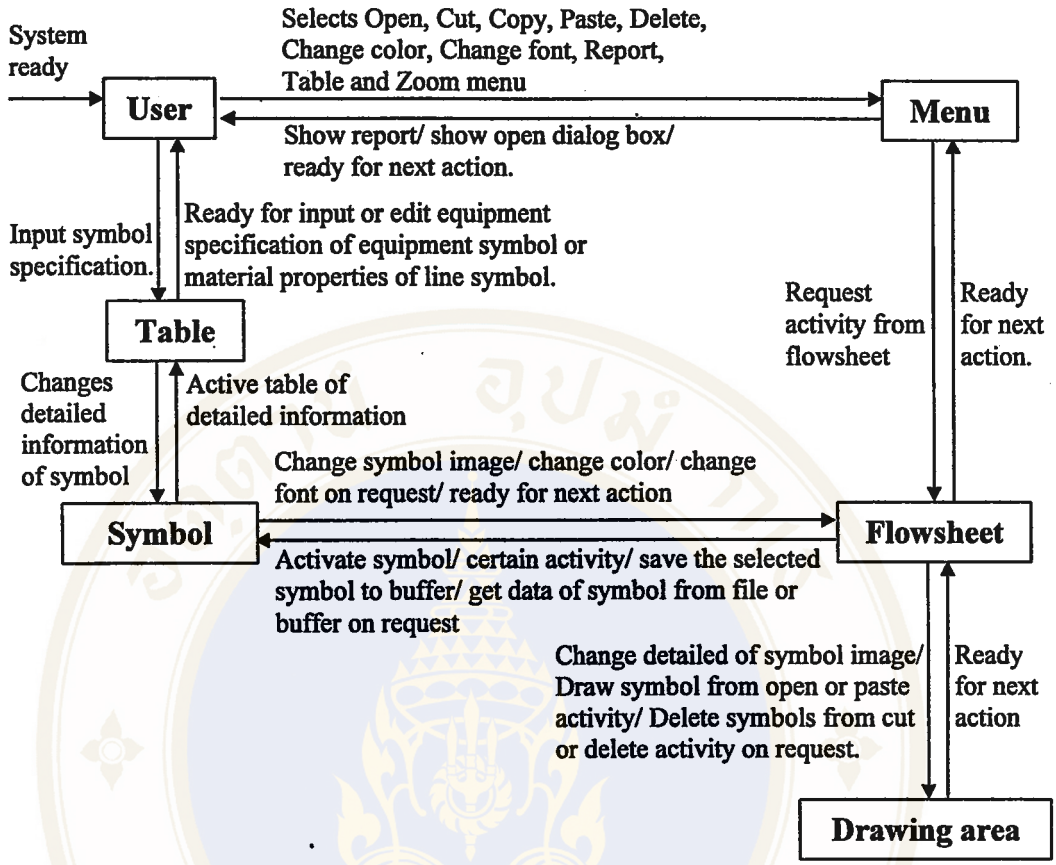


Figure 4.8 A partial event flow diagram for uses menu and input symbol specification

4.2 The result of design step

4.2.1 Subsystem design

The analysis mode is partitioned into the subsystem by their responsibilities.

The subsystem design is shown in Figure 4.9.

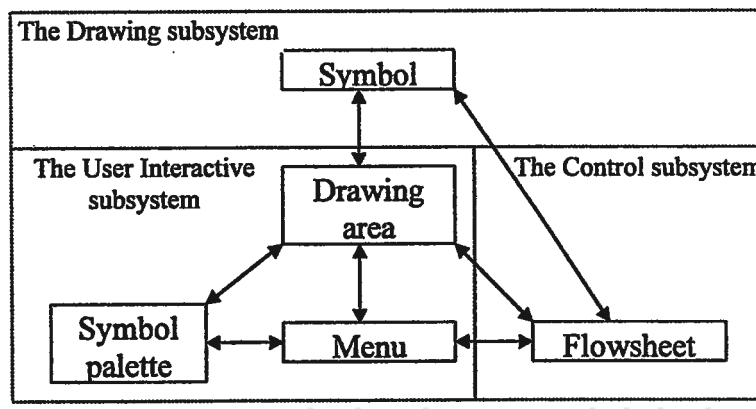


Figure 4.9 A subsystem design

When two subsystems communicate with one another, it is necessary to define the collaborations. Figure 4.10 provides a representation of collaboration.

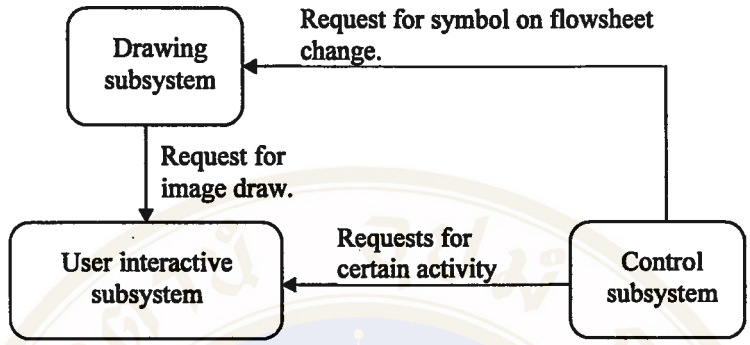


Figure 4.10 Abbreviated subsystem-collaboration graph for the system

4.2.2 Object Design

Object design of the "Drawing" subsystem

Class in this subsystem:

- Symbol
- Line
- Equipment
- Table
- Label

Class : ***Symbol***

Attributes : Scale (type integer)
LInfo (type line class)
SInfo (type equipment class)
Previous and Next (type symbol class)

Operations : Construction
Deconstruction
ClearLinkList
AllocateNodeLine

AllocateNodeSymbol

InsertNode

DeleteNode

CopyNode

Class : ***Line***

Attributes : ID, Type, Scale, disX, disY, StartPoint, SourceID, DestinateID, MaxData, NumData and numComp (type integer)

Bound (type struc of boundary that has Left, Right, Top, Bottom, Width, Height)

Name (type string)

Data (type array of integer)

NameData (type array of string)

Table, TableSpec and CompTable (type table)

Label (type label)

LColor (type color)

Operations : Construction

Deconstruction

SetBound

CheckInputAndOutput

DrawState

Resize

Drawing

Move

Modify

EditTable

DeleteRecord

DeleteCompRecord

EditCompTable

Save

SaveNode

Open

GetDataOnTable

ChangeColor

ChangeFont

Table Specifications

Table: Line

Table for storing the detailed information of Line object

Table 4.1 Column definitions of the line table

Key	Column name	Type	Size
✔	ID	Integer	-
	TypeID	Integer	-
	Type	Text	15
	Name	Text	15
	Component	Integer	-
	Source Symbol Type	Text	15
	Destinate Symbol Type	Text	15
	Source Symbol ID	Integer	-
	Destinate Symbol ID	Integer	-
	Source Point	Integer	-
	Destinate Point	Integer	-
	Left	Integer	-
	Right	Integer	-
	Top	Integer	-
	Bottom	Integer	-
	StartPoint	Integer	-
	DisX	Integer	-
	DisY	Integer	-
	StartX	Integer	-

Table 4.1 (continued)

Key	Column name	Type	Size
	StartY	Integer	-
	EndX	Integer	-
	EndY	Integer	-
	FontName	Text	30
	FontSize	Integer	-
	Color	Text	30

Table Specifications

Table: Component

Table for storing the material property of component in Line object

Table 4.2 Column definitions of the component table

Key	Column name	Type	Size
✔	TableID	Integer	-
✔	CompNo	Integer	-
	SpecNo	Integer	-
	Field0	Text	10
	FieName0	Text	25
	Field1	Text	10
	FieName1	Text	25
	Field2	Text	10
	FieName2	Text	25
	Field3	Text	10
	FieName3	Text	25
	Field4	Text	10
	FieName4	Text	25
	Field5	Text	10
	FieName5	Text	25
	Field6	Text	10

Table 4.2 (continued)

Key	Column name	Type	Size
	FieName6	Text	25
	Field7	Text	10
	FieName7	Text	25
	Field8	Text	10
	FieName8	Text	25
	Field9	Text	10
	FieName9	Text	25

Class : ***Equipment***

Attributes : ID, Type, Scale, InputNo, OutputNo, Input, Output, MaxData and NumData (type integer)
Bound (type struc of boundary that has Left, Right, Top, Bottom, Width, Height)
FileSymbol and Name (type string)
Data (type array of integer)
NameData (type array of string)
Table and TableSpec (type table)
Label (type label)
SColor (type color)

Operations : Construction
Deconstruction
SetBound
CheckInputAndOutput
DrawState
Resize
Drawing
Move
Modify
EditTable

DeleteRecord

DeleteCompRecord

EditCompTable

Save

SaveNode

Open

GetDataOnTable

ChangeColor

ChangeFont

Table Specifications

Table: Equipment

Table for storing the detailed information of Equipment object

Table 4.3 Column definitions of the equipment table

Key	Column name	Type	Size
✔	ID	Integer	-
	TypeID	Integer	-
	Type	Text	15
	Name	Text	15
	Input Number	Integer	-
	Output Number	Integer	-
	Left	Integer	-
	Right	Integer	-
	Top	Integer	-
	Bottom	Integer	-
	FontName	Text	30
	FontSize	Integer	-
	Color	Text	30
	FileSymbol	Text	30
	Spec No	Integer	-

Table 4.3 (continued)

Key	Column name	Type	Size
	Valid Input	Integer	-
	Valid Output	Integer	-
	Field0	Integer	-
	FieName0	Text	25
	Field1	Integer	-
	FieName1	Text	25
	Field2	Integer	-
	FieName2	Text	25
	Field3	Integer	-
	FieName3	Text	25
	Field4	Integer	-
	FieName4	Text	25
	Field5	Integer	-
	FieName5	Text	25
	Field6	Integer	-
	FieName6	Text	25
	Field7	Integer	-
	FieName7	Text	25
	Field8	Integer	-
	FieName8	Text	25
	Field9	Integer	-
	FieName9	Text	25

Class : ***Table***

Attributes : TabSheet (type TabSheet)

DBEdit (type DBEdit)

Panel (type Panel)

Comp (type array of boolean)

NameData (type array of string)

LineActive (type boolean)

numData and numComp (type integer)

Operations : Construction

SetTabSheet

Show

CloseTable

Class : ***Label***

Attributes : Bound (type struc of boundary that has Left, Right, Top, Bottom)

PenColor (type color)

FontName and Text (type string)

FontStyle (type font style)

FontSize, Scale, DisX and DisY (type integer)

Operations : Construction

SetBound

Move

Drawing

ChangeText

ChangeColor

ChangeFont

SetFontSize

Object design of the "Control" subsystem

Class in this subsystem:

- Flowsheet
- Group

Class : ***Flowsheet***

Attributes : Type, SymbolNo, LineNo and Scale (type integer)

LLine, LSymbol, LBuff and SBuff (type link list of symbol class)

Group and LGroup (type Group class)

InGroup (type Boolean)

GroupNo (type integer)

Operations : Construction

Deconstruction

SetCursor

DeleteSymbol

CleanNodeImage

Drawing

DrawState

Move

Modify

SaveFlowsheet

OpenFlowsheet

ShowReport

CopyToBuff

PasteToFlowsheet

ResizeFlowsheet

EditTable

CloseTable

UpdateData

GroupSymbol

SetGroup

UnGroup

CleanGroup

ChangeColor

ChangeFont

Class : ***Group***

Attributes : Scale, ID, Type and GroupID (type integer)

Previous and Next (type Group class)

Table (type Table)

Operations : Construction
CheckInGroup
InsertTo
DeleteNode

Table Specifications

Table: Group

Table for storing the detailed information of Group object

Table 4.4 Column definitions of the group table

Key	Column name	Type	Size
✔	GroupID	Integer	-
	SymbolID	Integer	-
	SymbolType	Integer	-

Object design of the "User Interactive" subsystem

Class in this subsystem:

- Menu
- Symbol palette
- Line palette
- Equipment palette
- Drawing area

Class : ***Menu***

Attributes : Flowsheet (type flowsheet class)
FileName and Font_Name (type string)
Font_Size (type integer)
Font_Style (type FontStyles)
New, Open, Save, SaveAs, Close, Print, Exit, Cut, Copy, Paste,
Delete, Group, Ungroup, Font, Bold, Italic, Underline, Change Color,

Change Specification, Symbol, Table, Zoom in, Zoom out, Report, Help Topics and About (type menu item)

Operations : Construction

ConfirmSaveFile

SetMenu

NewClick, OpenClick, SaveClick, SaveAsClick, CloseClick, PrintClick and ExitClick

CutClick, CopyClick, PasteClick, DeleteClick, GroupClick and UngroupClick

FontClick, BoldClick, ItalicClick and UnderlineClick

Change ColorClick and Change SpecificationClick

SymbolClick, TableClick, Zoom inClick, Zoom outClick and ReportClick

Help TopicsClick and AboutClick

Class : ***Symbol palette***

Attributes : Table (type table)

Type (type integer)

Operations : Show

Table Specifications

Table: Symbol Specification

Table for storing the specification of each symbol type.

Table 4.5 Column definitions of the symbol specification table

Key	Column name	Type	Size
✔	SymbolType	Integer	-
	Input Number	Integer	-
	Output Number	Integer	-
	Specification Number	Integer	-
	Name1	Text	25

Table 4.5 (continued)

Key	Column name	Type	Size
	Name2	Text	25
	Name3	Text	25
	Name4	Text	25
	Name5	Text	25
	Name6	Text	25
	Name7	Text	25
	Name8	Text	25
	Name9	Text	25
	Name10	Text	25
	Specification Name	Text	25

Class : ***Line palette***

Attributes : Point1, Line, Instrument, Orifice, GateValve, GlobeValve, CheckValve, ReliefValve and ControlValve (type button)

Operations : Click

Class : ***Equipment palette***

Attributes : Point2, HeatEx, WaterCooler, SteamHeater, Reboiler, Jcondenser, Condenser, Absorber, Stripper, MixerTank, LiquidTank, PressureTank, Jkettle, Reaction, Ppump, Cpump, Compressor, Furnace, Distillation, Bfractionating, Sfractionating,Separator, Vvessel, Hvessel and Other symbol (type button)

Operations : Click
OtherSymbolClick

Class : ***Drawing area***

Attributes : InitialX, InitialY, Type, State, ZoomInd (type integer)

Operations : Show

FormDestroy

MouseDown

MouseMove

MouseUp

ShowPosition

ZoomInOrOut

4.2.3 User Interface design

The interfaces of class in the user interactive subsystem are design following the C++ builder form. That is, the forms are divided into the menu form, the symbol palette form that consists of a line palette and an equipment palette and the drawing area form. Therefore, the software utility forms are also design such as the table form of the line and the equipment, the symbol specification table form, the report and the output report. The user interface design is shown in Appendix B.

4.2.4 Procedural design

The procedural of each class operations is illustrated in Appendix C.

CHAPTER V

DISCUSSION

In this chapter, the object-oriented technologies and C++ builder that used in this research methodology are discussed.

For this research method, the object-oriented technologies by Pressman, Roger S. approach are used to design and develop the ChemProc system. Based on the object-oriented technologies, the software can be adapted easier and fewer side effects when changes.

The first step in object-oriented approach is object-oriented analysis (OOA) that the classes are identified, created relationship and defined behavior. So the developer can clearly understand the requirement and know that what must be built.

The next step is object-oriented design (OOD) that uses the analysis model for the design in the C++ builder application. Which the designed component should be reusable and flexible. Thus, this step is the difficult task and spends a lot of time.

After that, the results in the design step are implemented in the C++ builder. The C++ builder is the tool, which support encapsulation, data hiding, reuse and inheritance that are the characteristics of object-oriented technology. As the results, the implementation is easy step. However, the C++ builder characteristics spend a lot of time on studying the component, properties and delegation model for building the program.

CHAPTER VI

CONCLUSION

This paper illustrated each step in these research methods that using object-oriented approach for designs and develops the software on the C++ builder. Thus the developer that interested in object-oriented technology for develop the system on C++ builder can use this research as a guideline. For this chapter, the results of this research are concluded from the objective, the expected results and the software working.

To achieve the objective of this research, the ChemProc is designed and developed for drawing the chemical process flowsheet. This software has the three main functions: the drawing function, the controlling function and the utility function.

- The drawing function creates or deletes the symbol on the flowsheet and changes position and size of the selected symbol.
- The controlling function manages all symbols on the flowsheet. That is, the symbols placed on the flowsheet are linked to another symbol on the flowsheet for managing the information of each symbol. Furthermore, the equipment symbol can be connected to any line symbol.
- The utility function provides user supports for ease of work with the software and supports for creating the completed flowsheet. Such as, the program has function to manipulate the flowsheet file or to update the symbol specification data in the symbol table. The symbol specification

data in the software contains data for 10 options specification that can be defined by user. That the user should define the symbol specification type at the initial drawing the flowsheet.

The ChemProc software can help the chemical engineer to draw the flowsheet in some parts that following the expected results:

- Save time in drawing the flowsheet.
- Easy to create each symbol on the flowsheet.
- Easy to manipulate flow. That is, the user can move any symbol on the flow and will automatically correspond to the moved symbol.
- New symbols can be easily added to the software.
- Reduce the human error by automatic validate information in the input and output symbol.
- Each symbol contains the specification data.

However, there are some disadvantages of the software. For the ChemProc, the change color function can not be used with the symbols defined by user. In addition to, the size of the symbol table that contains the specification data of each symbol is reasonably large. Because each symbol has many essential detailed specifications.

In summary, the ChemProc is developed, which achieve the objective and expected results.

Recommendation for future develops

Base on the ChemProc, the another functions can be added in this application such as:

- The flowsheet can be processed to produce a particular product or other relevant product.
- The flowsheet can be a reasonable approximation of the best economic flowsheet for production.

REFERENCES

1. William L.Luyben, Leonard A.Wenzel. Chemical process analysis: mass and energy balances. Prentice Hall, 1988
2. Alan Heaton. An introduction to industrial chemistry. Third edition, Blackie Academic & Professional, 1996
3. George T. Austin. Shreve's chemical process industries. McGraw-Hill Companies Inc., 1984.
4. Douglas, James M. Conceptual design of chemical processes. McGraw-Hill Companies Inc., 1988.
5. Felder, Richard M. and Rousseau, Ronal W. Elementary principles of chemical processes. John Wiley & Sons 1986.
6. Peters, Max S. and Timmerhaus, Klaus D. Plant design and economics for chemical engineer. McGraw-Hill Inc., 1991.
7. ChemPute Software. ChemCad – Process flowsheet simulation [Online] 1999. Available from: http://www.chempute.com/chemcad.htm [Accessed 2000 April 5].
8. Pressman, Roger S. Software engineering. McGraw-Hill Companies Inc., 1997.
9. Taylor, D.A., Object-oriented technology: A manager's Guide. Addison-Wesley, 1990.
10. Wirfs-Brock R., B. Wilkerson, and L. Weiner, Designing object-oriented software. Prentice-Hall, 1990.

11. Coad, P., and E. Yourdon. Object-oriented analysis. 2nd edition, Prentice-Hall, 1991.

12. Rambauge, J. et al. Object-oriented modeling and design. Prentice-Hall, 1991.

13. McDaniel, George. IBM dictionary or computing. McGraw-Hill Companies Inc., 1994.

14. Jan Goyvaerts. HelpScribble [Online] 2000. Available from: http://www.JustGreatSoftware.com [Accessed 2000 February 9].

APPENDIX A

FLOWSHEET

In this chapter, to illustrate some flowsheet and symbol that used in the Computer-Aided Chemical Process Flowsheet. The symbol is divided into the line symbol that shown in Figure A.1 and the equipment symbol that shown in Figure A.2.

Symbol	Description	Symbol	Description
	New lines		Globe valve
	Instrument lines		Check valve
	Orifice		Relief valve
	Gate valve		Control valve

Figure A.1 Some type of flow sheet symbols, particularly for detailed equipment flow sheet (Peters and Timmerhaus, 1991)

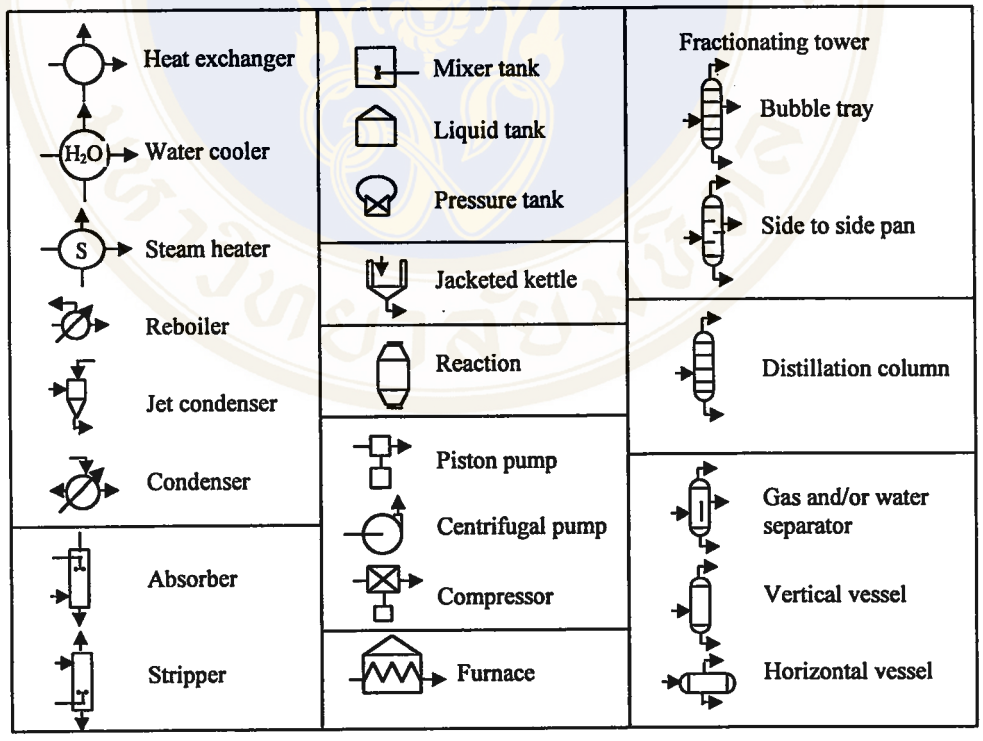


Figure A.2 Some type of equipment symbols (Peters and Timmerhaus, 1991)

Example of flowsheet

This section has some flowsheet that used as example for drawing and used in the validation test step.

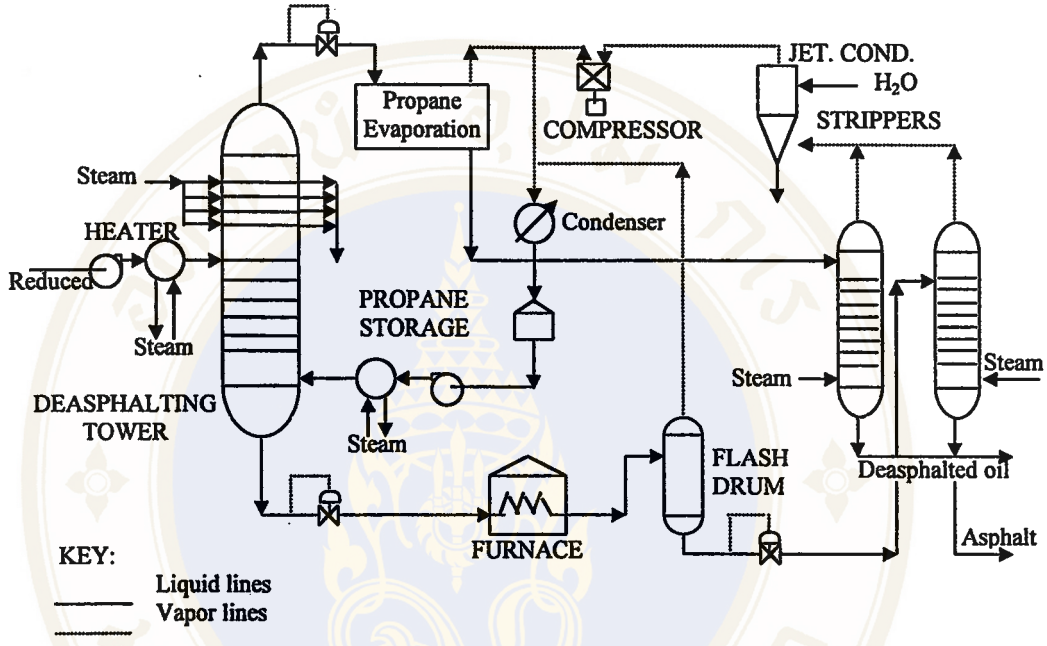


Figure A.3 Flowchart for lube oil deasphalting (George, 1984)

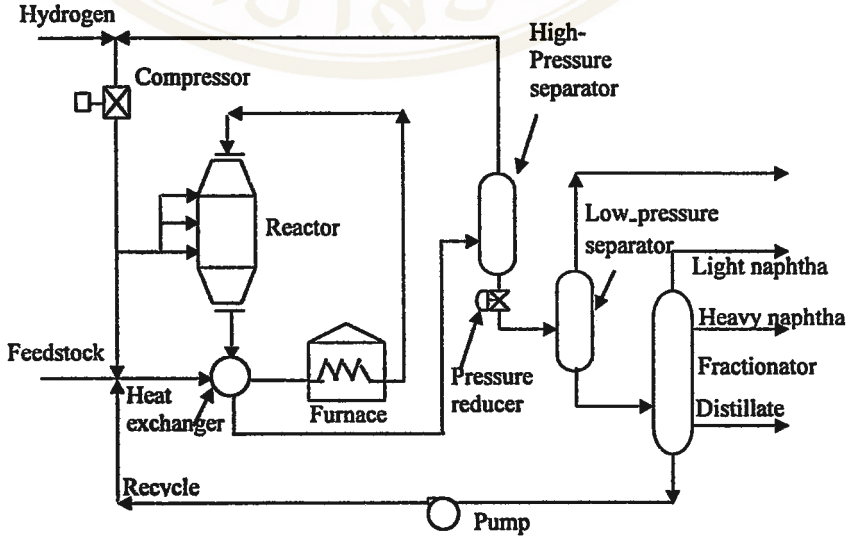


Figure A.4 Hydrocracking (George, 1984)

Computer-aided chemical process flowsheet has many symbol types that support flowsheet in the Figure A.3 and Figure A.4 as shown in table A.1.

Table A.1 Symbol types checklist

Symbol type	Symbol in figure A.3	Symbol in figure A.4
Control valve	3	1
Heat exchanger	2	1
Water cooler	-	-
Steam heater	-	-
Reboiler	-	-
Jet condenser	1	-
Condenser	1	-
Absorber	-	-
Stripper	-	-
Mixer tank	-	-
Liquid tank	1	-
Pressure tank	-	-
Jacketed kettle	-	-
Reactor	-	1
Piston pump	-	-
Centrifugal pump	2	1
Compressors	1	1
Furnace	1	1
Fractionating tower: bubble trays	1	1
Fractionating tower: side-to-side pans	-	-
Distillation column	2	-
Gas and/or water separator	-	2
Vertical vessels	1	-
Horizontal vessels	-	-

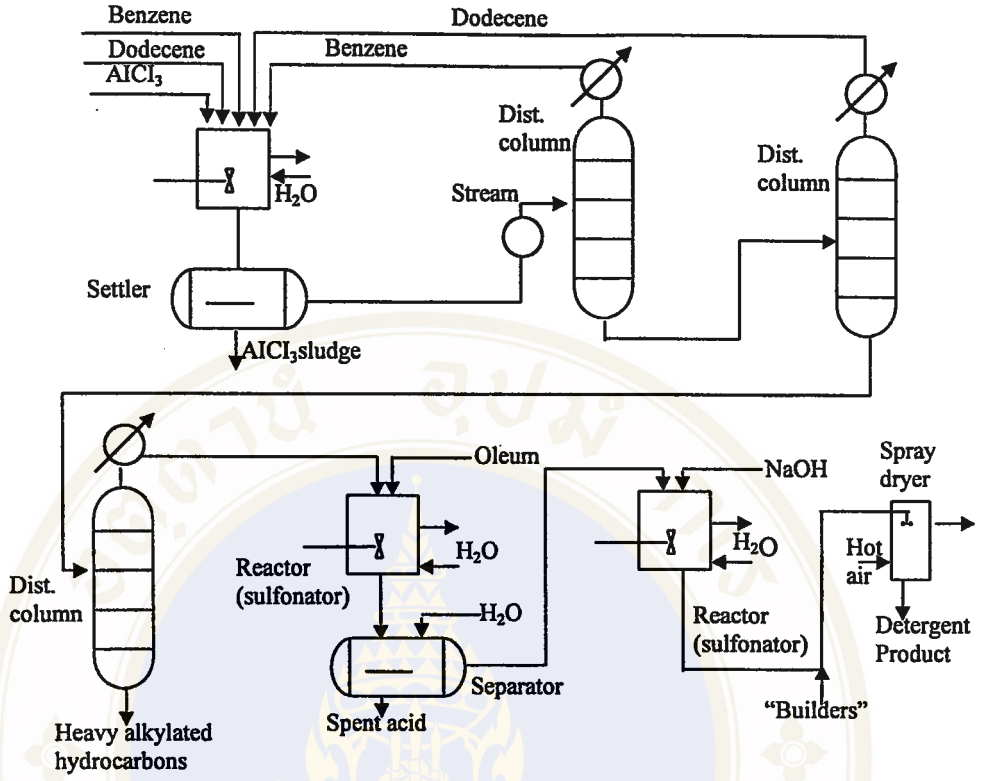


Figure A.5 Qualitative flow diagram for the manufacture of sodium dodecylbenzene sulfonate (Peters and Timmerhaus, 1991)

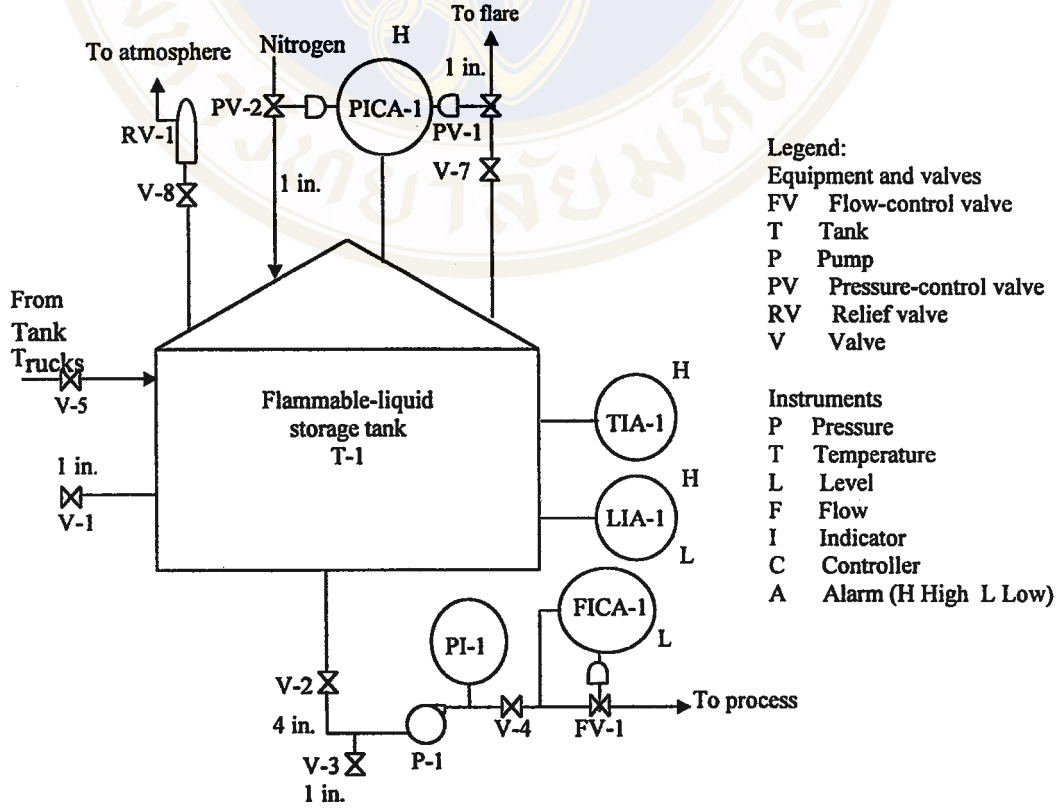


Figure A.6 Diagram used in HAZOP example (Peters and Timmerhaus, 1991)

APPENDIX B

CLASS HIERARCHIE AND USER INTERFACE DESIGN

This chapter illustrates the class hierarchies in the analysis step and the designed screen in the user interface design step.

The subclass structure in the subject references

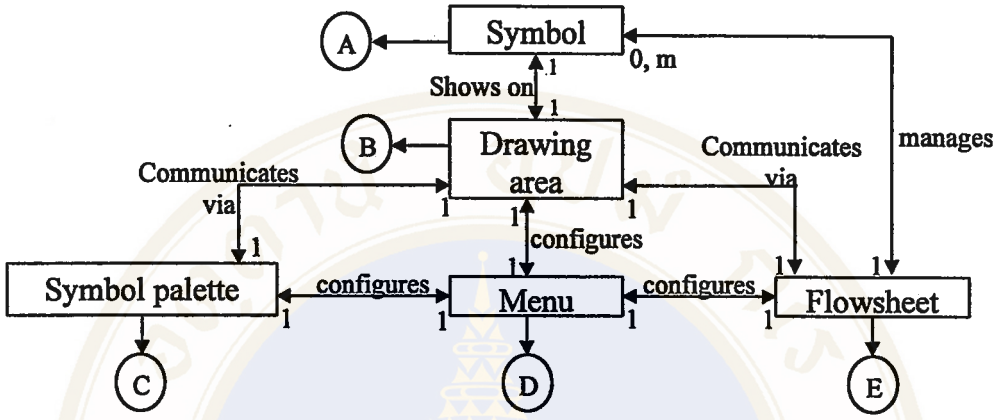


Figure B.1 Preliminary structure analysis with subject references

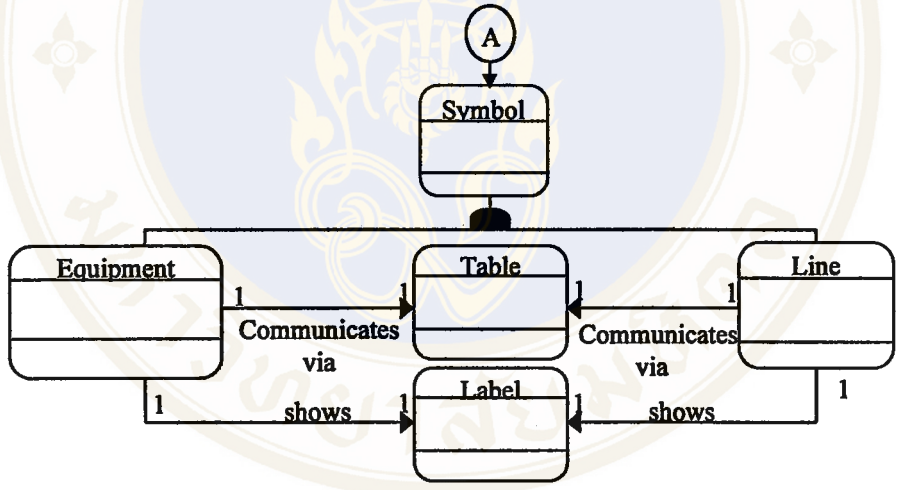


Figure B.2 The subclass structure in the symbol subject

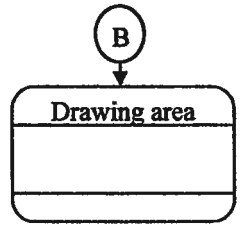


Figure B.3 The subclass structure in the drawing area subject

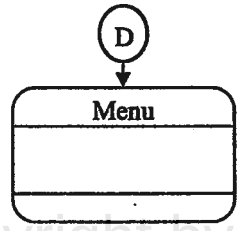


Figure B.4 The subclass structure in the menu subject

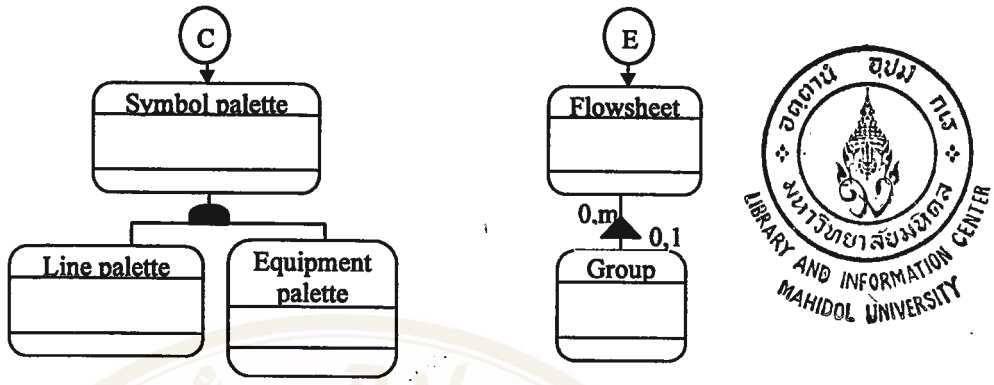


Figure B.5 The subclass structure in the symbol palette subject and the flowsheet subject

THE USER INTERFACE DESIGN

At the initial the ChemProc software, the starting screen is shown that illustrated in Figure B.6. This screen consists of the menu form that shown in Figure B.7 and B.8, the symbol palette form that shown in Figure B.9 and the drawing area form that shown in Figure B.10.

The starting screen

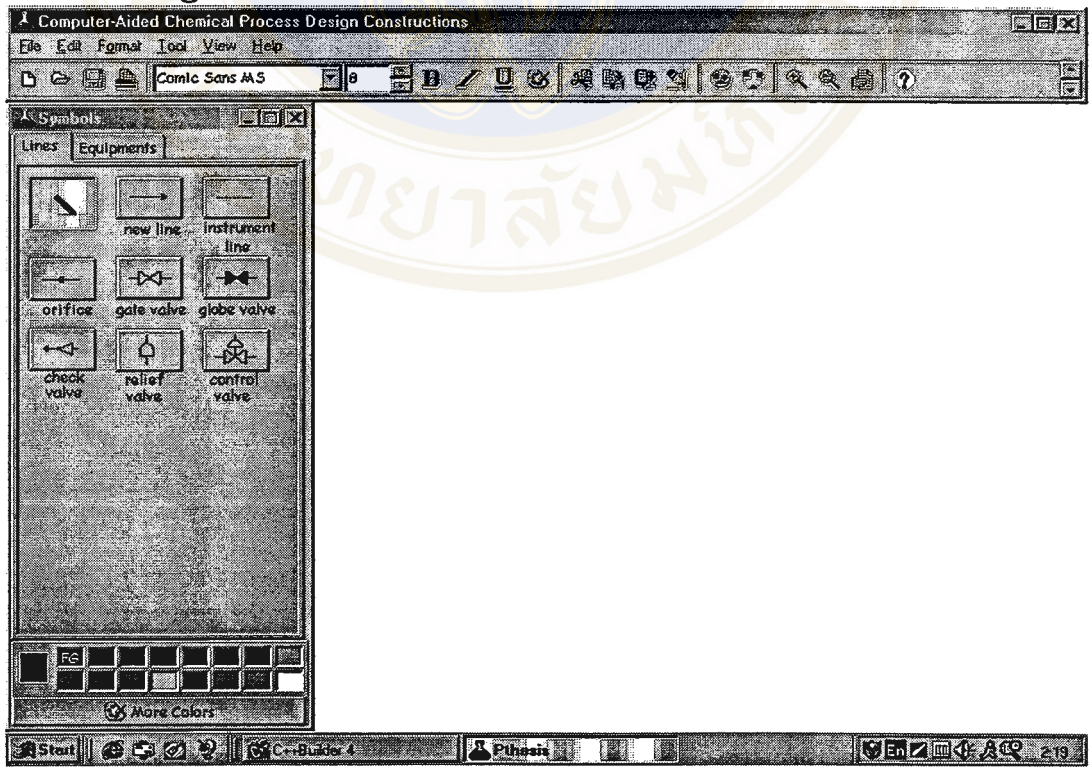


Figure B.6 The starting screen

The menus form

Figure B.7 The menus form

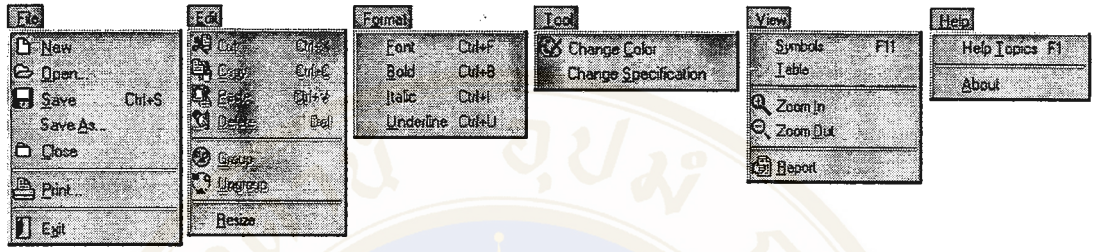
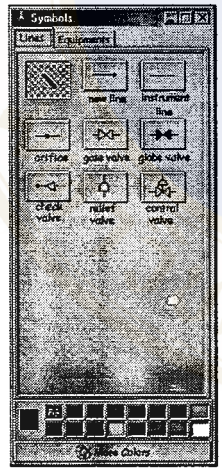


Figure B.8 The menu items in the menu form

The symbol palette form

This form is divided into the line palette and equipment palette.

1. Line palette



2. Equipment palette

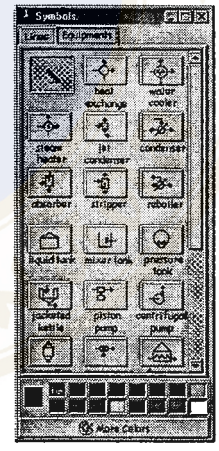


Figure B.9 The symbol palette form

The drawing area form

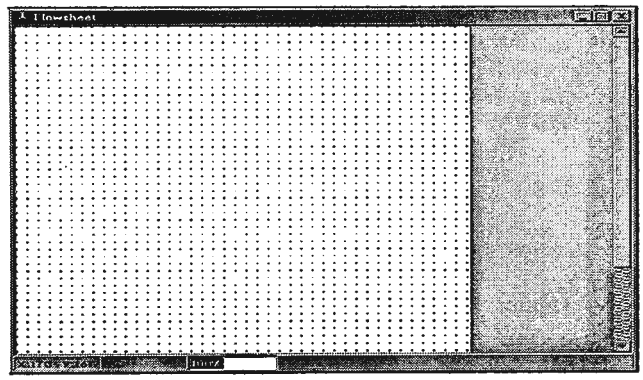


Figure B.10 The drawing area form

The table form

This form is shown when the user want to edit or update the technical specification of symbol on the flowsheet that divided into two type of symbol. Accordingly, this form divided into the line table and equipment table.

1. The line table consists of the line specification and the component specification.

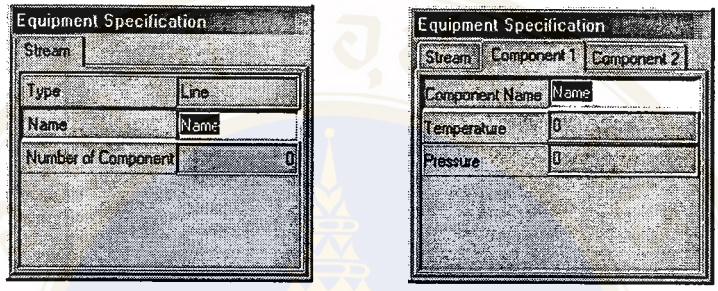


Figure B.11 The table form in the line specification page

2. The equipment table consists of the equipment specification.

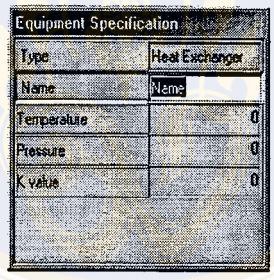


Figure B.12 The table form in the equipment specification page

The symbol specification form

This form is the table that the user can edit the specification type of each symbol.

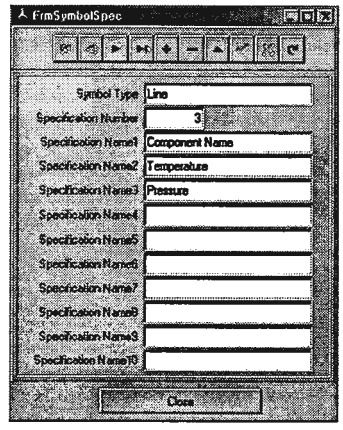


Figure B.13 The symbol specification table

The output report

This part is divided into the report of the line symbol and the equipment symbol that illustrated the detailed information. Another type of report is the flowsheet image that shown in the Figure B.14 using the flowsheet in the example of flowsheet in Figure A.4 at the Appendix A.

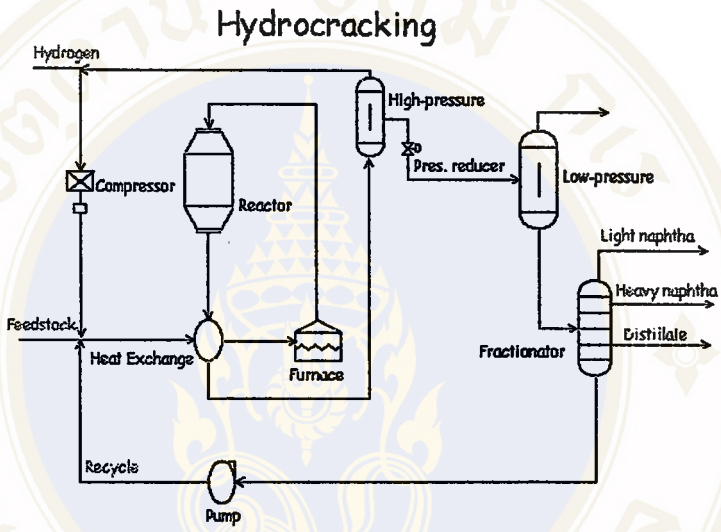


Figure B.14 The example of flowsheet is created by the ChemProc

The report form

The report form illustrates the technical specification of line and equipment that edited or updated by user.

1. The report form of the line symbol.

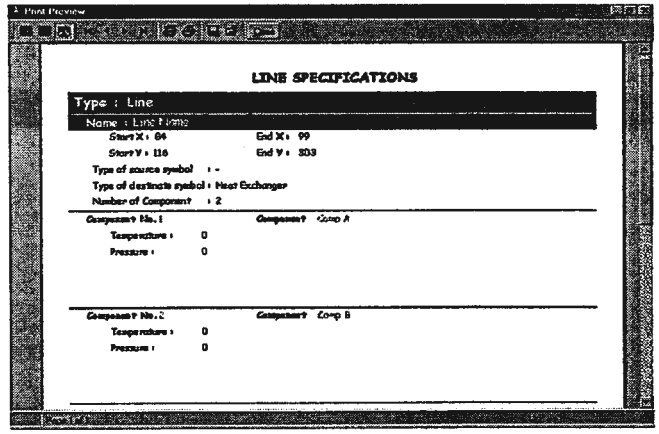


Figure B.15 The line report form

2. The report form of the equipment symbol.

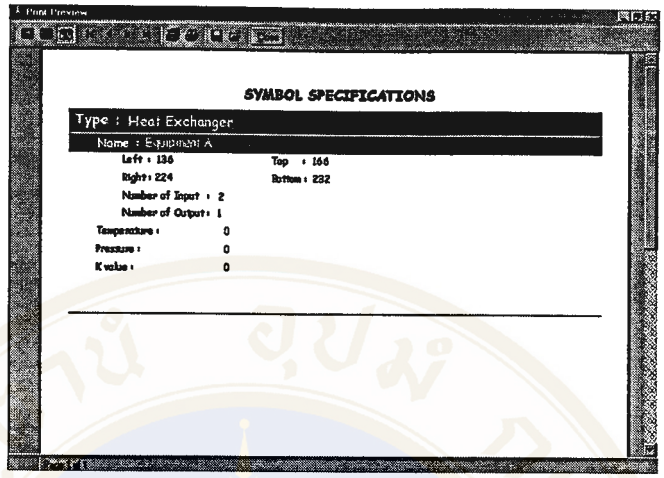


Figure B.16 The equipment report form

APPENDIX C

PROCEDURAL DESIGN

In this chapter, to illustrate the procedural of each operation in all classes. The procedural designed is divided into to 3 part following the subsystem:

1. The drawing subsystem
2. The main subsystem
3. The user interactive subsystem

Procedural design of the "Drawing" subsystem

Class in this subsystem:

- Symbol
- Line
- Equipment
- Table
- Label

The operation in the symbol class

Construction

Inputs : Type, No, Zoom (type integer)
Outputs : None
Local : None
Purpose : To initialize symbol class.
Algorithm : Assign Scale = Zoom
Assign Previous = NULL
Assign Next = NULL
Assign SInfo =NULL
Assign LInfo = NULL
If Type is line type then Create LInfo
Else Create Sinfo
End If

Deconstruction

Inputs : None
Outputs : None
Local : None
Purpose : To destroy the created object class.
Algorithm : Delete LInfo and Delete Sinfo

ClearLinkList

Inputs : LinkList (type symbol class)
Outputs : None
Local : Tmp (type symbol class)
Purpose : To delete all nodes in the LinkList.
Algorithm : Go to the first node of LinkList
Repeat
Assign Tmp = LinkList
Assign LinkList = Next of LinkList
Delete Tmp
Until LinkList is end

InsertNode

Inputs : Link, Node (type symbol class)
Outputs : None
Local : None
Purpose : To insert a node to the Link
Algorithm : Go to the first node of Link
Repeat
If Type of Node < Type of the current node of Link then
Copy Node from Previous of current node of Link
Return
Else Go to the next node of Link
End If
Until the current node of Link is the last node
Copy Node from Next of current node of Link

DeleteNode

Inputs : Node (type of symbol class)
Outputs : None
Local : Tmp (type symbol class)
Purpose : To delete a current node of link list of symbols
Algorithm : If a link list has one node then Delete Node
End If
If a current node is the first node of link list then
Assign Tmp = Node

Assign Node = Next of Node
Assign Next of Tmp = NULL
Delete Tmp
End If
If a current node is the last node of link list then
Assign Tmp = Node
Assign Node = Previous of Node
Assign Previous of Tmp = NULL
Delete Tmp
End If
Assign Tmp = Node
Assign Node = Next of Node
Assign Previous of Node = Previous of Tmp
Assign Previous of Tmp = NULL
Assign Next of Tmp = NULL
Delete Tmp

The operation in the Line class

Construction

Inputs : Id, type, Zoom (type integer)
Outputs : None
Local : None
Purpose : To initial the line class
Algorithm : Assign ID = id
Assign Type = type
Assign Scale = Zoom
Assign disX = 0
Assign disY = 0
Assign MaxData = 10
Assign numComp = 0
Assign Data[0..MaxData] = "0",
Assign Name = "Name"
Assign LColor = the current pen color
Create Label
Create Table that table name is "Line"
Create CompTable that table name is "Comp"
Create TableSpec that table name is "SymbolSpec"
Assign NumData by retrieve from TableSpec at Field Name "Specification Number"
Assign NameData by retrieve from TableSpec

Deconstruction

Inputs : None

Outputs : None
Local : None
Purpose : To destroy the created object class
Algorithm : Delete Table and Delete CompTable

SetBound

Inputs : X1, Y1, X2, Y2 (type integer)
Outputs : None
Local : None
Purpose : To set the boundary of line
Algorithm : Assign StartX = X1, StartY = Y1, EndX = X2 and EndY = Y2
If X1 < X2 then Assign Bound.Left = X1 and Bound.Right = X2
Else Assign Bound.Left = X2 and Bound.Right = X1
End If
If Y1 < Y2 then Assign Bound.Top = Y1 and Bound.Bottom = Y2
Else Assign Bound.Top = Y2 and Bound.Bottom = Y1
End If
If StartX = Bound.Left Then
If StartY = Bound.Top Then Assign StartPoint = 1
Else Assign StartPoint = 2
End If
Else If StartY = Bound.Top Then Assign StartPoint = 3
Else Assign StartPoint = 4
End If
End If
Assign Bound.Width = Bound.Right – Bound.Left
Assign Bound.Height = Bound.Bottom = Bound.Top
Call SetBound in Label

CheckInputAndOutput

Inputs : None
Outputs : None
Local : None
Purpose : To show that source and destination of line has equipment or not
Algorithm : If SourceID = 0 of DestinateID = 0 then Assign color of Label = "Red"
Else Assign color of Label = the same color as line
End If

DrawState

Inputs : None
Outputs : None
Local : None
Purpose : To show that the line is activate.
Algorithm : Set color = "Teal"

Draw the rectangle at control point.

Resize

Inputs : Zoom (type integer)
Outputs : None
Local : OldScale (type integer)
Purpose : To resize the line using percentage.
Algorithm : Assign oldScale = Scale
Assign Scale = Zoom
Assign StartX = (StartX * Scale)/oldScale
Assign EndX = (EndX * Scale)/oldScale;
Assign StartY = (StartY * Scale)/oldScale;
Assign EndY = (EndY * Scale)/oldScale;
Assign disX = (disX * Scale)/oldScale;
Assign disY = (disY * Scale)/oldScale;
Call SetBound with StartX, StartY, EndX, EndY
Call Drawing

Drawing

Inputs : None
Outputs : None
Local : Ver (type boolean)
Purpose : To draw the selected line type.
Algorithm : If Color is not White then
Assign the current pen color = LColor
Assign the current font color = LColor
End If
If Type is instrument line then
Assign the current pen style = Dot
Draw line from start position to end position
Assign the current pen style = Solid
Else Draw line from start position to end position
End If
Call CheckInputAndOutput
Drawing line image.

Move

Inputs : X1, Y1, X2, Y2 (type integer)
Outputs : Type boolean
Local : left, top, right, bottom (type integer)
Purpose : To move a current line or a line label to new position.
Algorithm : If the label is moved then
Return Move in Label with X1, Y1, X2, Y2
End If

```
Assign left = X2 - (X1 – Bound.Left)
Assign top = Y2 - (Y1 – Bound.Top)
Assign right = X2 + (Bound.Right - X1)
Assign bottom = Y2 + (Bound.Bottom - Y1)
If StartX is Bound.Left then Assign StartX = left and EndX = right
Else Assign StartX = right and EndX = left
End If
If StartY is Bound.Top then Assign StartY = top and EndY = bottom
Else Assign StartY = bottom and EndY = top
End If
Call SetBound with left, top, right, bottom
Call Drawing
Return true
```

Modify

Inputs : X1, Y1, X2, Y2 (type integer)
Outputs : Type integer
Local : None
Purpose : To modify the sizes of line at start position or end position.
Algorithm :

```
If (X1, Y1) is start position of line then
    Call SetBound with X2, Y2, EndX, EndY
    Return 1
End If
If (X1, Y1) is end position of line then
    Call SetBound with StartX, StartY, X2, Y2
    Return 2
End If
If (X1, Y1) is middle position of line then
    If Bound.Width > Bound.Height then
        Assign disX = X2 – Bound.Left
        Assign Bound.MidX = X2
        If StartX = Bound.Right then Assign disX = Bound.Width – disX
        End If
    End If
Else
    If Bound.Width > Bound.Height then
        Assign disY = Y2 – Bound.Top
        Assign Bound.MidY = Y2
        If StartY = Bound.Bottom then
            Assign disY = Bound.Height – disY
        End If
    End If
End If
Call Drawing
Return 3
```

EditTable

Inputs : Show (type integer)
Outputs : None
Local : None
Purpose : To prepare and edit the detailed information.
Algorithm : If Table has a record of this line then If Show = 1 then
Call Show in Table for edit detailed information
Return
End If
Insert a new record to Table.
Insert all data of line into Table (such as ID, Type, Name, ...)
If Show = 1 then Call Show in Table for edit detailed information
End If

EditCompTable

Inputs : CompNum (type integer)
Outputs : None
Local : curComp, i (type integer)
Purpose : To prepare and edit the component material property of line.
Algorithm : Assign curComp = CompNum+1;
If CompTable has a record of this component and this line then Return
End If
Insert ID to CompTable at field name "TableID"
Insert CompNo to CompTable at field name "CompNo"
From i = 0 until i = MaxData
Loop Insert Data[i] to CompTable
Insert NameData[i] to CompTable
End Loop

DeleteRecord

Inputs : None
Outputs : None
Local : NumC, i (type integer)
Purpose : To delete the record of this line from Table
Algorithm : If Table has not a record of this line then Return
End If
Go to a record of this line
Assign NumC by retrieve from Table at field name "Component Number"
From i = 0 until i > NumC
Loop Call DeleteCompRecord with i
End Loop
Delete Table at this record

DeleteCompRecord

Inputs : NumComp (type integer)
Outputs : None
Local : None
Purpose : To delete the record of this component and this line from CompTable
Algorithm : If CompTable has not a record of this component and this line then
Return
End If
Go to a record of this component and this line
Delete CompTable at this record

Save

Inputs : FileName (type string)
Outputs : None
Local : TmpTable (type Table)
Purpose : To copy the Table to the Table of file
Algorithm : If Table has not a record of this line then Return
End If
Insert the detailed information that can not edit by user
Create TmpTable that table name is FileName + "Line"
Assign TmpTable = Table
If CompTable has not a record of this line component then Return
End If
Create TmpTable that table name is FileName + "Comp"
Assign TmpTable = CompTable

SaveNode

Inputs : FileName (type string)
Outputs : None
Local : None
Purpose : To save the detailed information of line in the Table
Algorithm : If Table has not a record of this line then Return
End If
Change the table name of Table to FileName + "Line"
Call EditTable with 0 (don't show the table for edit)

Open

Inputs : FileName (type string)
Outputs : None
Local : TmpTable (type Table)
Purpose : To copy a table of file to the Table and the CompTable
Algorithm : Assign TmpTable = table that table name is FileName + "Line"
If TmpTable is not exist then Return

End If
Assign Table = TmpTable
Assign TmpTable = table that table name is FileName + "Comp"
If TmpTable is exist then Assign CompTable = TmpTable
End If

GetDataOnTable

Inputs : T (type Table)
Outputs : None
Local : left, right, top, bottom (type integer)
Purpose : To retrieve the detailed information from the table.
Algorithm : Assign all detailed information of line from Table. (Such as ID in field name "ID", Name in field name "Name", numComp in field name "Component Number".)
Call SetBound with left, top, right, bottom
Call ChangeText in Label with Name

ChangeColor

Inputs : C (type Color)
Outputs : None
Local : None
Purpose : To change color of a selected line.
Algorithm : Assign LColor = C
Call ChangeColor in Label with C
Call Drawing

ChangeFont

Inputs : FName (type string), FSize (type integer), FStyle (type FontStyle)
Outputs : None
Local : None
Purpose : To change font of a line label.
Algorithm : Call ChangeFont in Label with FName, FSize, Fstyle

The operation in the Equipment class

Construction

Inputs : Id, type, Zoom (type integer)
Outputs : None
Local : None
Purpose : To initial the equipment class
Algorithm : Assign ID = id, Type = type, Scale = Zoom, MaxData = 10,

Data[0..MaxData] = 0, Name = "Name", Scolor =the current pen color,
Create Label, Create Table that table name is "Symbol"
Assign FileSymbol = "-"
Create TableSpec that table name is "SymbolSpec"
Assign NumData by retrieve from TableSpec at Field Name "Specification Number"
Assign InputNo by retrieve from TableSpec at Field Name "Input Number"
Assign OutputNo by retrieve from TableSpec at Field Name "Output Number"
Assign NameData by retrieve from TableSpec

Deconstruction

Inputs : None
Outputs : None
Local : None
Purpose : To destroy the created object class
Algorithm : Delete Table, Delete TableSpec

SetBound

Inputs : X1, Y1, X2, Y2 (type integer)
Outputs : None
Local : None
Purpose : To set the boundary of equipment
Algorithm : If X1 < X2 then
Assign Bound.Left = X1 and Bound.Right = X2
Else Assign Bound.Left = X2 and Bound.Right = X1
End If
If Y1 < Y2 then
Assign Bound.Top = Y1 and Bound.Bottom = Y2
Else Assign Bound.Top = Y2 and Bound.Bottom = Y1
End If
Assign Bound.Width = Bound.Right - Bound.Lef
Assign Bound.Height = Bound.Bottom – Bound.Top
Assign Bound.MidX = Bound.Left + (Bound.Right – Bound.Left)/2
Assign Bound.MidY = Bound.Top + (Bound.Bottom – Bound.Top)/2
Call SetBound in Label with Bound.Right +2, Bound.Bottom-14, Scale

CheckInputAndOutput

Inputs : None
Outputs : None
Local : None
Purpose : To show that amount of input and output is valid or not.
Algorithm : If input != InputNo and InputNo != 0 Or output != OutputNo and

OutputNo != 0 then
Assign color of Label = "Red"
Else
Assign color of Label = the same color as line
End If

DrawState

Inputs : None
Outputs : None
Local : None
Purpose : To show that the equipment is activate.
Algorithm : Set color = "Teal"
Draw the rectangle at control point.

Resize

Inputs : Zoom (type integer)
Outputs : None
Local : oldScale (type integer)
Purpose : To resize the equipment using percentage for calculates.
Algorithm : Assign oldScale = Scale
Assign Scale = Zoom
Assign Bound.Left = (Bound.Left * Scale)/oldScale
Assign Bound.Right = (Bound.Right * Scale)/oldScale;
Assign Bound.Top = (Bound.Top * Scale)/oldScale;
Assign Bound.Bottom = (Bound.Bottom * Scale)/oldScale;
Call SetBound with Bound.Left, Bound.Top, Bound.Right, Bound.Bottom
Call Drawing

Drawing

Inputs : None
Outputs : None
Local : None
Purpose : To draw the selected equipment type.
Algorithm : If Color is not White then
Assign the current pen color = SColor
Assign the current font color = SColor
End If
Drawing equipment image.
Call CheckInputAndOutput

Move

Inputs : X1, Y1, X2, Y2 (type integer)
Outputs : Type boolean
Local : left, top, right, bottom (type integer)
Purpose : To move a current equipment or a equipment label to new position.
Algorithm : If the label is moved then
Return Move in Label with X1, Y1, X2, Y2
End If
Assign left = X2 - (X1 – Bound.Left)
Assign top = Y2 - (Y1 – Bound.Top)
Assign right = X2 + (Bound.Right - X1)
Assign bottom = Y2 + (Bound.Bottom - Y1)
Call SetBound with left, top, right, bottom
Call Drawing
Return true

Modify

Inputs : X1, Y1, X2, Y2 (type integer)
Outputs : None
Local : None
Purpose : To modify the sizes of equipment at control point.
Algorithm : If X1 = Bound.Left then Assign Bound.Left = X2
End If
If X1 = Bound.Right then Assign Bound.Right = X2
End If
If Y1 = Bound.Top then Assign Bound.Top = Y2
End If
If Y1 = Bound.Bottom then Assign Bound.Bottom Y2
End If
Call SetBound with Bound.Left, Bound.Top, Bound.Right, Bound.Bottom
Call SetBound in Label with Bound.Right +2, Bound.Bottom-14, Scale
Call Drawing

EditTable

Inputs : Show (type integer)
Outputs : None
Local : None
Purpose : To prepare and edit the detailed information.
Algorithm : If Table has a record of this equipment then
If Show=1 then
Call Show in Table for edit detailed information.
End If
Return

End If
Insert a new record to Table.
Insert all data of the equipment into Table
If Show = 1 then
  Call Show in Table for edit detailed information
End If

DeleteRecord

Inputs : None
Outputs : None
Local : None
Purpose : To delete the record of this equipment from Table
Algorithm : If Table has not a record of this equipment then Return
End If
Go to a record of this equipment
Delete Table at this record

Save

Inputs : FileName (type string)
Outputs : None
Local : TmpTable (type Table)
Purpose : To copy the Table to the Table of file
Algorithm : If Table has not a record of this equipment then Return
End If
Insert the detailed information that can not edit by user
Create TmpTable that table name is FileName + "Symbol"
Assign TmpTable = Table

SaveNode

Inputs : FileName (type string)
Outputs : None
Local : None
Purpose : To save the detailed information of equipment in the Table
Algorithm : If Table has not a record of this equipment then Return
End If
Change the table name of Table to FileName + "Symbol"
Call EditTable with 0 (don't show the table for edit)

Open

Inputs : FileName (type string)
Outputs : None
Local : TmpTable (type Table)

Purpose : To copy a table of file to the Table.
Algorithm : Assign TmpTable = table that table name is FileName + "Symbol"
If TmpTable is not exist then Return
End If
Assign Table = TmpTable

GetDataOnTable

Inputs : T (type Table)
Outputs : None
Local : left, right, top, bottom (type integer)
Purpose : To retrieve the detailed information from the table.
Algorithm : Assign all detailed information of equipment from Table. (such as ID in field name "ID", Name in field name "Name", input in field name "Input Number".)
Call SetBound with left, top, right, bottom
Call ChangeText in Label with Name

ChangeColor

Inputs : C (type Color)
Outputs : None
Local : None
Purpose : To change color of a selected equipment.
Algorithm : Assign SColor = C
Call ChangeColor in Label with C
Call Drawing

ChangeFont

Inputs : FName (type string), FSize (type integer), FStyle (type FontStyle)
Outputs : None
Local : None
Purpose : To change font of a equipment label.
Algorithm : Call ChangeFont in Label with Fname, FSize, Fstyle

The operation in the Table class

Construction

Inputs : LineOrSymbol (type boolean), numData and numComp (type integer) and FieldName (type array of string)
Outputs : None
Local : None
Purpose : To initial a table class

Algorithm : Assign LineActive = LineOrSymbol
Assign numData = nData, Assign numComp = nComp
Assign NameData[0..numData] = FieldName[0..numData]

SetTabsheet

Inputs : Start (type integer)
Outputs : None
Local : None
Purpose : To create the tabsheet for each component of line.
Algorithm : Create TabSheet from 0..numComp
Create Panel from 0..numData In each TabSheet
Create DBEdit from 0..numData In each TabSheet
Assign Caption in Panel[0..numData] = NameData[0..numData]

Show

Inputs : None
Outputs : None
Local : None
Purpose : To set the table form for show.
Algorithm : If LineActive = true then Call SetTabSheet with true
Else
Create Panel from 0..numData
Create DBEdit from 0..numData
Assign Caption in Panel[0..numData] = NameData[0..numData]
Assign DataSource in DBEdit = DataSource in activate symbol
End If

CloseTable

Inputs : None
Outputs : None
Local : None
Purpose : To post the data to table and close the form.
Algorithm : Call UpdateData in the active symbol.
delete TabSheet
delete Component
delete Panel
delete DBEdit
delete NameData

The operation in the Label class

Construction

Inputs : Str (type string), color (type color)
Outputs : None
Local : None
Purpose : To initial Label class
Algorithm : Assign Text = str, PenColor = color, FontName = a current font name, FontSize = a current font size, FontStyle = a current font style, disX = 0 and disY = 0

SetBound

Inputs : X, Y, Zoom (type integer)
Outputs : None
Local : OldFontSize (type integer)
Purpose : To set the boundary of label
Algorithm : Assign OldFontSize = FontSize
Assign FontSize = (OldFontSize*Zoom)/100
Assign Scale = Zoom
Assign X = X + disX
Assign Y = Y + disY
Assign a current font name = FontName
Assign a current font size = FontSize
Assign a current font style = FontStyle
Assign Bound.Left = X
Assign Bound.Top = Y
Assign Bound.Right = X + the Text Width
Assign Bound.Bottom = Y + the Text Height

Move

Inputs : X1, Y1, X2, Y2 (type integer)
Outputs : Boolean
Local : None
Purpose : To move a label to a new position
Algorithm : Assign disX = X2 - (X1 - disX)
Assign disY = Y2 - (Y1 - disY)
Assign Bound.Left = X2 - (X1 - Bound.Left)
Assign Bound.Top = Y2 - (Y1 - Bound.Top)
Assign Bound.Right = Bound.Left + a Text Width
Bound.Bottom = Bound.Top + a Text Height
Call Drawing
Return false

Drawing

Inputs : None
Outputs : None
Local : None

Purpose : To show a text of label
Algorithm : If a current pen color is not white then Assign a font color = PenColor
Assign a font name = FontName
Assign a font size = FontSize
Assign a font style = FontStyle
End If
Show Text within boundary

ChangeText

Inputs : newText (type string)
Outputs : None
Local : None
Purpose : To change a text of label
Algorithm : Delete the old text
Assign text = newText
Call Drawing

ChangeColor

Inputs : C (type color)
Outputs : None
Local : None
Purpose : To change a color of label
Algorithm : Delete the old text
Assign PenColor = C
Call Drawing

ChangeFont

Inputs : FName (type string), FSize (type integer), FStyle (type FontStyle)
Outputs : None
Local : C (type color)
Purpose : To change a font format of label
Algorithm : Delete the old text
Assign FontName =FName
Assign FontSize = FSize
Assign FontStyle = FStyle
Call Drawing

Procedural design of the "Main" subsystem

Class in this subsystem:

- Flowsheet

- Group

The operation in the Flowsheet class

Construction

Inputs : None
Outputs : None
Local : None
Purpose : To initialize flowsheet class.
Algorithm : Assign LineNo = 0, SymbolNo = 0, Scale = 100 and InGroup = false

DeConstruction

Inputs : None
Outputs : None
Local : None
Purpose : To Destroy the created object class.
Algorithm : Delete LLine, Delete LSymbol, Delete Lbuff and Delete SBuff

SetCursor

Inputs : X, Y, getNode (type integer)
Outputs : output (type integer)
Local : output (type integer)
Purpose : To set a cursor at position X, Y.
Algorithm : Assign output = 0
If (X, Y) is not in boundary of all node in LLine and LSymbol then
Assign output = 0
End If
If (X, Y) is in boundary of the node in LLine or LSymbol then
If (X, Y) is in the control point of the node in LLine or LSymbol then
Assign output = 2
Else Assign output = 1
End If
End If
If output = 0 then Assign Cursor = crDraw
End If
If output = 1 then Assign Cursor = crMove
End If
If output = 2 then Assign Cursor = crModify
End If
Return output

DeleteSymbol

Inputs : None
Outputs : None
Local : None
Purpose : To delete a current symbols node on LLine or LSymbol.
Algorithm :
```
If Active symbol is grouped then
    Go to first node in Group
    Repeat
        If current node in Group is Line type then
            Go to the node in LLine that ID = ID in Group
            Delete current node of LLine
        Else
            Go to the node in LSymbol that ID = ID in Group
            Delete current node of LSymbol
        End if
    Until Group is end
Else
    If Type of active symbol is Line type then
        Delete current node of LLine
    Else Delete current node of LSymbol
    End If
End If
```

CleanNodeImage

Inputs : None
Outputs : None
Local : None
Purpose : To clean an activated symbol images on drawing area.
Algorithm :
```
Assign color = clWhite
If type of current symbol is line type then Call Drawing in LLine
Else Call Drawing in LSymbol
End If
If current symbol is in group (InGroup = true) then
    Delete all symbol image in group
End If
Draw background on boundary of cleaned symbol
```

Drawing

Inputs : DrawState (type integer) (1 = draw state, 0 = don't draw state)
Outputs : None
Local : None
Purpose : To draw an activated symbol images on drawing area.
Algorithm :
```
If Type of active symbol is line type then
    Call Drawing in Lline
```

```
        If DrawState = 1 then Call DrawState in LLine
        End If
    Else
        Call Drawing in LSymbol
        If DrawState = 1 then Call DrawState in LLine
        End If
    End If
    If InGroup = true then Draw all symbol image in Group
        If DrawState = 1 then Draw state all symbol in Group
        End If
    End If
```

DrawState

Inputs : None
Outputs : None
Local : None
Purpose : To draw a state of symbol for an activated symbol on drawing area.
Algorithm : If a active symbol is line type then Call DrawState in LLine
Else Call DrawState in LSymbol
End If

Move

Inputs : X1, Y1, X2, Y2 (type integer)
Outputs : None
Local : None
Purpose : To change position of the selected symbols on flowsheet and to move the selected symbols on drawing area.
Algorithm :

```
If a active symbol is line type then
    If InGroup = true then Call Move in LLine with X1, Y1, X2, Y2
    Else call MoveSymbol with X1, Y1, X2, Y2, LLine
    End If
Else Call MoveSymbol with X1, Y1, X2, Y2, LSymbol
End If
If InGroup = false then return
End If
Go to the first node in Group
Repeat
    If a current node in Group is line type then
        Go to a line node in LLine that pointed by current node in
        Group
        Call Move in LLine with X1, Y1, X2, Y2
    Else
        Go to a symbol node in LSymbol that pointed by current
        Node in Group
        Call MoveSymbol with X1, Y1, X2, Y2 Lsymbol
```

```
    End If
Until Group is end
```

Modify

Inputs : X1, Y1, X2, Y2 (type integer)
Outputs : None
Local : OldStartX, oldStartY, oldEndX, oldEndY, output, ModifyPoint (type integer)
Purpose : To change position of the selected symbols and to modify size of the selected symbols.
Algorithm :

```
If a active symbol is line type then
    Assign oldStartX = StartX in LLine
    Assign oldStartY = StartY in LLine
    Assign oldEndX = EndX in LLine
    Assign oldEndY = EndY in LLine
    Assign ModifyPoint by call Modify in LLine with X1,Y1,X2,Y2
    If flowsheet has symbol then
        If ModifyPoint = 1 (modify at start position of line) then
            Call DeSnap with 0, oldStartX, oldStartY
        End If
        If ModifyPoint = 2 (modify at end position of line) then
            Call DeSnap with 1, oldEndX, oldEndY
        End If
    End If
Else
    Call Modify in LSymbol with X1, Y1, X2, Y2
    Call Drawing in LSymbol
End If
```

SaveFlowsheet

Inputs : FileName (type string)
Outputs : None
Local : None
Purpose : To save all symbol and its detail information on flowsheet to table of file.
Algorithm :

```
If flowsheet has line symbol then
    Call Save in LLine with FileName
    Call SaveLinklist with FileName and LLine
End If
If flowsheet has LSymbol symbol then
    Call Save in LSymbol with FileName
    Call SaveLinklist with FileName and LSymbol
End If
If flowsheet has group of symbol then Call SaveGroup with FileName
End If
```

OpenFlow$_S$heet

Inputs : FileName (type string)
Outputs : None
Local : Table (type table)
Purpose : To retrieve detailed information of all symbols on flowsheet file.
Algorithm : If flowsheet has line or symbol then
Clear detailed information of all symbols on this flowsheet
End If
Open Table by assign table name = FileName + "Line"
If Table has data then
Call GetDataOnTable with Lline, Table, FileName
End If
Open Table by assign table name = FileName + "Symbol"
If Table has data then
Call GetDataOnTable with LSymbol, Table, FileName
End If
Open Table by assign table name = FileName + "Group"
If Table has data then
Call GetDataOnGroupTable with Table, FileName
End If

ShowReport

Inputs : None
Outputs : None
Local : None
Purpose : To prepare the detailed information of all symbol for show report.
Algorithm : If flowsheet has line symbol then Call Save in LLine
End If
If flowsheet has equipment symbol then Call Save in LSymbol
End If

CopyToBuff

Inputs : None
Outputs : None
Local : None
Purpose : To Copy the selected symbol on flowsheet to Buffer
Algorithm : If Buffer has data (LBuff or SBuff != NULL) then Delete Buffer
End If
If InGroup = false then
If a active symbol is line type then
Copy current node in LLine from Lbuff
Else Copy current node in Lsymbol from SBuff
End If
Return

```
        End If
        Go to a first node in Group
        Repeat
            If current node in Group is line type then
                Go to a node in LLine that pointed by current node in Group
                Copy current node in LLine from Lbuff
            Else
                Go to a node in LSymbol that pointed by current node in Group
                Copy current node in LSymbol from SBuff
            End If
        Until Group is end
```

PasteToFlowsheet

```
Inputs :     None
Outputs :    None
Local :      X1, Y1, X2, Y2 (type integer)
Purpose :    To paste all symbol in Buffer to flowsheet
Algorithm :  If Buffer has not symbol then return
             End If
             If Buffer (LBuff and SBuff) has symbol more than one symbol then
                Assign InGroup = true
             Else Assign InGroup = false
             End If
             Assign X2 = current position X of cursor in flowsheet
             Assign Y2 = current position Y of cursor in flowsheet
             If LBuff has data then
                Assign X1 = left of current node in Lbuff
                Assign Y1 = top of current node in Lbuff
             Else
                If SBuff has data then
                   Assign X1 = left of current node in SBuff
                   Assign Y1 = left of current node in SBuff
                End If
             End If
             If LBuff has data then
                Repeat Copy current node in LBuff from LLine
                Until LBuff is end
             End If
             If SBuff has data then
                Repeat Copy current node in SBuff from LSymbol
                Until SBuff is end
             End If
             Call Move with X1, Y1, X2, Y2
             Call Drawing with 1
```

ResizeFlowsheet

Inputs : Zoom (type integer)
Outputs : None
Local : None
Purpose : To resize all symbol on flowsheet.
Algorithm : Assign Scale = Zoom
Set boundary all symbol on flowsheet
Draw all symbol from new size
Call DrawState

EditTable

Inputs : None
Outputs : None
Local : None
Purpose : To point to a active symbol for edit it's table
Algorithm : If a active symbol is line type then Call EditTable in LLine
Else Call EditTable in LSymbol
End If

CloseTable

Inputs : None
Outputs : None
Local : None
Purpose : To delete data in table of all symbol
Algorithm : If flowsheet has line symbol then Delete Table of LLine
End If
If flowsheet has equipment symbol then Delete table of symbol
End If

UpdateData

Inputs : None
Outputs : None
Local : None
Purpose : To update data from that input detailed information by user
Algorithm : If a active symbol is line type then Call UpdateData in Line
Else Call UpdateData in Lsymbol
End If

GroupSymbol

Inputs : X1, Y1, X2, Y2 (type integer) (called boundary)
Outputs : Boolean

Local : None
Purpose : To group the selected symbol
Algorithm : If LGroup != NULL then GroupNo++
Else GroupNo = 1
End If
If flowsheet has line symbol then
Repeat Go to the node in LLine that boundary within boundary
Call InsertNode in Group with ID of node, Type of node, Group
Until LLine is end
End If
If flowsheet has equipment symbol then
Repeat
Go to the node in LSymbol that boundary within boundary
Call InsertNode in Group with ID of node, Type of node, Group
Until LSymbol is end
End If
If Group = NULL then return false (all symbols aren't in boundary)
End If
Assign InGroup = true
return true

SetGroup

Inputs : None
Outputs : None
Local : None
Purpose : To insert a group buffer to the group of flowsheet
Algorithm : Call InsertTo in Group with Lgroup

UnGroup

Inputs : None
Outputs : None
Local : id, type, groupID (type integer)
Purpose : To separate group of symbol
Algorithm : Assign id = ID of a active symbol
Assign type = Type of a active symbol
Assign groupID by call CheckInGroup in Group with id, type
Call DeleteNode in Group with groupID, LGroup

CleanGroup

Inputs : None
Outputs : None
Local : None
Purpose : To delete all data in Group
Algorithm : If Group has not data then return

```
        end if
        Delete Group
        Assign InGroup = false
```

ChangeColor

```
Inputs :      None
Outputs :     None
Local :       None
Purpose :     To change color of a selected symbol
Algorithm :   If InGroup = true then
                 Go to first node in Group
                 Repeat
                    If current node in Group is line type then
                       Go to a node in LLine that pointed by current node in Group
                       Call ChangeColor in LLine  with Color
                    Else
                       Go to a node in LSymbol that pointed by current node in
                       Group
                       Call ChangeColor in LSymbol  with Color
                    End if
                 Until Group is end
              Else
                 If a active symbol is line type then
                    Call ChangeColor in LLine  with Color
                 Else Call ChangeColor in Lsymbol  with Color
                 End If
              End If
```

ChangeFont

```
Inputs :      FName (type String) FSize (type integer), FStyle (type FontStyle)
Outputs :     None
Local :       None
Purpose :     To Change a font name of the selected symbol
Algorithm :   If InGroup = true then
                 Go to first node in Group
                 Repeat
                    If current node in Group is line type then
                       Go to a node in LLine that pointed by current node in Group
                       Call ChangeFont in LLine  with Color
                    Else
                       Go to a node in LSymbol that pointed by current node in
                       Group
                       Call ChangeFont in Lsymbol  with Color
                    End if
                 Until Group is end
```

```
        Else
            If a active symbol is line type then
                Call ChangeFont in LLine  with Color
            Else Call ChangeFont in LSymbol  with Color
            End If
        End If
```

The operation of the Group class

Construction

Inputs : Gid, id, type (type integer)
Outputs : None
Local : None
Purpose : To initial a construction class.
Algorithm : Assign ID = id, Type = type, GroupID = Gid, Previous = NULL and Next = NULL
Assign Table = table that table name "Group"

CheckInGroup

Inputs : Id, type (type integer)
Outputs : Integer
Local : None
Purpose : To get GroupID that has an activate symbol.
Algorithm : If ID = id and Type = type then return GroupID
End If
While Previous = NULL do
If ID in Previous = id and Type in Previous = type then
Return GroupID in Previous
End If
End While
While Next = NULL do
If ID in Next = id and Type in Next = type then
Return GroupID in Next
End If
End While
Return 0

InsertTo

Inputs : DesG (type Group class)
Outputs : None
Local : Tmp (type Group class)
Purpose : To insert a selected symbol in the link list of group.

Algorithm : If DesG = NULL then create DesG with ID, Type and GroupID End If
Go to the last node in the link list of DesG.
Go to the first node in the link list of temp group.
Loop Create tmp with ID, Type, GroupID
Assign Previous in tmp = DesG
Assign Next in DesG = tmp
Assign Next in tmp = NULL
Until temp group is end

DeleteNode

Inputs : GroupID (type integer), group (type Group class)
Outputs : None
Local : None
Purpose : To delete the node in link list that has GroupID = groupID
Algorithm : If group has one node then delete group
End If
Assign tmp = node in link list that has GroupID = groupID
Delete tmp

Procedural design of "User Interactive" subsystem

Class in this subsystem

- Menu
- Drawing area
- Symbol palette
- Line palette
- Equipment palette

The operation of the Menu class

Construction

Inputs : None
Outputs : None
Local : None
Purpose : To initial the menu class
Algorithm : Assign FileName = ""
Assign ChangeFont = true

Create flowsheet

ConfirmSaveFile

Inputs : None
Outputs : Type boolean
Local : None
Purpose : To ask the user for save the created flowsheet when the flowsheet is closed.
Algorithm : If flowsheet is modified then
show dialog box for confirm
If user chooses cancel the close flowsheet action then return false
End if
If user chooses the save action then call SaveClick
End If
End If
Call CloseTable in the flowsheet
If user close the menu then call Terminate in the system
Else call FormDestroy in the drawing area
End If
Call SetMenu with false
return true

SetMenu

Inputs : Start (type boolean)
Outputs : None
Local : None
Purpose : To set the menu item button.
Algorithm : If Start = true then the menu item is enabled
Else the menu item is not enabled
End If

NewClick

Inputs : None
Outputs : None
Local : None
Purpose : To create the new flowsheet.
Algorithm : If the system has the active flowsheet then call ConfirmSaveFile
end If
Show the drawing area

OpenClick

Inputs : None

Outputs : None
Local : None
Purpose : To open an exist flowsheet from file.
Algorithm : If the system has the active flowsheet then call ConfirmSaveFile
End If
Open the open dialog box for selects the flowsheet file and assign FileName.
Show the drawing area.
Call OpenFlowsheet in flowsheet with FileName

SaveClick

Inputs : None
Outputs : None
Local : None
Purpose : To save the activate flowsheet in its file.
Algorithm : If FileName = "" then call SaveAsClick
Else call SaveFlowsheet in flowsheet
End If

SaveAsClick

Inputs : None
Outputs : None
Local : None
Purpose : To save the activate flowsheet in the new file.
Algorithm : Show the save dialog box for selects the flowsheet file.
Assign FileName = FileName in dialog box
Call SaveFlowsheet in flowsheet with FileName

CloseClick

Inputs : None
Outputs : None
Local : None
Purpose : To close the activate flowsheet.
Algorithm : If the system has an activate flowsheet then call ConfirmSaveFile
end if

PrintClick

Inputs : None
Outputs : None
Local : None
Purpose : To print the activate flowsheet.
Algorithm : Show the print dialog box

Send the flowsheet image for print to the printer.

ExitClick

Inputs : None
Outputs : None
Local : None
Purpose : To terminate the application.
Algorithm : If the system has an activate flowsheet then call ConfirmSaveFile end if
Terminate the system.

CutClick

Inputs : None
Outputs : None
Local : None
Purpose : To cut the selected symbols to the buffer.
Algorithm : Call CopyToBuff in flowsheet
Call DeleteClick

CopyClick

Inputs : None
Outputs : None
Local : None
Purpose : To copy the selected symbols to the buffer.
Algorithm : Call CopyToBuff in flowsheet

PasteClick

Inputs : None
Outputs : None
Local : None
Purpose : To paste the symbol in the buffer in the activate flowsheet.
Algorithm : Call PasteToFlowsheet in flowsheet

DeleteClick

Inputs : None
Outputs : None
Local : None
Purpose : To delete the selected symbols.
Algorithm : Call CleanNodeImage in flowsheet
Call DeleteSymbol in flowsheet

GroupClick

Inputs : None
Outputs : None
Local : None
Purpose : To group the selected symbol in the flowsheet.
Algorithm : Call SetGroup in flowsheet

UngroupClick

Inputs : None
Outputs : None
Local : None
Purpose : To separate the grouped symbol.
Algorithm : Call UnGroup in flowsheet.

FontClick

Inputs : None
Outputs : None
Local : None
Purpose : To set the font of the system.
Algorithm : Show the font dialog box
Assign Font_Name = Font name in dialog
Assign Font_Size = Font size in dialog
Assign Font_Style = Font style in dialog
If Font_Style has Bold then down the Bold button
End If
If Font_Style has Italic then down the Italic button
End If
If Font_Style has Underline then down the Underline button End If

BoldClick

Inputs : None
Outputs : None
Local : None
Purpose : To set the font style to Bold style.
Algorithm : If Bold button is down then Font_Style + Bold
Else Font_Style - Bold
End If
Call ChangeFont in flowsheet with Font_Name, Font_Size, Font_Style

ItalicClick

Inputs : None

Outputs : None
Local : None
Purpose : To set the font style to Italic style.
Algorithm : If Italic button is down then Font_Style + Italic
Else Font_Style – Italic
End If
Call ChangeFont in flowsheet with Font_Name, Font_Size, Font_Style

UnderlineClick

Inputs : None
Outputs : None
Local : None
Purpose : To set the font style to underline style.
Algorithm : If Underline button is down then Font_Style + Underline
Else Font_Style - Underline
End If
Call ChangeFont in flowsheet with Font_Name, Font_Size, Font_Style

Change ColorClick

Inputs : None
Outputs : None
Local : None
Purpose : To change the current system color.
Algorithm : Show the color dialog box for selects the color
Call ChangeColor in flowsheet

Change SpecificationClick

Inputs : None
Outputs : None
Local : None
Purpose : To access the table SymbolSpec for edit the specification of all symbols.
Algorithm : Show the symbol specification table for edit.

SymbolClick

Inputs : None
Outputs : None
Local : None
Purpose : To show the symbol palette.
Algorithm : Show the symbol palette.

TableClick

Inputs : None
Outputs : None
Local : None
Purpose : To show the table of the activate symbol.
Algorithm : Call EditTable in flowsheet.

Zoom inClick

Inputs : None
Outputs : None
Local : None
Purpose : To increase the size of drawing area.
Algorithm : Call ZoomInOrOut in the drawing area with true

Zoom outClick

Inputs : None
Outputs : None
Local : None
Purpose : To decrease the size of drawing area.
Algorithm : Call ZoomInOrOut in the drawing area with false

ReportClick

Inputs : None
Outputs : None
Local : None
Purpose : To report all symbol with the detailed information on the flowsheet.
Algorithm : Call ShowReport in flowsheet.

Help TopicsClick

Inputs : None
Outputs : None
Local : None
Purpose : To show the help of the software.
Algorithm : Show help

AboutClick

Inputs : None
Outputs : None
Local : None
Purpose : To show the about of the software.

Algorithm : Show the about dialog

The operation of the Drawing area class

Show

Inputs : None
Outputs : None
Local : None
Purpose : To initial the drawing area for draw.
Algorithm : Show the text of modification at bottom = ""
Show the text of zoom at bottom = 100%

FormDestroy

Inputs : None
Outputs : None
Local : None
Purpose : To close the drawing are form.
Algorithm : Assign Visible = false

MouseDown

Inputs : Button (type TMouseButton), Shift (typeShiftState), X and Y (type integer)
Outputs : None
Local : None
Purpose : To start the image drawing.
Algorithm : If the right click then call Popup in RightClickMenu and return End If
Assign InitialX = X
Assign InitialY = Y
Assign State by call SetCursor in flowsheet with X, Y, 1
If State = 0 then
assign Type = Type in SymbolPalette
Change text of modification at bottom = "Modified"
Call Drawing in flowsheet with InitialX, InitialY, X, Y
Else Call CleanNodeImage in flowsheet
End If

MouseMove

Inputs : Shift (type ShiftState), X and Y (type integer)
Outputs : None
Local : None
Purpose : To change the size of image or to group the image or to set the cursor.

Algorithm : Call ShowPosition with X, Y
If the mouse don't click then call SetCursors in flowsheet in X, Y, 0
Return
EndIf
If State = 0 then
If the mouse don't click on the symbol image then
Draw Rectangle and return
Else call Drawing in flowsheet with InitialX, InitialY, X, Y
End If
Else
If State=1 then call Move in flowsheet with InitialX, InitialY, X, Y
End If
Else
If State = 2 then call Modify in flowsheet with InitialX, InitialY,X,Y
End If
End If
Show the text of modification at bottom = "Modified"

MouseUp

Inputs : Button (type TMouseButton), Shift (typeShiftState), X and Y (type integer)
Outputs : None
Local : None
Purpose : To end the action on the drawing area.
Algorithm : If Type = 0 and State = 0 then call GroupSymbol in flowsheet with InitialX, InitialY, X, Y
Else Call Drawing in flowsheet with 1
End If

ShowPosition

Inputs : X and Y (type integer)
Outputs : None
Local : None
Purpose : To show the text of position at bottom of the drawing area.
Algorithm : Show the text of X position at bottom is X
Show the text of Y position at bottom is Y

ZoomInOrOut

Inputs : ZoomIn (type boolean)
Outputs : None
Local : None
Purpose : To zoom the drawing area and the symbol image on the drawing area.
Algorithm : If ZoomIn then ZoomInd >= 200 then return End If
If !ZoomIn then ZoomInd <= 30 then return End If

If ZoomIn then ZoomInd = ZoomInd + 10
Else ZoomInd = ZoomInd – 10
End If
Show the text of zoom at bottom = ZoomInd
Call ResizeFlowsheet in flowsheet with ZoomInd

The operation of the Symbol palette class

Show

Inputs : None
Outputs : None
Local : None
Purpose : To initial the symbol palette class.
Algorithm : Type = 0
Create Table by assign table name = "SymbolSpec"

The operation of the Line palette class

Click

Inputs : None
Outputs : None
Local : None
Purpose : To set the type of the palette
Algorithm : If Point1 is clicked then Type = 0 End If
If Line is clicked then Type = 1 End If
If Instrument is clicked then Type = 2 End If
If Orifice is clicked then Type = 3 End If
If GateValve is clicked then Type = 4 End If
If GlobeValve is clicked then Type = 5 End If
If CheckValve is clicked then Type = 6 End If
If ReliefValve is clicked then Type = 7 End If
If ControlValve is clicked then Type = 8 End If

The operation of the Equipment palette class

Click

Inputs : None
Outputs : None
Local : None
Purpose : To set the type of the palette

Algorithm : If Point2 is clicked then Type = 0 End If
If HeatEx is clicked then Type = 9 End If
If WaterCooler is clicked then Type = 10 End If
If SteamHeater is clicked then Type = 11 End If
If Reboiler is clicked then Type = 12 End If
If Jcondenser is clicked then Type = 13 End If
If Condenser is clicked then Type = 14 End If
If Absorber is clicked then Type = 15 End If
If Stripper is clicked then Type = 16 End If
If MixerTank is clicked then Type = 17 End If
If LiquidTank is clicked then Type = 18 End If
If PressureTank is clicked then Type = 19 End If
If Jkettle is clicked then Type = 20 End If
If Reaction is clicked then Type = 21 End If
If Ppump is clicked then Type = 22 End If
If Cpump is clicked then Type = 23 End If
If Compressor is clicked then Type = 24 End If
If Furnace is clicked then Type = 25 End If
If Distillation is clicked then Type = 26 End If
If Bfractionating is clicked then Type = 27 End If
If Sfractionating is clicked then Type = 28 End If
If Separator is clicked then Type = 29 End If
If Vvessel is clicked then Type = 30 End If
If Hvessel is clicked then Type = 31 End If
If Other symbol is clicked then Type = 32
Call OtherSymbolClick
End If

OtherSymbolClick

Inputs : None
Outputs : None
Local : None
Purpose : To set the image and specification of the new symbol.
Algorithm : Show the open dialog box for selects the image file
Show the table of specification for inserts the new symbol to the SymbolSpec table.

APPENDIX D

USER MANUAL

The result of study for this research is the ChemProc software. Which designed and developed by Object-Oriented technology.

This chapter described step of using the ChemProc program, screens of the application and the change of related data. This software divided into seven main functions below:

- Initialization application
- Creation the flowsheet
- Edit the symbol specification type
- Identification the new symbol
- Edit and update the symbol technical specification
- manipulation the flowsheet file
- The program utilities

Initialization application

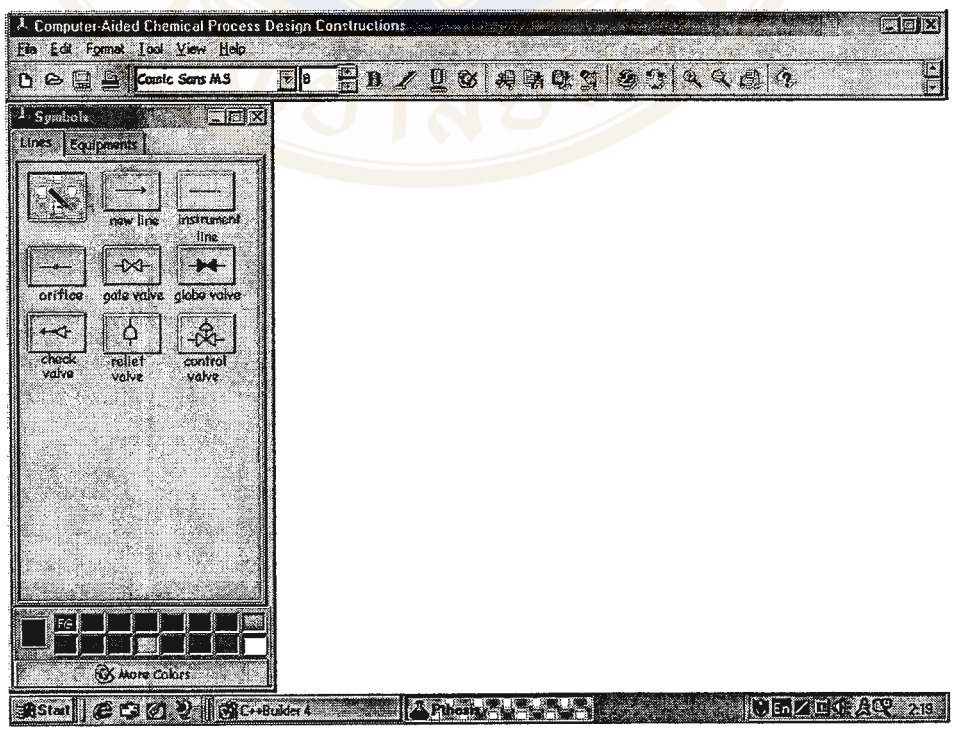


Figure D.1 The starting program

When the user starts the program, the system will show screen in Figure D.1, which illustrated that the system is ready for next action such as creation a new flowsheet, open the exists flowsheet or change the symbol specification. The Initial screen consists of the menu bar (shown on the top of screen) and the symbol palette (shown on the left of screen).

Creation the flowsheet

At the beginning, if the user wants to open the existed flowsheet, user will chooses the open menu on the menu bar. If the user wants to create the new flowsheet, user will chooses the new menu on the menu bar. After that, the dialog box that shown in Figure D.2 will be shown for identification the size of the flowsheet.

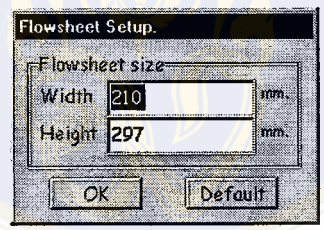


Figure D.2 The form for setting the flowsheet size

The creation a flowsheet can be divided into 5 steps:

1. Drawing the symbol
2. Moving the selected symbol
3. Changing the size of selected symbol
4. Connection the equipment symbol with the line symbol
5. Saving the flowsheet

Drawing the symbol

Now, the user can draw the flowsheet by selects any symbol on the symbol palette.

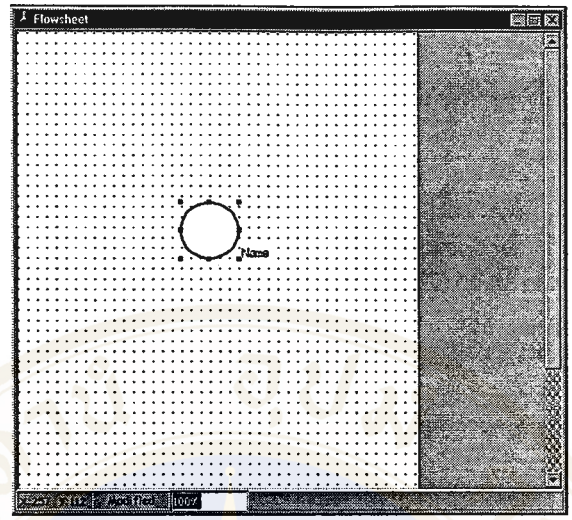


Figure D.3 The example of drawing the heat exchange symbol

The next action is to place the selected symbol by clicking the mouse ( ) at wanted position on the drawing area. The Figure D.3 illustrated the heat exchange symbol drawn on the drawing area.

Before any symbol on the flowsheet will be operated, the user must select the wanted symbol. Accordingly, The user must click at the symbol, when one symbol wanted. If many symbols are wanted, the user must create the boundary that the wanted symbol is in the boundary. By clicking and moving the mouse ( ) as rectangle.

Moving the selected symbol

The user can move the selected symbol by clicking the mouse ( ) at the wanted symbol on the drawing area and moving it to the wanted position. And then relinquishing the mouse.

Changing the size of selected symbol

If the symbol changed the size is the equipment, the user must click the mouse at the control point ( ) on that equipment or when the cursor of mouse is .

If the symbol changed the size is the line, the user can click the mouse at the start point or the end point for increase or decrease the length that the cursor of mouse same as the changing size of the equipment symbol. In addition to, the user can click the mouse at the middle point that the cursor of mouse is ↔ or ↕ for modification the format of the line symbol.

Connection the equipment symbol with the line symbol

The one characteristic of this application is connect the equipment symbol with the line symbol for easy to modify the flowsheet. By moving or changing the start or end point of the line symbol until the control point of line and the equipment is the same position. This stage is illustrated on the control point of the line symbol that has the pink color shown in the Figure D.4. In addition to, this part checks the input and output information of each symbol that illustrated via the color of invalid symbol is red but the color of valid symbol same as the color of symbol image.

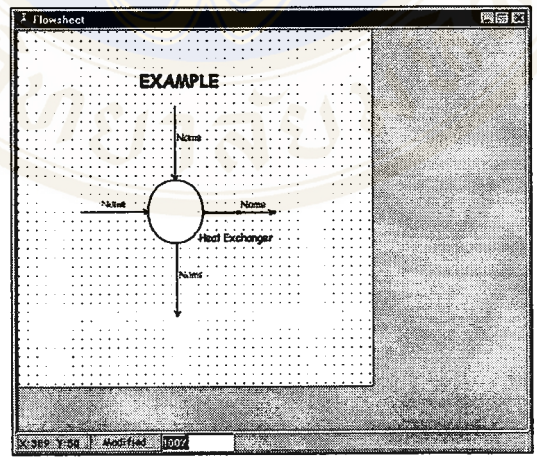


Figure D.4 The example of connecting the line with the equipment symbol

Saving the flowsheet

The user can save the flowsheet by choose the save menu, if wants to save in a new location or choose the save as menu, if to save in an old location. The

flowsheet saved consists of a main file (*.FSH), an image file (*.BMP) and the database that have the information of all symbols in the flowsheet.

Edit the symbol specification type

The symbol in this system has the technical specification that the user must identify the type of each symbol specification. By choosing the Tool|Change Specification on the menu bar that the form in the Figure D.5 is shown. For this form, the user can edit the technical specification of each symbol. By the way, the user must identify the specification number following the specification type that inputted. However, this application can support 10 technical specification type of all symbols.

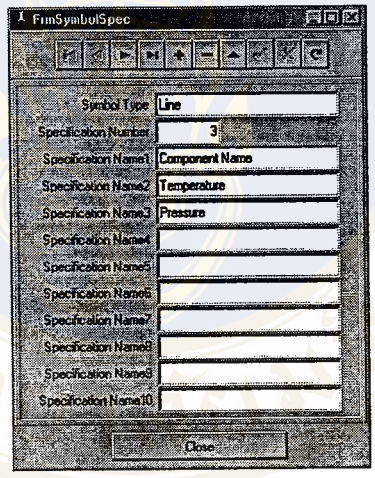


Figure D.5 The symbol specification form

Identification the new symbol

The user can identify the new equipment symbol. By choosing the other symbol button on the equipment palette at the symbol palette and selecting the image file of new symbol on the open dialog box. After that, the specification table of the new symbol is shown for identifying the essential detailed information. The table is shown in Figure D.6 that the Specification No. can define less than 10.

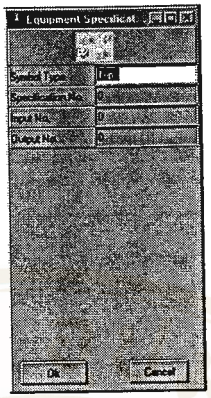


Figure D.6 The specification table

Edit and update the symbol technical specification

This part is manipulated the technical specification of each symbol on the flowsheet. Which the user must double click on the wanted symbol on the drawing area. After that the system will illustrated the table form for editing or updating the technical specification.

Figure D.7 The table of line specification and its component specification

The table form is divided into the line that illustrated in Figure D.7 and the equipment table that illustrated in the Figure D.8.

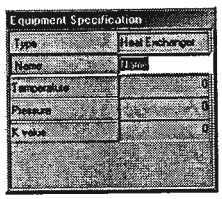


Figure D.8 The table of equipment specification

Manipulation the flowsheet file

In this part is divided into the saving file and the opening file. The saved flowsheet file consists of the main 3 files that are:

- The main files that the type is FSH (FlowSHeet).
- The image files that saving the image of the flowsheet. This file is type BMP (Bitmap).
- The database of the flowsheet that consists of the line table, the line component table and the equipment table. These tables have Paradox type that the alias name is DefaultDD.

In addition to, if the flowsheet has the user-defined symbol, the image file of this user-defined symbol is necessary for drawing. Because, the symbol image are used via the image file in manipulation every time.

The user can save the flowsheet file using the save menu for saves in the old file or the save as menu for save in the other file. Which the saved main file and the image file is in the selected directory. But the saved database is in the directory the defined by alias table. Accordingly, the table name of the saved database consists of the (filename)line, (filename)component and (filename)symbol.

The program utilities

This software has some functions for accommodation in drawing the flowsheet. The utilities are:

- Changing the color
- Changing the font
- Zoom in and out
- Cut, copy, past and delete the selected symbol
- Output of the working flowsheet

Changing the color

The color of each symbol on the drawing area can be changed by chooses the Tool|Change Color on the menu bar or change color on the symbol palette. However, the user-defined symbol can not use this function.

Changing the font

This function manipulates the font of the symbol name. That is, the user can change the font name by select the font exists in her computer, the font size or the font type that divided into the bold, italic and underline type.

Zoom in and out

In this function, the user can increase or decrease the flowsheet image and everything in the flowsheet by choosing the zoom in or zoom out on the menu bar. The percent of zoom is illustrated on the bottom of the drawing area.

Cut, copy, paste and delete the selected symbol

Before manipulation the symbol, the user must select the needed symbol that one or more. The user can manipulate the symbol via the menu on the menu bar or the right click at the drawing area. The new symbol that created by copying or cutting other symbol has the technical specification same as the copied or cut symbol.

Output of the working flowsheet

After creation the flowsheet, the user can report the flowsheet illustrated in the Appendix B. The report can illustrate the technical specification of all line symbols or all equipment symbols on the drawing area. Which symbol technical specification report can also be printed on the paper. Furthermore, the flowsheet image that is output of this application can be used in the other task.

BIOGRAPHY

NAME	Miss. Roschong Sakdumrong
DATE OF BIRTH	1 March 1976
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	Chulalongkorn University, 1993-1997: Bachelor of Science (Computer Science) Mahidol University, 1997-2000: Master of Science (Technology of information system management)