

California State University, San Bernardino

CSUSB ScholarWorks

Theses Digitization Project

John M. Pfau Library

2004

Conceptual model builder

Chia-Yang Lin

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Software Engineering Commons](#)

Recommended Citation

Lin, Chia-Yang, "Conceptual model builder" (2004). *Theses Digitization Project*. 2708.
<https://scholarworks.lib.csusb.edu/etd-project/2708>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

CONCEPTUAL MODEL BUILDER

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science


by
Chia-Yang Lin
March 2004

CONCEPTUAL MODEL BUILDER


A Project
Presented to the
Faculty of
California State University,
San Bernardino


by
Chia-Yang Lin
March 2004

Approved by:


Dr. Josephine G. Mendoza

3-8-04
Date


Dr. George Georgiou


Dr. David Turner

© 2004 Chia-Yang Lin

ABSTRACT

Whenever one designs a new database system, an Entity-Relationship Diagram (ER diagram) is always needed to present the structure of this database. Using the graphically well-arranged ER Diagram helps you to easily understand the entities, attributes, domains, primary keys, foreign keys, constraints, and relationships inside a database.

Conceptual Model Builder (CMB) is a project aimed at offering a Java-based conceptual diagram tool in several diagrammatic notations. It will support a user-friendly graphic interface. Since object-modeling methodologies such as Universal Modeling Language and Object Modeling techniques are popular, CMB will provide UML-style diagram. A user can switch among these different diagrams, or customize the conceptual diagram notations. The current CMB version allows a conceptual model with about twenty entities and their relationship diagrams.

This data-modeling tool is an ideal choice for companies and developers.

ACKNOWLEDGMENTS

One of the great pleasures of writing this document is acknowledging the efforts of these people who helped me during the project development.

Dr. Josephine Mendoza, my advisor, her precious assistance and advice on both program design and documentation are the key to the success of this project.

Dr. David Turner and Dr. George Georgiou: my committee members. They provide the valuable assistance.

Mr. Ken Han, his advice and help, especially on operating system installation facilitated the smooth development of the project.

Joseph and John provided their aid on system testing and documentation review.

My parents for providing full financial aid, without them this project cannot be finished on time.

The support of the National Science Foundation under award 9810708 is also gratefully acknowledged.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENTS	iv
LIST OF FIGURES	ix
CHAPTER ONE: SOFTWARE REQUIREMENTS SPECIFICATION	
1.1 Introduction	1
1.1.1 What is a Graphic Conceptual Model?	1
1.1.2 Why we Need a New Tool?	2
1.2 Purpose of the Project	5
1.3 Significance of the Project	6
1.4 System Requirements	6
1.5 Limitations	7
1.6 Definition of Terms	8
1.7 Organization of the Project	9
CHAPTER TWO: SOFTWARE DESIGN	
2.1 Introduction	10
2.2 User Interface Component	11
2.3 Diagram Engine Component	13
2.4 Notation Data Component	14
2.5 Script and Output Component	15
2.6 The Network Component	15
2.7 Model-View-Controller	16
2.8 The Detail Design	18
2.8.1 CmbMenu Class	18

2.8.2	TableObject Class	19
2.8.3	DataIO Class	22
2.8.4	Open Class	22
2.8.5	Save Class	23
2.8.6	InsertTable Class	26
2.8.7	InsertRelation Class	26
2.9	Summary	26
CHAPTER THREE: SOFTWARE QUALITY ASSURANCE		
3.1	Introduction	28
3.2	Unit Test Plan	28
3.3	Integration Test Plan	30
3.4	System Test Plan	32
3.5	Summary	32
CHAPTER FOUR: MAINTENANCE		
4.1	Introduction	34
4.2	Model Data Maintenance	34
4.3	The Graphic Engine Maintenance	35
4.4	User Interface Maintenance	36
4.4.1	Main Menu	36
4.4.2	Save/Load	37
4.4.3	South Menu	37
4.4.4	Tree	37
4.4.5	The Model View	37
4.5	Network Interface	38
4.6	Summary	41

CHAPTER FIVE: USERS MANUAL

5.1 The Main Menu	42
5.2 The Model View	43
5.3 The South Submenu	44
5.4 The File Submenu	46
5.4.1 Login	46
5.4.2 Save	46
5.4.3 Open	47
5.4.4 Print	48
5.4.5 Page Setup	48
5.5 The Table Submenu	49
5.5.1 Entities	49
5.5.2 Attributes	50
5.6 The Tool Submenu	51
5.6.1 Setup Font	51
5.6.2 Options	52
5.7 The Connection Submenu.....	52
5.7.1 Server	53
5.7.2 Client	53

CHAPTER SIX: CONCLUSIONS

6.1 Introduction	55
6.2 Future Design	56
6.2.1 Extend Model Views	56
6.2.2 Improve Menu Functions	56

6.2.3 Improve the Print Preview Features	56
6.2.4 Add More Managing Tools for Network Component	57
6.2.5 Make Graphic User Interface More Friendly	57
6.3 Summary	57
APPENDIX: PROJECT SOURCE CODES	59
REFERENCES	61

LIST OF FIGURES

Figure 1.	Phases of Database Design	1
Figure 2.	A Example of ER-Diagram	2
Figure 3.	Alternative Notations, Entity Types, Attributes, and Relationships	3
Figure 4.	Cardinality Ratios and (Min, Max) Notations. Several Variations of Displaying Specializations/ Generalizations	4
Figure 5.	Alternative Specializations/ Generalizations Notations	5
Figure 6.	User Interface Component	11
Figure 7.	Model-View-Controller Architecture	17
Figure 8.	Mapping CMB to MVC	18
Figure 9.	CmbMenu Class	19
Figure 10.	TableObject Class	20
Figure 11.	TableObject Class	21
Figure 12.	DataIO Class	22
Figure 13.	Open Class	23
Figure 14.	Save Class	23
Figure 15.	InsertTable Class	24
Figure 16.	InsertRelation Class	25
Figure 17.	A Sample of Unit Test	29
Figure 18.	The Unit Test List	30
Figure 19.	The Integration Test List	31
Figure 20.	The System Test List	32
Figure 21.	The Major Model Data Structure	34

Figure 22. The Graphic Engine	36
Figure 23. Main Menu	42
Figure 24. One of Model View	43
Figure 25. Another Model View	44
Figure 26. The Metal Look And Feel (Part 1)	44
Figure 27. The Metal Look And Feel (Part 2)	45
Figure 28. The Metal Look And Feel (Part 3)	45
Figure 29. The South Menu Bar	46
Figure 30. Login Interface	46
Figure 31. Save Interface	47
Figure 32. Load Interface	47
Figure 33. Print Interface	48
Figure 34. Page Setup Interface	49
Figure 35. Entities Interface	50
Figure 36. Attributes Interface	51
Figure 37. Setup Font Interface	52
Figure 38. Option Interface	52
Figure 39. Server Interface	53
Figure 40. Client Interface	54

CHAPTER ONE
SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction

1.1.1 What is a Graphic Conceptual Model?

As illustrated in Figure 1, conceptual modeling is an important phase in designing a successful database. [1]

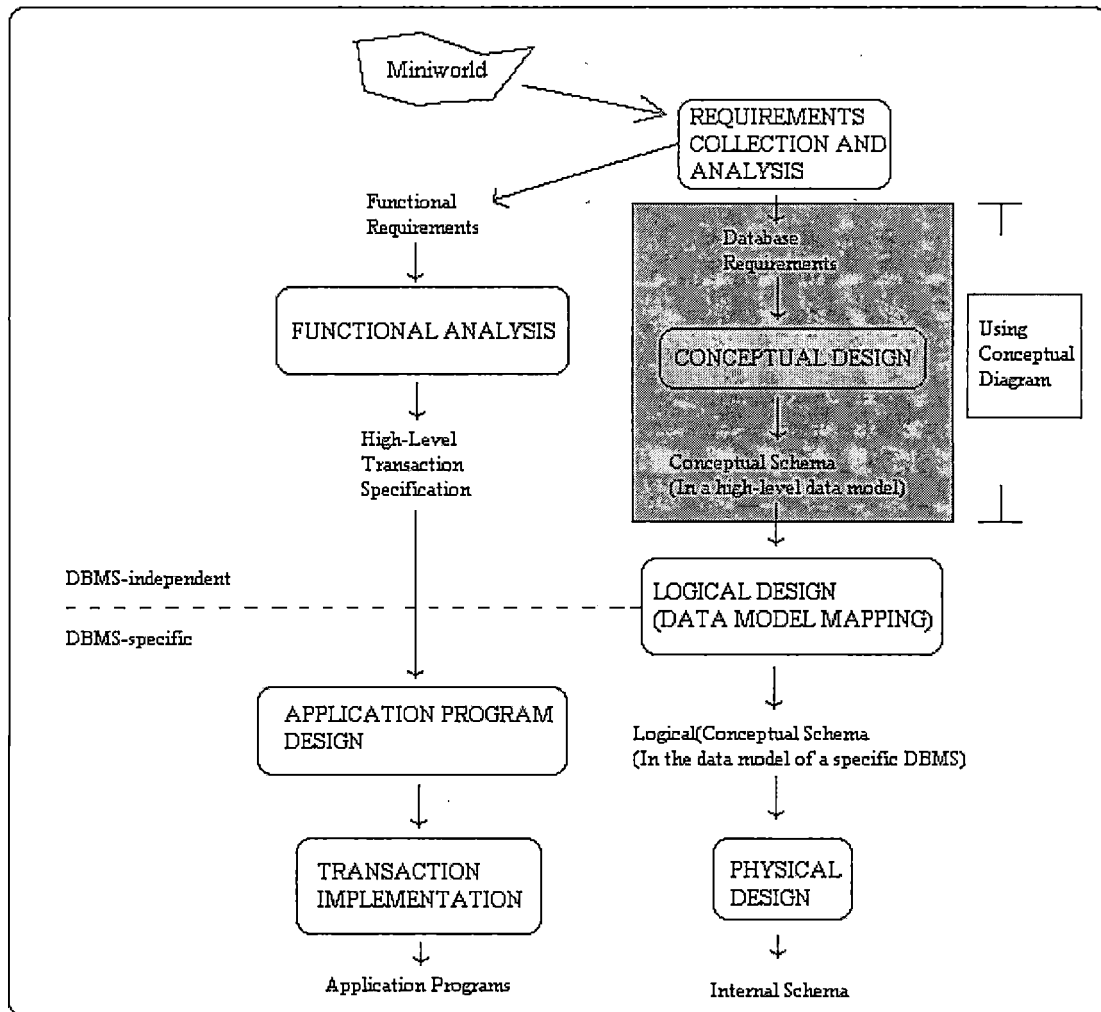


Figure 1. Phases of Database Design

In conceptual modeling, a diagram can be used to represent the overall logical structure of the database. For example, the E-R (entity-relationship) diagram is a data modeling technique that creates a graphical representation of the entities and the relationships between entities within an information system.

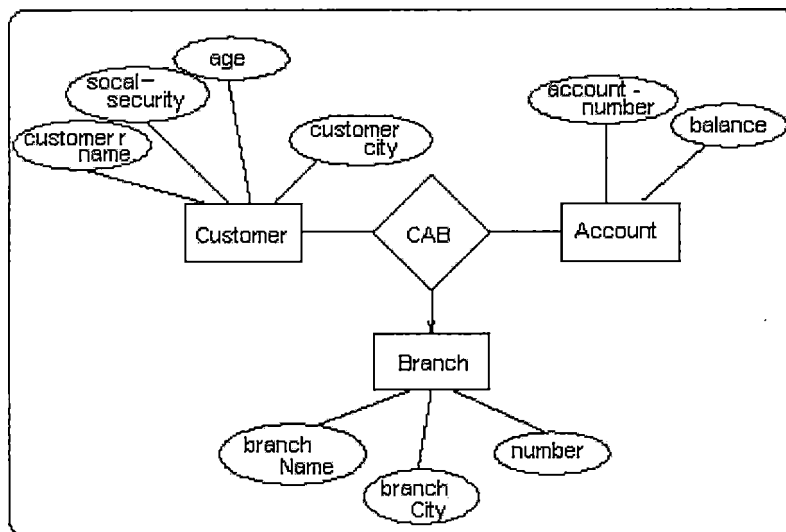


Figure 2. A Example of ER-Diagram









1.1.2 Why we Need a New Tool?

Several of the conceptual modeling tools have been developed to facilitate conceptual modeling designs. The most popular are ER/Studio, ERwin, Oracle Designer, Visio2000 and PowerDesigner. Table 1 compares these products.

However, these products have common weaknesses. First, these products only provide "one style" of

notation. There are different notations for representing an entity-relationship conceptual model.

Figures 3, 4, and 5 show different notations for entity types/classes, attributes, relationships, and specializations notation/generalizations.

1. Entity type/class symbols i  ii 
2. Attribute symbols i  ii  iii 
3. Relationship symbols i  ii  iii 
4. Notations for attaching attributes to entities and how detailed the information is. Figure 3 (iii) and (iv) are popular in OOA methodologies and in some case tools.

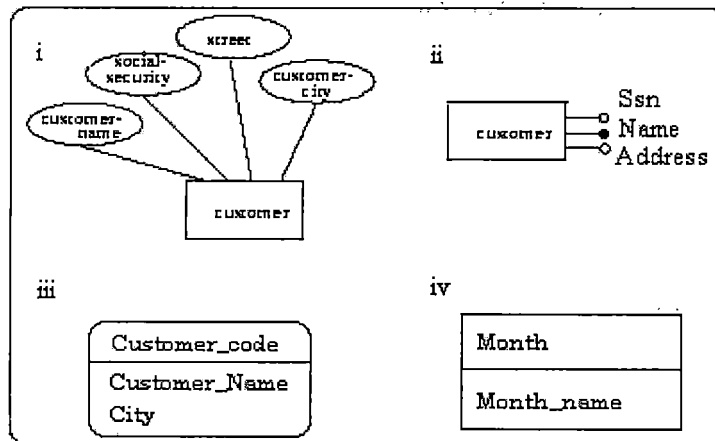


Figure 3. Alternative Notations, Entity Types, Attributes, and Relationships

There are also various types to demonstration relationships as well.

Figure 4(a) shows various notations for representing the cardinality ratio of binary relationships. Figure 4(b) shows several variations for displaying structural (min, max) constraints.

Second, for family users or students, the prices of the products are too expensive (See Table 1).

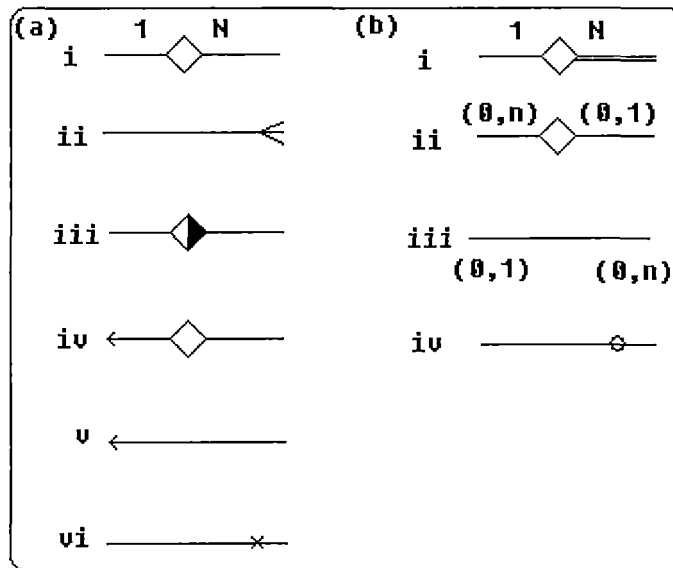


Figure 4. Cardinality Ratios and (Min, Max) Notations. Several Variations of Displaying Specializations/Generalizations

Third, some products are too complex or not user-friendly. Take ER/Studio as an example. It provides

multi-level designs, cross-project and reusable models which are only suited for enterprise users.

Fourth, some products can only be used on certain platforms.

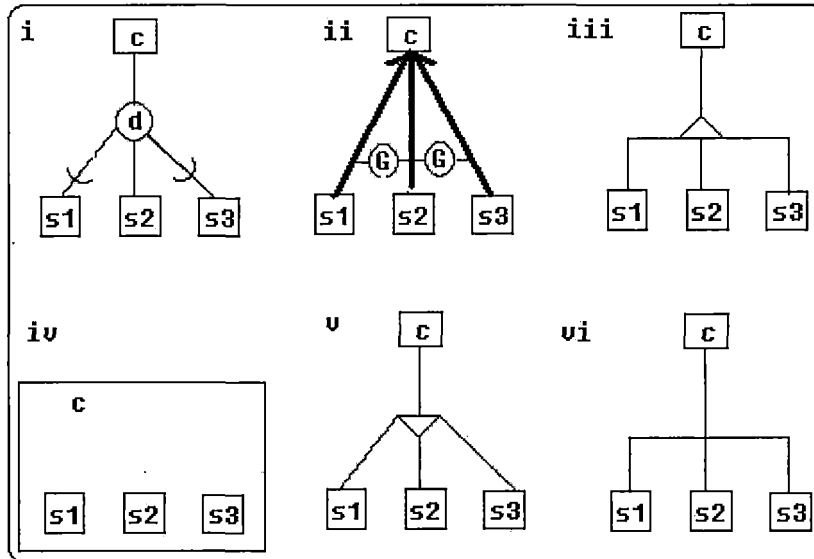


Figure 5. Alternative Specializations/Generalizations Notations

1.2 Purpose of the Project

This master project "Conceptual Model Builder" (CMB) will offer a Java-based conceptual diagram tool in several diagrammatic notations. It also will support a user-friendly graphic interface.

Since object-modeling methodologies such as UML (Universal Modeling Language) and OMT (Object Modeling Technique) are popular, CMB will provide UML-style

diagram. A user can switch among these different diagrams, or customize the conceptual diagram notations. A script generator which creates SQL statements for table definitions will also be available.

1.3 Significance of the Project

This project will benefit students in database design classes, professors who teach database system, and database programmers and consultants.

This tool can act as an excellent launching pad for students who desire advancement through database design. It is very simple to use and the help document is easy enough to understand.

1.4 System Requirements

This section lists the hardware and software requirements for installing CMB:

1. Hardware requirements:

- 64 MB RAM (128 recommended)
- 40 MB hard disk space (80 recommended)
- Pentium, Celeron, AMD or Cyrix processor
- 1024 x 768 Screen Resolution

2. Operating system requirements:

CMB requires the following minimum operating system versions. Some Java Virtual Machines (JVMs) and web servers have more stringent requirements.

- Windows 98/ME/NT/2000/XP
- Solaris 7, 8
- Red Hat Linux 6.2, 7.x,
- SUSE Linux 7.2, 7.3
- TurboLinux 6.5

3. Java requirements:

To install CMB, a user minimally needs a JRE version 1.3.0 or later on the user's computer. However, to take full advantage of all CMB capabilities, install the Java 2 Software Development Kit (SDK), sometimes called the Java Development Kit (JDK), version 1.3.0 or later. You can obtain the latest Java versions at <http://java.sun.com>

1.5 Limitations

Due to the limited screen size of the operating system, the current CMB version allows a conceptual model with about 23 entities and their relationship diagrams. On average, there will be about two to three persons who may work on one conceptual model. So a CMB server will be

designed to serve four remote users. But this can be extended if necessary.

1.6 Definition of Terms

The following terms are defined as they apply to the project.

Java - The premier solution for rapidly developing and deploying mission-critical, enterprise applications, J2SE provides the essential compiler, tools, runtimes, and APIs for writing, deploying, and running applets and applications in the Java programming language.

IP Address - The address of a computer when it connects to network. IP stands for "Internet protocol.

Port Number - The port number is a way to identify a specific process which is connecting to a server. The CMB uses the Registered Ports in the range 1024-49151.

Component - A component is a set of process to accomplish related tasks. A component may include many classes.

Model View - It is the display of a conceptual model on current CMB main graphic window.

Socket - A socket is a package for communications between two computers.

Swing - The Swing is one part of the Java Foundation
Classes for designing the Graphic User Interface.

1.7 Organization of the Project

This project document is divided into six chapters. Chapter one provides software requirement specifications, purpose of the project, and significance of the project, limitations, and definitions of terms. Chapter Two consists of the software design. Chapter Three documents the steps used in testing the project. Chapter Four presents the maintenance procedure. Chapter Five presents the users manual for the project. Chapter Six presents conclusions drawn from the development of the project. The Appendices containing the project code follows Chapter Six. Finally, the references for the project are presented.

CHAPTER TWO

SOFTWARE DESIGN

2.1 Introduction

CMB has five components: user interface, diagram engine, notation data, script and output, and network.

The user interface component will help users create the target conceptual diagram. The target diagram will automatically be generated by the diagram engine component.

The diagram engine component allows users to transform from one style of a conceptual model to another. This function needs the diagram engine to reference the notation data component.

The script and output component allows the user to save the diagram in a binary file. It also generates the SQL scripts to create the table definitions. These SQL scripts will be saved as text files.

The network component provides remote control through a network. CMB checks the user's ID and password to protect CMB from unauthorized access.

Detail of CMB components and techniques used will be described in the following subsections.

and an "Entities" window will allow users to make changes on data related to entities.

The south menu provides references for mouse movements. Three "look and feel" buttons provide the "metal", the "window" and the "motif" style of overall CMB appearances. These styles each provide a different graphic presentation of buttons, menus and windows. For example: when a user chooses to use the "window style", all the windows and buttons of CMB will completely look like a Windows 2000/XP application.

The tree menu displays additional information regarding tables and their attributes. Users can easily understand the structure of entities and their attributes. The current function on this panel only displays information. But more mouse-related operations can be implemented in the future.

The conceptual model panel reflects the graphic model view provided by the diagram engine. The default size of this panel is 800x600 pixels. This is larger than what a user actually can see when CMB starts. This means some areas of the conceptual model panel are outside the overall CMB display window, however two scroll bars are provided to aid the users. The vertical scroll bar allows users to scroll up or down the conceptual model. The

horizontal scroll bar allows users to scroll the conceptual model to the left or to the right.

A listener is also implemented in the conceptual model panel for retrieving mouse movements. This listener feeds movement information into the diagram engine and also launches the graphic drag and drop animations in response to current user operations.

2.3 Diagram Engine Component

The diagram engine component plays the main role of database demonstration. It has two major subcomponents. The concept model data component generates the model data and stores the data. Two classes are involved with data storage: The data for entities is handled by `tableDataObject` class. The data for relationships is handled by `relationDataObj` class.

Each class has its data section and method section. The major data structure for data section is linked lists. Choosing linked-lists as the main storage structure makes it easy for CMB to add, remove and modify data. Since there is no "order" on tables or relationships, a linked list is a better choice than an array or a vector. The methods of these two classes provide functions for user interface component and network component to acquire model

data. These functions include inserting, deleting or updating tables/relationships, summarizing total number of tables or relationships, saving/loading tables or relationships and transferring tables or relationships to another remote CMB application through the network.

The second subcomponent of the diagram component is the graphic generation component. This component has algorithms to analyze the concept model data and generate the graphic data for notation data component. This component has multiple classes. Each class supports one kind of conceptual model view. These classes also tell the user interface how to draw the graphics and responds to user input.

2.4 Notation Data Component

This component keeps the information on notations and the graphic data for CMB. It also has multiple classes. The control Mode class keeps the trace of the overall CMB setting like font types, the graphic size, notations status and so on. The remaining classes of notation data component store the graphic data generated by the diagram engine and provide functions for diagram engine to manage these data. Because the diagram engine of CMB provides more than one model view and each model has it own graphic

data, CMB uses multiple graphic data classes for each model. For example, the "DrawTableInfoVector" notation data class provides the graphic data for "EnterprisePanel" diagram engine class to exhibit the enterprise model view. The "DrawBasicPanelInfo" notation data class provides the graphic data for "BasicPanel" diagram engine class to show the student learning style model view.

2.5 Script and Output Component

This component retrieves the conceptual model data and generates a binary file for CMB. This component also is related to the user interface component for accepting the user's request to save or load files. There are three classes-- the save class and load class in the CMB handle the user's operation for Save/Load files. The script class generates the current conceptual model script for users.

2.6 The Network Component

When CMB activates its network service, a listener will start running. This listener is a stand-alone thread and therefore will not interrupt the rest of CMB operations.

Once a client is connected, its IP address along with other information will be recorded in a CMB data array. This data array helps the server to transmit and receive

any network package. The data inside these socket packages include something like data of entities and attributes, lock requirements, graphic information and so on. A lock request is a package issued by servers to keep data consistency. On the user interface, CMB provides a server panel and a client panel for users to receive connection information and monitor network activities.

2.7 Model-View-Controller

CMB uses Model-View-Controller technique to relate each component. The detail of this technique is described below.

The Model-View-Controller paradigm is a way of breaking an application, or even just a piece of an application's interface into three parts: the model, the view, and the controller. MVC was originally developed to map the traditional input, processing, output roles into the GUI realm:

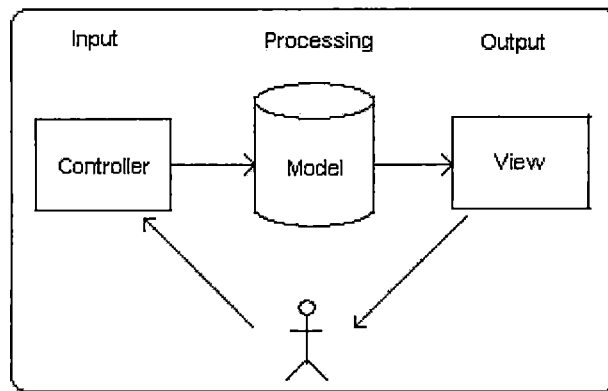


Figure 7. Model-View-Controller Architecture

The user input, the modeling of the external world, and the visual feedback to the user are separated and handled by model, view port and controller objects. The controller interprets mouse and keyboard inputs from the user and maps these user actions into commands that are sent to the model and/or view port to effect the appropriate changes. The model manages one or more data elements, responds to queries about its state, and responds to instructions to change state. The view port manages a rectangular area of the display and is responsible for presenting data to the user through a combination of graphics and text.

CMB combines the control and view together as a user interface component. This has been used in most of Java applications and speeds up the graphic performance.

However control and view are still different objects and perform different tasks in a CMB user interface.

The CMB components map to MVC as follows: diagram engine component and notation data component belong to the model; the user interface component and network interface component both have control and views.

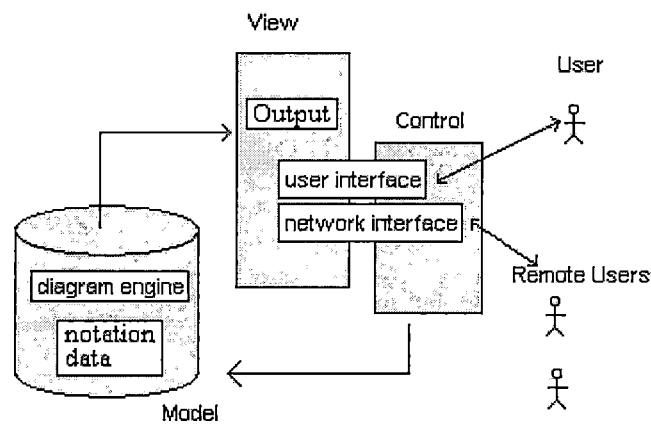


Figure 8. Mapping CMB to MVC

2.8 The Detail Design

2.8.1 CmbMenu Class

The CmbMenu class is the only class with the main() class and users must launch CMB from this class. It is also a data transfer center for other processes. Any update information will be sent to this class first then submitted to other classes.

The variables and functions of cmbMenu are described below.

extends JFrame implements ActionListener, MenuListener, Observer CmbMenu	
None	
+CmbMenu() +actionPerformed() +menuSelected() +menuDeselected() +menuCanceled() +main() +repaintAll() +update()	Initialize. A listener for the main toolbar. Call necessary processes for user request. Preserve. Preserve. Preserve. Activate the CMB. Repaint the main menu. Activate when model data has been modified by other components.

Figure 9. CmbMenu Class

2.8.2 TableObject Class

The TableObject object stores the model data for the diagram engine component. Each tableObject class matches one table. All tableObject class connects through a linked list like Figure 10.

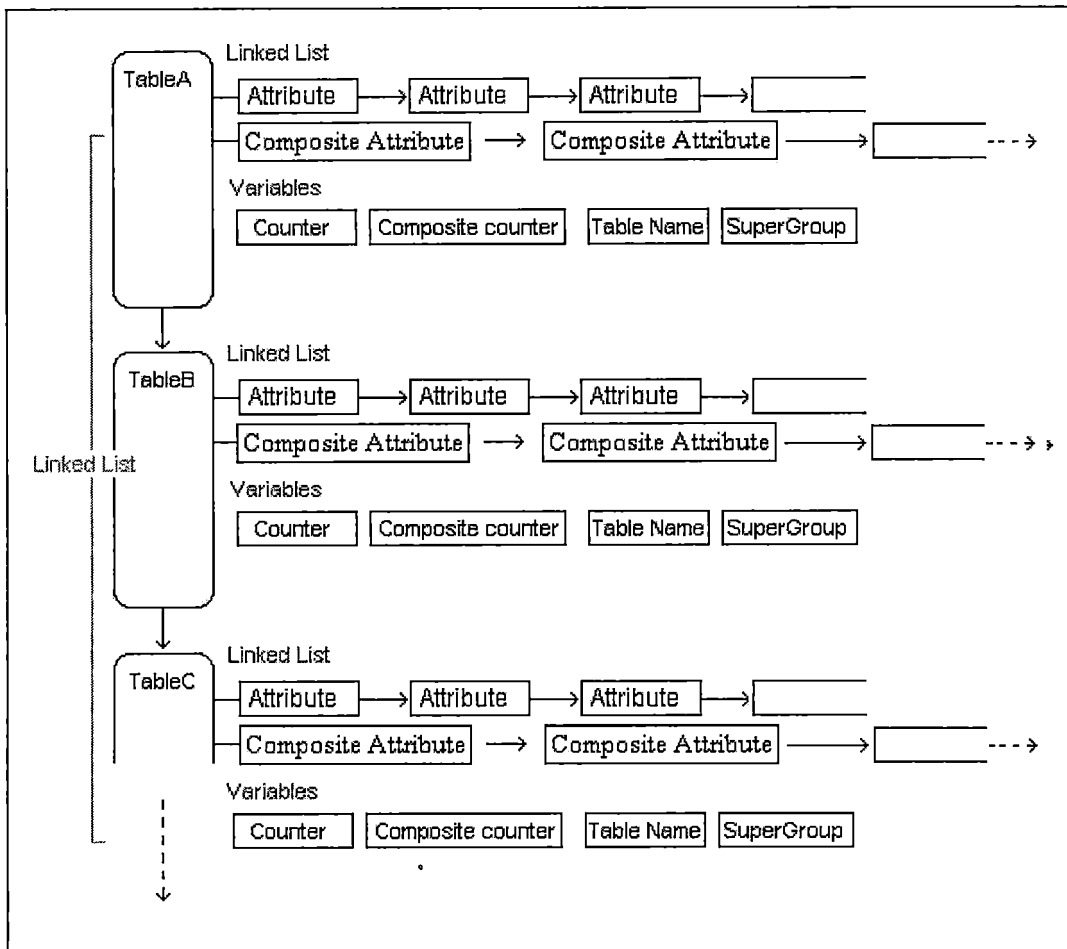


Figure 10. TableObject Class

The variables inside a tableObject are:

- Table name
- Attributes
- Composite Attributes
- Number of attributes
- Super class name (if any)
- Sub Group (only useful if it has a super class)

- Maximum character length of a table name.
- Maximum number of attributes

Except for attributes, most of variables are numbers and characters.

TableObject	
String[] emptyElement Int numbers, n, int subGroup String tablename String superClassName boolean superClass java.util.List attributes, comboAttributes static final int NAME_SIZE static final int ECORD_SIZE	
+ tableDataObj() + addAttribute() + addComboAttribute() + setSubClass() + returnAttribute() + returnComboAttributes() + String returnName() + setTableName() + getNumbers() + getComboNumbers() + getGroup() + getSuperClassname() + ifsuperClass() + writeData(DataOutput out) throws IOException + readData(DataInput in) throws IOException + print() + removeAttribute() + removeComboAttribute()	Initialize the data. Add another attribute. Add composite attribute. Record the sub classes. Return attributes. Return composite attributes. Return the table name. Modify table name. Get the total of attributes. Get the total of composite attributes. Get sub group number. Get name of super class. Check super class state. Write data to I/O. Read data through I/O. Print all data information to dos prompt. Delete an attribute. Delete a composite attribute.

Figure 11. TableObject Class

The attributes and composite attributes are two separate linked lists. Each element inside these linked lists is an array which stores the data of an attribute -- attribute name, attribute type, data type, and key type. Figure 11 shows the tableObject class.

2.8.3 DataIO Class

This class provides functions for converting between the characters and the string. This class is called when the save component wants to save or load files. It will translate the string into characters when saving and translate characters back to string when loading.

DataIO	
None	
+ readFixedString()	Read fix sized characters and output a string. Write a string as fix- sized characters. Read fix-sized characters and output a string (for network use). Write a string as fix-sized characters (for network use).
+ writeFixedString()	
+ readNetString()	
+ writeNetString()	

Figure 12. DataIO Class

2.8.4 Open Class

The Open class is designed to load binary CMB files. Once the file is opened, this class will check if it is a CMB file. If it is recognized as a CMB file, the open class will acquire the information about user, tables and

relationships from this binary file and update this information to the model data component.

Open	
boolean DrawOn = false java.util.List tableData relationDataObj relationData controlMode controlCode int counter=0,total=0 javax.swing.Timer activityMonitor	
+open() +cleanData() +print()	Initialize. Include opening the file stated by user and read the file. Throw I/O exception. Clean the previous CMB data on memory. Dump the file data for debugging.

Figure 13. Open Class

2.8.5 Save Class

Save class is the counter part of the open class. It saves all model data into a binary file. It also has a user interface component which will acquire the file name and target directory from users.

Save	
+boolean activeSave +boolean DrawOn +String path +java.util.List listTables +java.util.List tableData +relationDataObj +relationData +controlMode controlCode +int tableNumbers	
None	

Figure 14. Save Class

InsertTable	
<p>JPanel panelNorth, panelSouth, panelCenter JPanel comboPanelNorth, comboPanelSouth, comboPanelCenter JPanel tablesNorth, tablesSouth, tablesCenter Container contentPane JTextField IfSave, TextField1, TextField2, TextField3, TextField4 JComboBox DataTextBox, tableChoose, comboTableChoose, SuperCo mboChoose +JComboBox editSuperAttributeColumn +java.util.List tableData +controlMode controlCode +int columns, numbers, addAttribute, comboRowCounter +int newTableCounter +int tableIndex +String superAttributeChooseed +String[] columnNames +String[] tableModelNamesObject[] +Object[] newComboRow, +comboColumnNames +String[] CharType +DefaultTableModel DefaultTable +JTable table, +comboTable, tableList +JScrollPane ScrollTable, +ComboScrollTable, tablesScrollPane +extendDefaultTableModel model +extendDefaultTableModel_2 +comboAttributeModel +extendDefaultTableModel +tableModel +JDialog colorDialog +JTabbedPane tabbedPane +boolean isTableExist</p>	
<p>+addAttributeSouthPanel() +addAttributeNorthPanel() +add() +addAttributeTabbedPane() +setListOfAttribute() +setListOfComboAttribute() +addComboAttributeTabbedPane() +addComboAttributeNorthPanel() +addTableTabbedPane() +addTableSouthPanel() +addComboAttributeSouthPanel() +setListOfRelation() +setComboTableChoose() +setSuperAttributeChoose() +editSuperAttributeColumn() +repaintAll() +reflashTableChoose()</p>	<p>Generate buttons on the entities window. Generate table options Add a table. Add attribute Pane on the window. Display attributes. Display composite attributes. Add composite attribute Pane on the windows. Generate the composite attribute option. Add table Panel on the windows. Add the buttons to the south panel. Add the composite attribute option buttons. Set type of an attribute. Check if a table has composite attribute. Check if a table has super class. Edit a super class. Notify all processes about the change of table data. Make change for tables</p>

Figure 15. InsertTable Class

<u>InsertRelation</u>	
<p>JPanel panelNorth, panelSouth, panelCenter JPanel comboPanelNorth, comboPanelSouth, comboPanelCenter JPanel tablesNorth, tablesSouth, tablesCenter Container contentPane JTextField IfSave, TextField1, TextField2, TextField3, TextField4 JComboBox Data TypeBox, tableChoose, comboTableChoose, Su perComboChoose +JComboBox editSuperAttributeColumn +java.util.List tableData +controlMode controlCode +int colums, numbers, addAttribute, comboRowCounter +int newTableCounter +int tableindex +String superAttributeChooseed +String[] columnNames +String[] tableModelNamesObject[] +Object[] newComboRow, +comboColumnNames +String[] CharType +DefaultTableModel DefaultTable +JTable table, +comboTable, tableList +JScrollPane ScrollTable, +ComboScrollTable, tablesScrollPane +extendDefaultTableModel model +extendDefaultTableModel_2 +comboAttributeModel +extendDefaultTableModel +tableModel +JDialog colorDialog +JTabbedPane tabbedPane +boolean isTableExist</p>	
<p>+addTableTabbedPane() +addRelationsSouthPanel() +findRelNumber() +add() +editRelations() +checkRel() +repaintAll()</p>	<p>Add a relationship Panel to the windows. Add relationship option buttons. Get the relation number. Add a new relationship Modify a relationship Check if a relationship is legal before displaying it. Notify all processes about the change of relationship data.</p>

Figure 16. InsertRelation Class

2.8.6 InsertTable Class

InsertTable class is designed to modify model data. All the operations regarding tables and attributes are only allowed to be changed by this class. A user can add, remove and modify both tables and attributes through the panel provided by this class. Some basic input error handling is also included to make sure a user inputs legal data only. Any data modification will automatically be updated to other CMB processes.

2.8.7 InsertRelation Class

This class handles all the changes to relationships. Any data change to relationships needs to go through this class. Only the legal change will be displayed. The insertRelation class also generates a user interface for users to insert, delete or update data on relationships.

2.9 Summary

The software design of CMB includes various types of techniques: user graphic input/output, file system, operating system, sockets, thread and so on. The powerful Java code library provides great aids to this kind of software development.

Most of the work is related to the diagram engine component. To create a diagram engine class for any type

of conceptual model, an algorithm must be developed to tell the diagram engine class how to analyze the model data and generate the graphic data. After that, a data structure must be designed to store this graphic data. Then the designer tells this class how to transfer graphic data into graphic presentation. Next, the designer creates the mouse input method for this class. Finally this conceptual model view class is ready for use.

The last development component is the network interface. It still requires more functions in the future to improve stability and security.

CHAPTER THREE

SOFTWARE QUALITY ASSURANCE

3.1 Introduction

During CMB development, each class has been tested to ensure that it can handle the data correctly. Most tests are simply involved in printing out necessary data and allow the programmer to check if this process is running correctly.

The remaining tests focus on the user interface component. CMB issues this kind of test by adding graphic related method to display the hidden points or, adjusting mouse sensor range and so on.

3.2 Unit Test Plan

CMB performs unit test by implementing a print function inside each class. Any vital data in a class can be printed out during the CMB operation. These functions are disabled after data transfers are found to be accurate; however it can be enabled again for future development and debugging.

Figure 17 shows how a unit test function on CMB works. If a designer wants to make sure the table data is correctly saved, the print function can be included on both the load and save classes. After he runs the CMB and

performs saving and loading, a list of table information will show on the DOS prompt for checking.

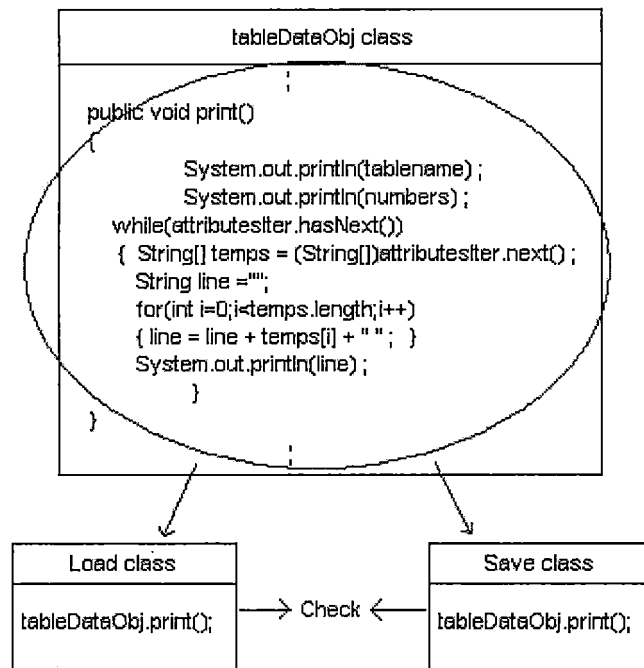


Figure 17. A Sample of Unit Test

Here is the list of classes which have passed the unit test.

Class name	Testing focus
tableDataObject	<ul style="list-style-type: none"> • Verify all required input data • Verify null value handling • Check all selected functions work correctly
relationDataObj.java	<ul style="list-style-type: none"> • Verify all required input data • Verify null value handling • Check all selected functions work correctly
controlMode	<ul style="list-style-type: none"> • Check all default data • Check all input information • Check all settings for the data • Verify function return value
open/save	<ul style="list-style-type: none"> • Check all data saved and loaded correctly • Check error handling message
drawtableInfoVextor	<ul style="list-style-type: none"> • Verify received data • Verify generated graphic data • Check any modify function works correctly for graphic data
DrawBasicPanelInfo	<ul style="list-style-type: none"> • Verify received data • Verify generated graphic data • Check any modify function works correctly for graphic data
passwdDialog	<ul style="list-style-type: none"> • Verify user input • Verify user name and password storage

Figure 18. The Unit Test List

3.3 Integration Test Plan

Since CMB is a Java application, it follows the object-oriented design. The risk of integration error is reduced to the minimum. For CMB, most of the integration tests focus on how a model view responds to the user input. The major integration tests are listed below.

Class name	Testing focus
InsertTable	<ul style="list-style-type: none"> • Verify all received input data • Verify null value handling • Check input function • Check delete function • Check update function • Check warning message
InsertRelation	<ul style="list-style-type: none"> • Verify all received input data • Verify null value handling • Check input function • Check delete function • Check update function • Check warning message
SouthMenu.java	<ul style="list-style-type: none"> • Check all button works perfectly • Check mouse location displays • Check internal data transfer
BasicERPanel	<ul style="list-style-type: none"> • Verify all received graphic data • check error handling message • Check if graphic output the correct data • Check each relationship connection points • Check each table connection points
EnterpriseERPanel	<ul style="list-style-type: none"> • Verify all received graphic data • check error handling message • Check if graphic output the correct data • Check each relationship connection points • Check each table connection points
OptionDialog	<ul style="list-style-type: none"> • Verify received data • Verify generated option button • Check any modify function works correctly for graphic data
RandomIntGenerator	<ul style="list-style-type: none"> • Verify generated random numbers

Figure 19. The Integration Test List

3.4 System Test Plan

The system test requires CMB to input simple database information. This test starts when all model data components and necessary graphic components are finished. The major system tests are listed below.

Data	Table
Tables	Insert Tables Delete Tables Update Tables
Attributes	Insert Attributes Delete Attributes Update Attributes Insert Composite Attributes Delete Composite Attributes Update Composite Attributes
Relations	Insert Relations Delete Relations Update Relations
Save/load	S/L original data S/L graphic Data
Display	Display table names, Attributes, Keys, Display attributes names, Data Type, Data Value, Key Type, Multi Value, Composite Key, Display Relations Name, From, To, Active

Figure 20. The System Test List

3.5 Summary

The most difficult test was addressed on the diagram engine component. There are no error messages on graphic appearance. If something goes wrong, the entire panel may be frozen or displayed in a strange way. When this happens, each graphic data component must be checked.

Some graphic problems are hard to detect by the human eyes. During CMB development, such problems only could be founded when the user inputs massive data or does a graphic test on it.

The null value is another problem in CMB design. Sometimes when a user deletes something from a model data, it will create a null value on the graphic data until the graphic data has been renewed. This will affect the user interface component since it does not allow null value during data to graphic transformations.

But every known CMB test problem has been dealt with and these code improvements should reduce the bug of future CMB design.

CHAPTER FOUR

MAINTENANCE

4.1 Introduction

This chapter consists of four parts: the model data maintenance, the graphic engine maintenance, the user interface maintenance and the network maintenance.

4.2 Model Data Maintenance

As mentioned earlier, model data includes two major classes: `tableDataObject` class and `relationDataObject` class. Both classes have two linked list data structures to store data. Each unit of a linkedlist is an object. One linkedlist stores the string objects and another stores the arrays of integers. Using array makes it easy for other components to acquire the data from fixed array positions and avoid long search time.

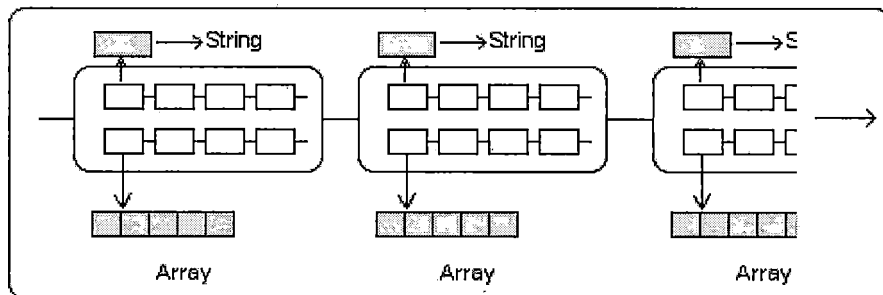


Figure 21. The Major Model Data Structure

If any future design requires more information storage, a designer can enlarge the size of integer array without any problem. If a designer wants to add more strings, s/he can change the first linked list to a different type of object which allows for more strings. S/he must modify the methods related to this linked-list as well, but there is no need to modify anything outside this class. Adding another linked list and creating its methods are also allowed.

4.3 The Graphic Engine Maintenance

The graphic data class is similar but more complicated than the model data. Multiple graphic data classes are allowed and each of them stores particular graphical data for its model view. A designer creates a new model view by creating a new class to store the data and a new interface to read this data and display it. So what is inside a graphic data class depends on what kind of model view it stores and how the interface works.

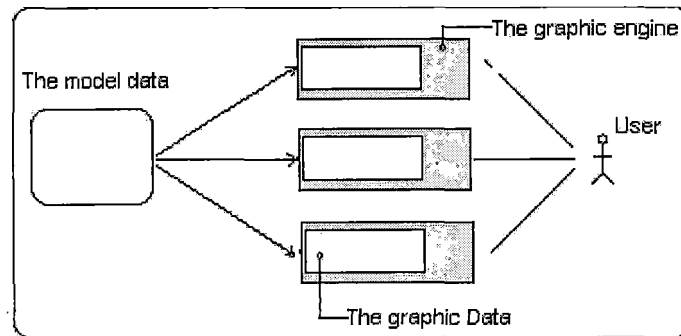


Figure 22. The Graphic Engine

There are some important concepts on creating or maintaining these graphic data classes. First, a graphic data is the presentation of the model data so major structures should better be linked lists. Second, it should have some mouse movement detection methods for the interface to respond to the operations from users. Third, it should provide 'load', 'save' and 'transfer' methods for files and network purposes.

4.4 User Interface Maintenance

The user interface component is a huge set. It includes classes for saving and loading files, classes for displaying a main menu, classes to activate the network connection, and classes to change the model views.

4.4.1 Main Menu

The class for main menu is cmbMenu class. This is the class where everything is started. It generates the initial model data and activates the remaining interface

when necessary. Any new submenu bar and submenu units should be added here. The new model panel will also be declared here.

4.4.2 Save/Load

There are two classes for this interface: save and load classes. Both classes use the data stream to transfer the data. The user interface of these two classes come from the Java standard library and requires few modifications.

4.4.3 South Menu

The southMenu class handles this part. It is possible to add more overall look-and-feel style buttons here. Some other user information also can be included here.

4.4.4 Tree

The tree class is the simplest component. It is the best place for future designers to add new features. The connection between tree and cmbMen is already established. This makes it easier for future development.

4.4.5 The Model View

The model view display is based on which view the user has chosen from the main menu. The cmbMenu keeps updating every model view class but only the one chosen by the user will be displayed. These views also include the functions for mouse drag-and-place on the top of the class.

4.5 Network Interface

Two classes are included in this interface. The `cmbServer` class receives information from the clients and sends necessary information through network. The `cmbClient` class makes connection with the server for sending or receiving socket package from the server.

As shown in Figure 2.2, when the server process is launched, a "listener" will also be generated by the server to accept connection requests. A list will be created to keep track "where" and "how many" clients are currently connected to the server. The maximum number of clients is set to 4.

A list of IP addresses for possible servers is located in the file called `client_list.cfg`. When a client is started, it will read this test file to generate a list for the user to choose the IP address for connection. If the user enters or chooses the correct IP address for the server, a connection can be made by pushing the connect bottom.

Once the server accepts the server request, the server listener will then call the server process to create two additional processes, one is called the "sender" and the other is called the "receiver".

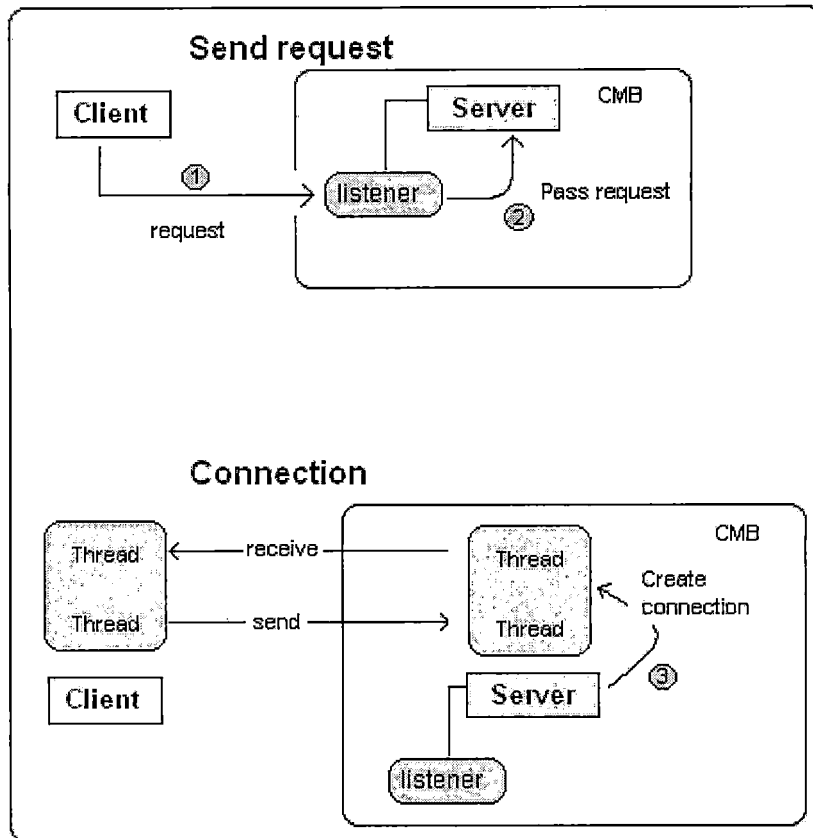


Figure 23. The Connection Process

In the mean time, the client side will also generate two processes corresponding to the server. The "sender" of the client will connect to the "receiver" of the server. The "receiver" of the client will connect to the sender of the server.

The client "sender" will be responsible for packaging any client request and send it to the server. And the client "receiver" will be responsible for any messages received from the server. Another class called "share

area" will provide data exchange between these two processes.

An example for data transferring is as follows: First, the server will send original data to each client. After this data transfer is complete, any client can then request a data lock for modifying data but only one request can be accepted at the same time. After data modification like add, insert or delete has been made and the client sends the release lock signal to the server, the server will unlock the data for other clients. The client also can modify graphic presentations if desired to do so.

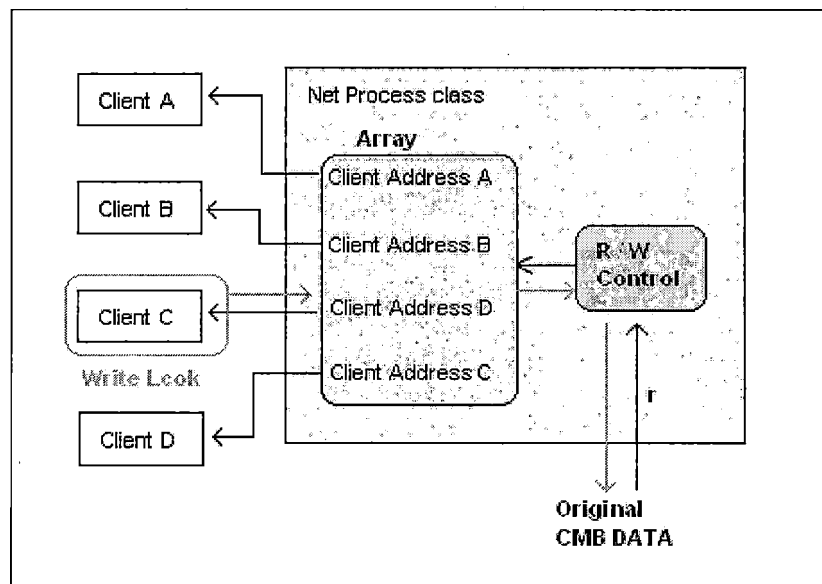


Figure 24. The Connection Process 2

4.6 Summary

Since Java is fully object-oriented, a future designer can find it is easier to improve the CMB programs. As shown above, CMB divides different tasks into different classes. It is easy for designers to make a modification on any component and activate it without encountering any serious bugs. For example, a designer can declare more data and methods on the model data class or the graphic data class without modifying any other interfaces. Then the designer can improve the necessary interface to "pick up" this new data inside the model or graphic data class. The most time-consuming work will be changing the model view interfaces. To allow a user to switch between different views and enjoy the same functions, it requires the designer to write these functions on each model view interface. Since model views are different from one another, the chance to reuse the same methods on all model view interfaces is slim. Java provides several network methods for designers so it is easy to improve the network functions on CMB.

CHAPTER FIVE

USERS MANUAL

5.1 The Main Menu

A menu bar on the top provides pull-down submenus. In the middle is the display panel. The left panel displays a tree which includes the entities and attributes. The right panel provides display a conceptual view. On the bottom is the system bar which provides overall options for appearance and the current the mouse location.

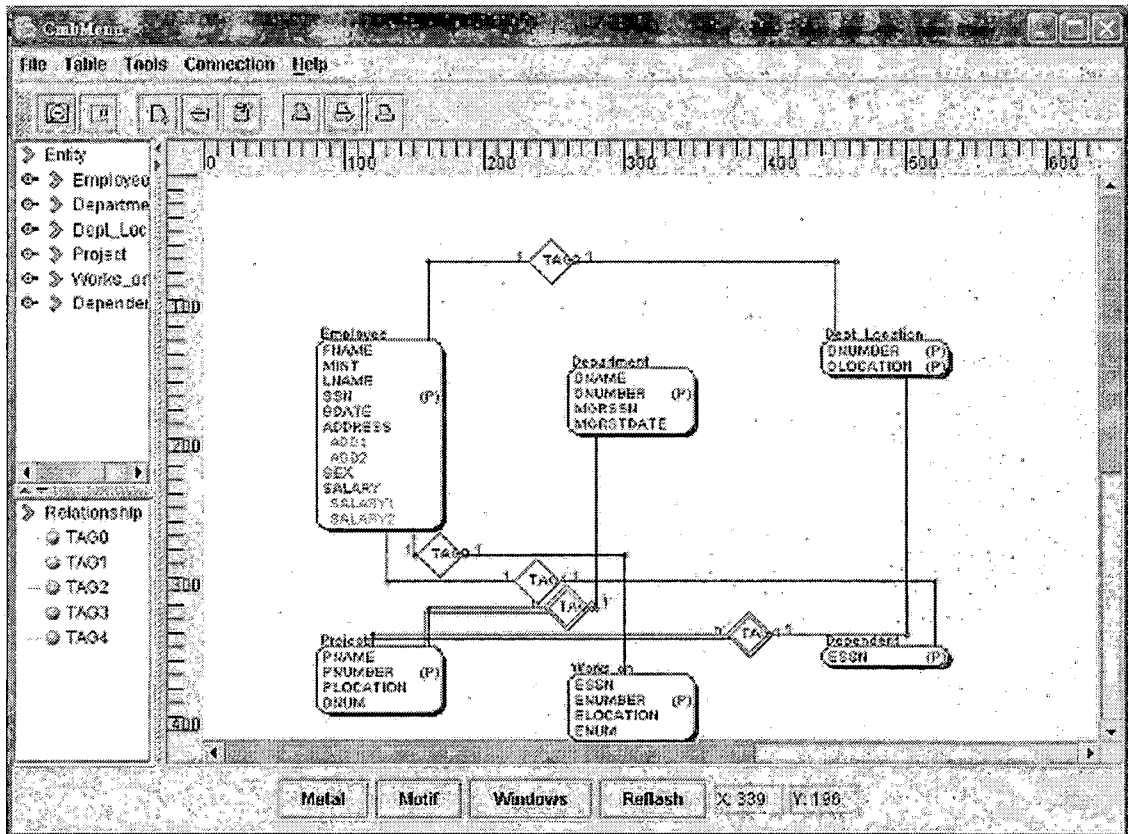


Figure 23. Main Menu

The initial panel size of the main menu is 800*600 pixels but it can be resized according to user's needs.

5.2 The Model View

CMB provides multiple model views. These views represent the same conceptual model. A user can switch between different views using the Tool->Options submenu at anytime. The component of these views can be changed by dragging the left mouse button to the new location. The component will be pasted when the user releases the left mouse button

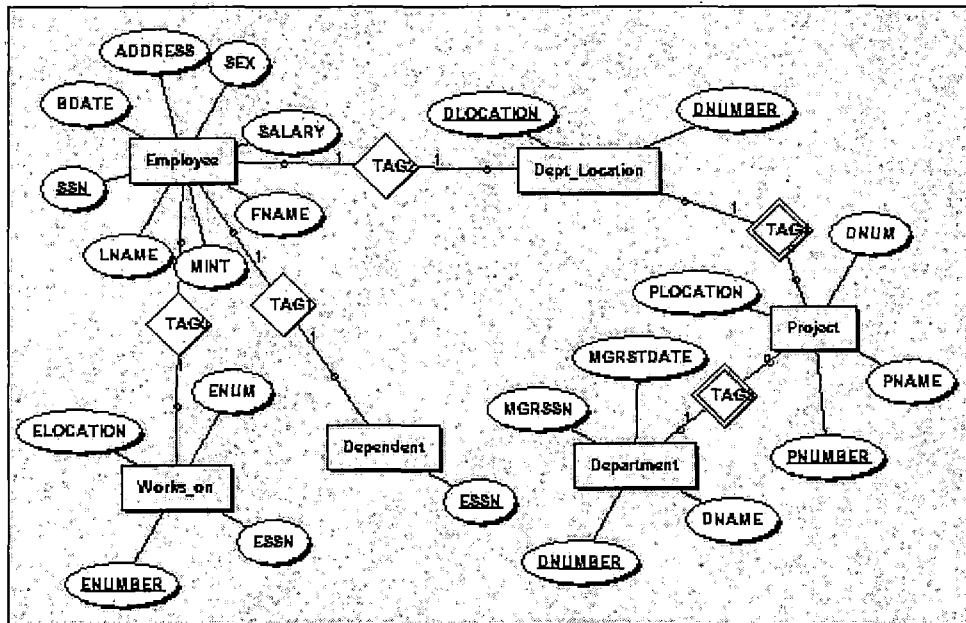


Figure 24. One of Model View

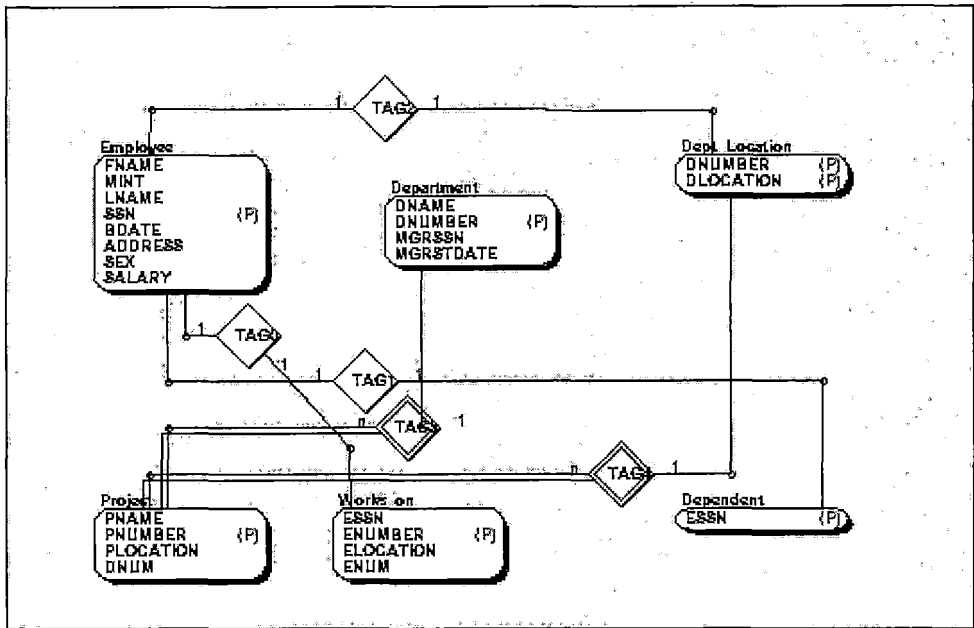


Figure 25. Another Model View

5.3 The South Submenu

The South submenu bar provides three types of main "look and feel" functions for users: the metal type, the windows type and the motif type.

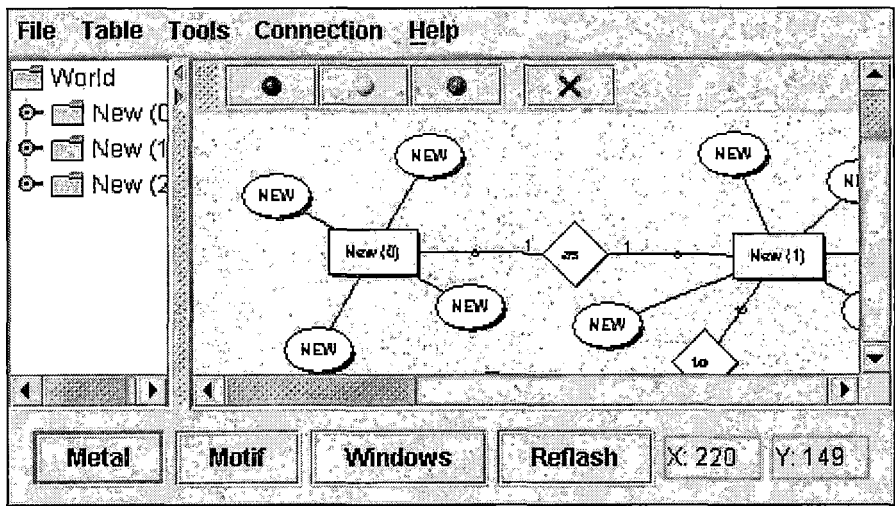


Figure 26. The Metal Look And Feel (Part 1)

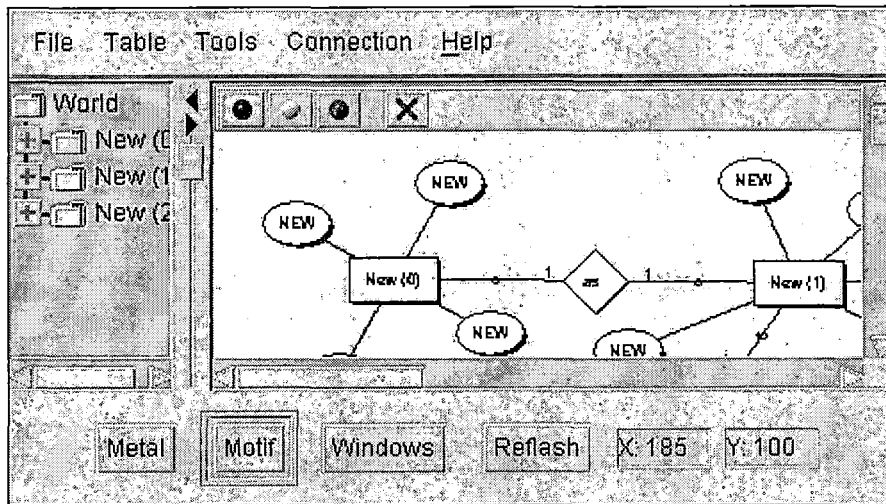


Figure 27. The Metal Look And Feel (Part 2)

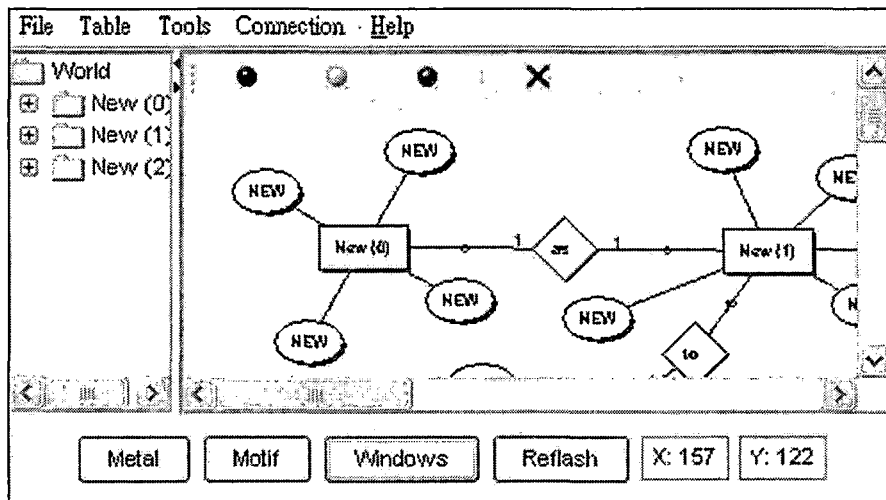


Figure 28. The Metal Look And Feel (Part 3)

The location of the mouse in the CMB panel is given in X and Y coordinates.



Figure 29. The South Menu Bar

5.4 The File Submenu

This menu provides operations like open/save files, user login and print functions. Each operation has its own interface.

5.4.1 Login

The login interface is activated when a user clicks the password icon on file submenu. To access protected files or connect to a server, a user must enter his user name and password for authorization.

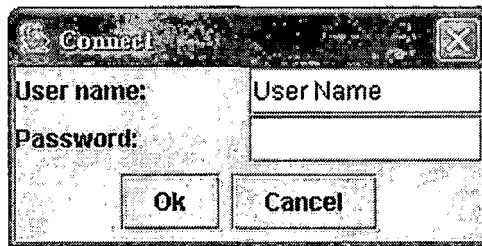


Figure 30. Login Interface

5.4.2 Save

When a user saves a new file or chooses "save as" option, this interface will be displayed. If a user click in the save function and the current model is already file name exists, CMB will save the contents to this exists filename and will not displaying pop-up panel. If a user

wants to avoid overwriting the original file, the "save as" function must be used.

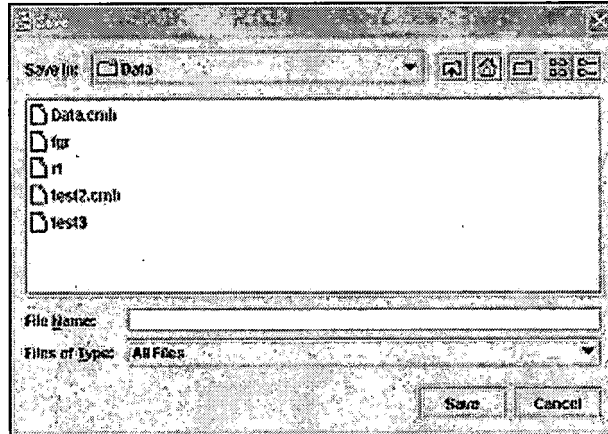


Figure 31. Save Interface

5.4.3 Open

This interface allows you to open an existing CMB file. A user can double click the left mouse button on a target file or click the open button to open it.

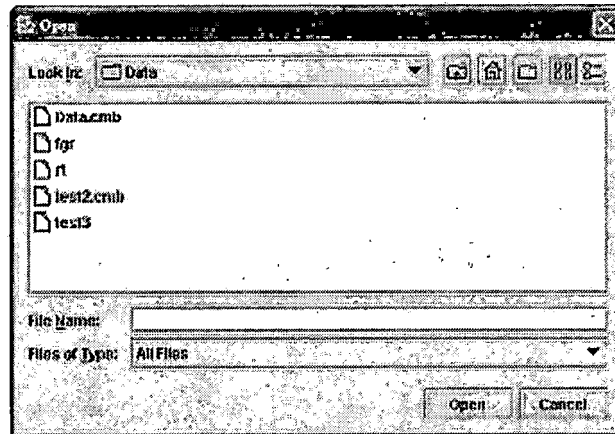


Figure 32. Load Interface

5.4.4 Print

This interface allows you to choose the type of printer and set the number of print pages.

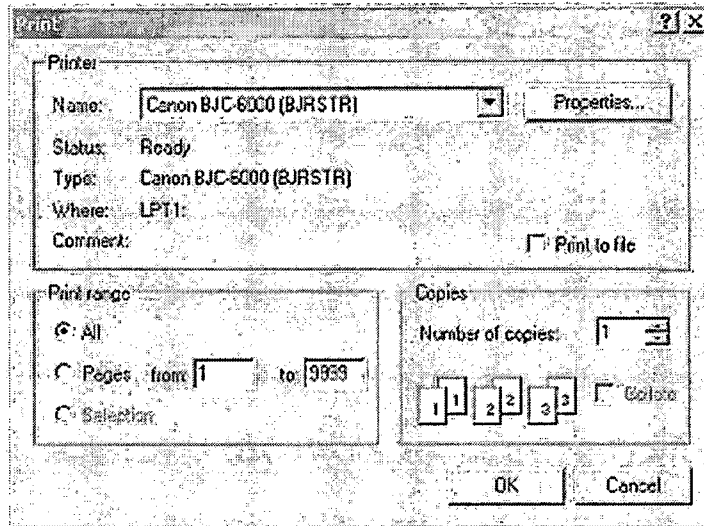


Figure 33. Print Interface

5.4.5 Page Setup

This interface displays a setup panel. The user can setup page type, margins and orientation (portrait or landscape) here.

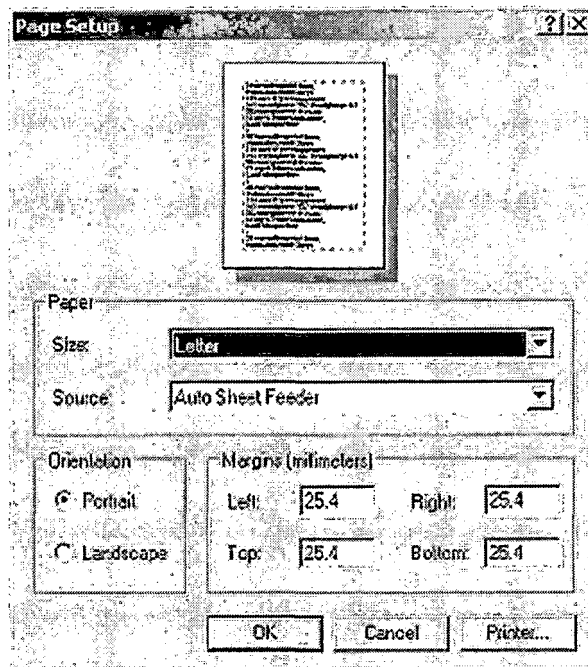


Figure 34. Page Setup Interface

5.5 The Table Submenu

The table menu provides tools for inserting, deleting or modifying entities and relationships.

5.5.1 Entities

This interface provides functions required to input information about entities. A user can add a new entity, rename an existing entity here. A user can remove an entity if this entity has no relationships.

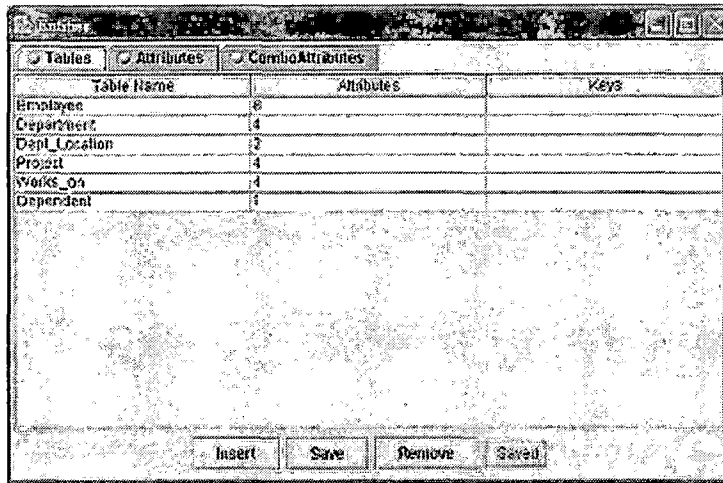


Figure 35. Entities Interface

5.5.2 Attributes

This is a sub panel of the entities interface. First, a user chooses an entity from the list of existing entity names. The attributes for the selected entity will automatically display on the table. Users can change values of these attributes. A user also can insert or remove a target attribute. The result will be directly displayed to the model view panel.

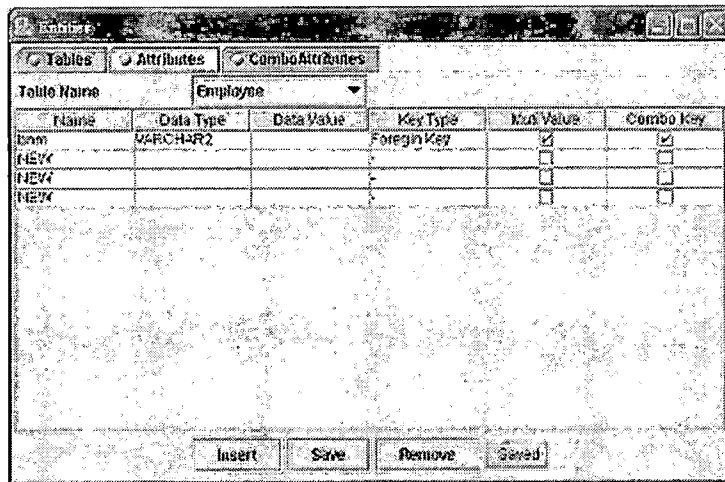


Figure 36. Attributes Interface

5.6 The Tool Submenu

The tools menu provides interfaces for additional graphic options.

5.6.1 Setup Font

Any change the user sets here will change the font type on all model views. Fonts of the current computer operating system will be automatically included in the font list. A user can choose any fonts he wants but some of the fonts may not be suitable for display. The font size range allowed is from 8 to 18. The styles are plain, bold or italic. A sample of chosen font is displayed in the middle panel. The default button allows users to set back the original font setting. The original font setting is Sans Serif, plain with size of ten.

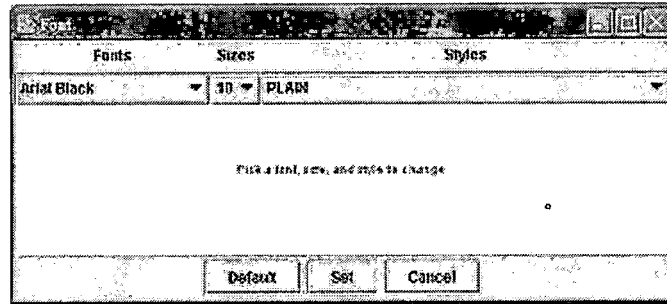


Figure 37. Setup Font Interface

5.6.2 Options

A user can change the model here.

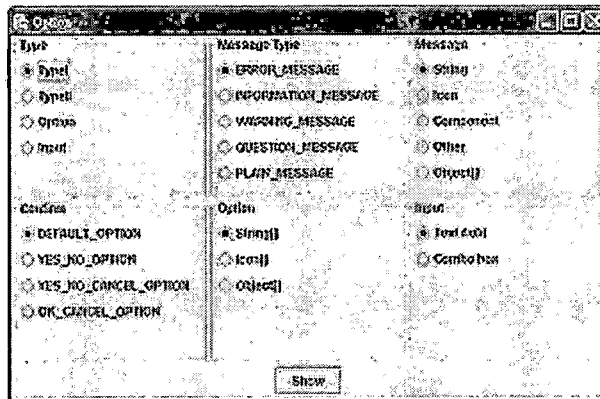


Figure 38. Option Interface

5.7 The Connection Submenu

This submenu provides two interfaces: server and client

The server interface will deal with data transfer; it accepts incoming connections.

The client interface provides the tool to connecting to the server.

5.7.1 Server

This interface allows a user to connect to other CMB applications. The current machine IP address and port number are displayed on the interface. The set port panel will display you how many users are connected to the server at this time.

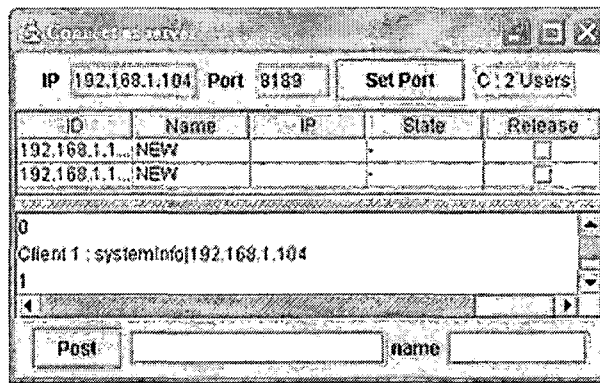


Figure 39. Server Interface

5.7.2 Client

If a server exists and a remote user knows the server's IP address and port number, a connection can be made by inputting these information into the panel and clicking the connect button.

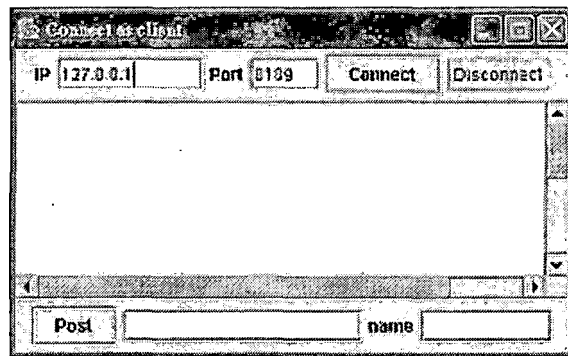


Figure 40. Client Interface

CHAPTER SIX

CONCLUSIONS

6.1 Introduction

The CMB development environment is well supported by Java. Although the graphic user interface of CMB may not be as colorful as those Microsoft applications developed using Microsoft Foundation Class or Visual Basic (Java graphic engines aim to suit every platform, not only one), the multiple-platform support and low bug environments make Java the best choice for developing a multiple-purpose user application like CMB.

The current CMB do not fully utilize the powerful Java network capability but still benefits from it. There is no doubt that more improvements will be required. The "Future Design" section will provide some tips on how to improve the CMB ability.

The last section will summarize the CMB. This chapter lets users understand more about the uniqueness of CMB. This chapter is also gives users a better idea on how to use CMB and hopefully, make users can benefit from it.

6.2 Future Design

The possible improvements which can be made for CMB are listed below:

6.2.1 Extend Model Views

The most feasible improvement will be modifying the current model views. The graphic display can be more colorful or detailed than the current one. Also, adding another model view can be helpful to users. Most of the current CMB source code is reusable, so it will be easier for the designer to make these changes.

6.2.2 Improve Menu Functions

In the main menu bar more submenus can be added to assist users. For example, adding a "redo/undo" submenu allows users to go back to their previous actions and adding a "screen control" submenu controls the screen size of CMB.

More detailed function can also be added in the submenus. For example, adding "sort" button sorts the information displayed on the window and adding a "print" button prints out the information provided by submenu.

6.2.3 Improve the Print Preview Features

Current version of CMB only provides the simplest printing preview ability. This preview only shows the border of printable area. It should be required in the

future to provide more knowledgeable preview functions and add a window to display the expected print output.

6.2.4 Add More Managing Tools for Network Component

Current server system only has mirrored administration ability. This is not enough if too many users are on-line and use CMB at the same time. Also, if CMB proves to be useful and become a frequently used application, additional network security functions like transmitting protection are necessary for future CMB design.

6.2.5 Make Graphic User Interface More Friendly

A system's graphical user interface along with its input devices is sometimes referred to as its "look-and-feel". CMB aims to provide a "friendly" user interface, but there are still many possibilities for improvements. For example, a pop up menu can be added for users to click the right mouse button and some table information can be displayed when a users puts the mouse pointer on a table for a while.

6.3 Summary

The CMB project provides a new approach for designing the conceptual diagram tool. The idea of providing different model views makes the polymorphous graphic

appearance possible. CMB provides "One concept model with multiple graphic presentations". Users no longer need to adapt to limited diagram tools. CMB will satisfy the needs of the users in most cases.

The automatic graphics-generation capability of CMB makes it an easy-to-use tool. Users no longer need to learn how to put the diagram together. Now they can finish their concept diagram easier and faster.

The use of Java also makes CMB flexible for updating operation systems. Unlike other applications, CMB users never need to worry about changing systems. Although CMB is a client-server application in its current design, it is easy to make CMB become a Web-based application if new techniques related to applets are released. Current applet platforms will take a lot of time to load the CMB to the client.

The network capability of CMB also makes it ideal for teamwork projects.

APPENDIX
PROJECT SOURCE CODES

The Conceptual Model Builder source codes are included in this disk.

REFERENCES

- [1] Ramez Elmasri and Shamkant Navathe, *Fundamentals of Database System*, third ed. Elmasri Navathe, 1997.
- [2] Embarcadero, "ER/Studio",
<http://www.embarcadero.com/products/erstudio/index.asp>, 2002.
- [3] Osmar R. Zaï Ane, "Other Styles of E-R Diagram"
<http://www.cs.sfu.ca/CC/354/zaiane/material/notes/Chapter2/node9.html>, 2002.
- [4] Cay S. Horstman, Gary Conell, *Core Java 2 - Volume 1 Fundamentals*. Sun Microsystems, 1999.
- [5] Cay S. Horstman, Gary Conell, *Core Java 2 - Volume 2 Advanced Features*. Sun Microsystems, 1999.
- [6] Deitel[1999], *Advanced Java 2 Platform -How To Program-*. Prentice Hall, 1999.
- [7] Dean Helman, "Objective Toolkit Pro whitepaper",
<http://ootips.org/mvc-pattern.html>, 2003.