Visible® Analyst Tutorial

Visible Systems Corporation
24 School Street, 2nd floor
Boston, MA 02108
617-902-0767

https://www.visiblesystemscorp.com
https://twitter.com/VISIBLECorp

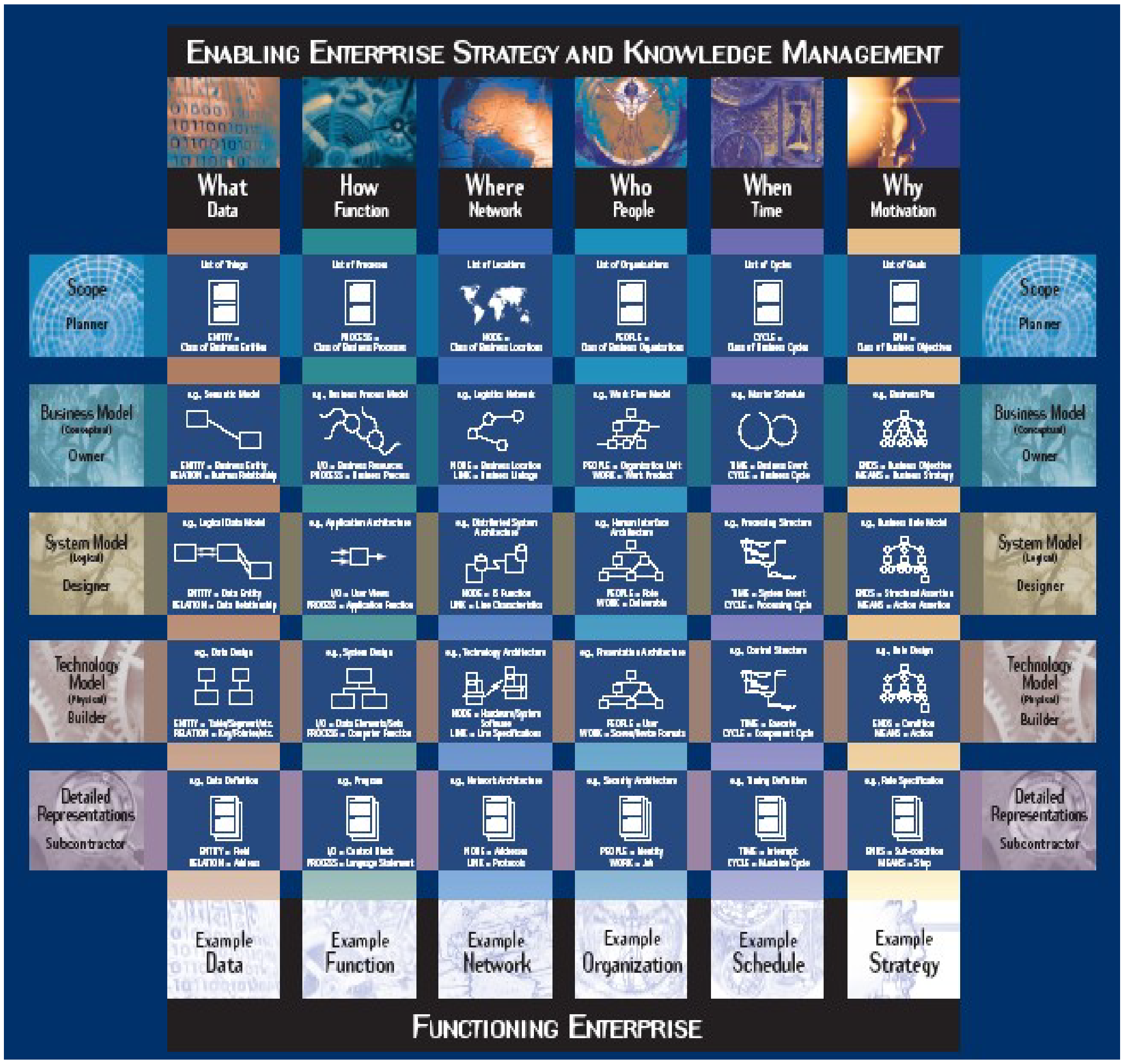
Email: contact@visiblesystemscorp.com

Visible®

Visualize. Align. Transform.™

Visualize patterns, align strategy and transform change into meaningful business outcomes.

Enterprise-wide Analysis, Design and Planning for Improvement.



This manual was prepared using Microsoft Word for Windows.

Visible Analyst
Tutorial on Structured Methods, Repository Management and The Zachman

Framework Visible Analyst® is a registered trademark of Visible Systems

Corporation.

Dear Colleagues:

Thank you for your time in selecting our product, the Visible Analyst. At Visible, we take your time and effort seriously. To that end, we pride ourselves on delivering the most appropriate, value-oriented solutions. We feel that we offer the very best in product support that often differentiates us from our competitors.

As you read though the tutorial, please take the time to understand that our approach to software development is one of a model driven approach. Within the framework of this approach, Visible, in part, supports the Model Driven Architecture (MDA) as defined by the Object Management Group (OMG). This group, commonly referred to as the OMG, is an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise-wide applications. For more information about the OMG and in particular their MDA specification, please reference their web site at http://www.omg.org/mda/.

In conjunction with a model driven approach, Visible has incorporated a framework to enable you to better plan and manage your Enterprise Architecture effort. In this edition, The Zachman Framework, is the framework of choice. However, you can customize the Visible Analyst to implement other frameworks like, for example, the US Federal Enterprise Architecture Framework (FEAF).

Visible Systems Corporation. Visible Analyst, Visible Developer, Visible Data Governance, Visible Web Portal, Visible Self Service Data Discovery, Visible Sight (Context-driven business insights), Razor SCM, Polaris (Task Management).

Getting to Know Visible Analyst

INTRODUCTION

The Visible Analyst Zachman Edition provides a Model Driven approach for defining, designing, building, testing, documenting and supporting Enterprise Architecture (EA), information systems and software products. Model Driven Architecture (MDA) tools are based on logical dissection of the real world into understandable models, processes and components. MDA tools provide mechanisms for evaluating current information activities, defining proposed changes, producing and validating new information processes and focusing on changes that will enhance the performance and operation of the organization. The successful use of MDA tools requires an understanding of the underlying concepts and logic and a comfortable knowledge of the operation and use of the MDA tool.

Visible Analyst has been created to make the implementation of MDA techniques a logical, flexible, natural and easy-to-perform process. Visible Analyst is a seamless MDA tool that integrates all phases of planning, analysis, design, code generation, and reverse engineering. Visible Analyst provides facilities for the development of function, object/class, state transition, data, data flow (process), activity, Use Case, sequence, collaboration, and structure chart (product) models for an information system. With the introduction of the Business Process Modeling Notation (BPMN) the Visible Analyst provides a modeling notation that can be communicated to and understood by all business users, from the business analysts developing the models, to the technical analysts implementing the model processes, to the business people who manage and monitor the processes. An integrated repository containing all defined model elements, extensive additional component definitions and free-form notes and definition fields provides a continuous life-cycle library of the design and development process. The Visible Analyst repository is used for reports of project content and to generate various forms of schema and application software code.

These lessons have been designed to lead you through the Visible Analyst mechanics and to demonstrate how easy Visible Analyst is to use. These lessons cover the entire development process, from drawing functional diagrams to generating program code. You can follow the lessons in sequence or you can select just the ones of interest to you. Like Visible Analyst itself, you have the flexibility to use any piece of the tool in any order that is reasonable within the project.

The tutorial also provides you with some insight into MDA concepts and underlying logic. These concepts are basically simple and logical. They allow you to break the complex real world into smaller and more manageable chunks that can be defined quickly and then be used to build operational pieces that *work* in the complex real world. Each of the MDA models provides a different view of the real world. Visible Analyst ties these models together and provides a vehicle for using them to define and evaluate current information operations. Proposed changes in the information processes, procedures and sequences are reflected into the MDA models and then are used to build a new set for the proposed change operations. The analysts, designers, developers and users interact with the Visible Analyst models and data repository to verify and validate the information steps and procedures for their organization and operations.

Once the architecture of the new information system is considered sound and solid, the software designer moves to defining and building the new product components and the software code. Visible Analyst supports the development of physical programming modules through the structure chart model. It also supports the definition and recording of pseudo code in the Visible Analyst repository. From these definitions and the data model, Visible Analyst generates database schema, SQL code and application shell code. Test plans, sequences, test cases and scenarios can also be generated in the repository notes fields.

One new feature of the Visible Analyst has been the additional support for the Business Process Modeling Notation based on the Business Process Modeling Initiative developed by the Object Management Group (omg.org). The complete specification can be downloaded from the OMG website, www.omg.org. The primary goal of BPMN is to provide a modeling notation that can be communicated to and understood by all business users, from the business analysts developing the models, to the technical analysts implementing the model processes, to the business people who manage and monitor the processes. The BPMN models describe the sequence of business processes with support for parallel and conditional behavior.

FAST TRACK USERS

Those who like to work on the Fast Track should read Lesson 5 - Diagramming Basics and follow the steps for creating a project, creating a diagram, and some optional settings that are available with Visible Analyst. Lesson 5 gives you the basic skills for working with Visible Analyst. We recommend that you work through the other lessons to discover the more advanced features that make Visible Analyst a powerful tool. Throughout the tutorial are references to features that are not demonstrated in the tutorial but that may be of interest to you. You can find more information about these features in the *Operation Manual,* which can be downloaded from our Web site using this link https://www.visiblesystemscorp.com/Products/Analyst/manual.pdf. The online help feature in Visible Analyst, accessed from the Help menu or by pressing F1, also provides you with more information on the referenced subjects.

Note

🗎 Since Visible Analyst is available in multiple configurations, the software you purchased may not include all of the diagram types or advanced features described in these lessons. The basic drawing techniques apply to all diagram types, and you are encouraged to work through the brief exercise in Lesson 5 - Diagramming Basics. Thereafter, you can skip chapters that do not apply to your Visible Analyst package.

OVERVIEW OF MDA CONCEPTS

MDA concepts involve creating and defining different models or views of the real world and then using these models to analyze and develop changes and modifications to the information processes of the organization. Some of the models provide definitions of factual items such as business functions, objects and data entities; others show how things flow, connect or relate to one another. Some of the models evolve and expand to match reality and others are done as snapshots, showing as-is and then as-proposed operations. The views are composed graphically using symbolic objects, line connectors and some rules of logic and structure. The objects are given names called labels that populate the data repository with entries that can be retrieved, expanded, detailed and used to define and document the contents of the project. There are logic rules for many parts of the models. The models can be tested and evaluated for completeness, consistency, rule compliance and other factors. All of the models and the repository are interrelated, and many share common components such as databases, objects and/or actions. The development of the models is iterative, often requiring several sessions before the models are complete and realistic. The ability to move from one model to another and to work on different ones at different times is critical to a successful MDA tool.

The rules of MDA deal with the checking of consistency and logical structures such as naming and complete linkages. Errors found in models are reported during the Visible Analyst *analyze* process. These errors should be corrected to maintain consistency and accuracy of the models. However, Visible Analyst, unlike software compilers, allows you to continue with any reasonable MDA operation without waiting until you have corrected all errors. This allows you to continue progress on the project and its components. However, it also leaves you responsible for returning and correcting your errors.

The Basic MDA Models

The basic MDA models include:

Functional Decomposition Model (also known as a Business Model) - Shows the business functions and the processes they support drawn in a hierarchical structure.

Entity Relationship Model (also known as a Data Model) - Shows the data entities of the application and the relationships between the entities. The entities are things and the relationships are actions. The data attributes can be defined for the entities via the repository and then shown on the diagram. Entities and relationships can be selected in subsets to produce views of the data model.

Object Model (also known as an Object Class Model) - Shows classes of objects, subclasses, aggregations and inheritance. Defines structures and packaging of data for an application.

State Transition Model (also known as the Real Time Model) - Shows how objects transition to and from various states or conditions and the events or triggers that cause them to change between the different states.

Process Model (also known as the Data Flow Diagram) - Shows how things occur in the organization via a sequence of processes, actions, stores, inputs and outputs. Processes are decomposed into more detail, producing a layered hierarchical structure.

Product Model (also known as a Structure Chart) - Shows a hierarchical, top-down design map of how the application will be programmed, built, integrated and tested.

Use Case Model – Shows the relationship between a user and a computer system.

Activity Model – Is a special form of state diagram where states represent the performance of actions or sub-activities. Transitions are triggered by the completion of the actions or sub-activities.

Sequence Model – Shows how objects collaborate in some behavior.

Collaboration Model – Shows an interaction organized around the objects in the interaction and their links to each other.

Business Process Modeling Notation- Provides a modeling notation that can be communicated to and understood by all business users, from the business analysts developing the models, to the technical analysts implementing the model processes, to the business people who manage and monitor the processes.

Repository or Library Model (also known as the Project Database) - Keeps the records of all recorded objects and relationships from the diagrams and allows for the definition of detailed specifics and extensions of the individual items. Used for evaluation, reporting and generation of details about the project and its products.

Visible Analyst Choices

Today systems designers have multiple choices. They can follow the Structured Analysis and Structured Design (SA/SD) approach and build on functions/processes, data models and product concepts; or they can follow the object-oriented approach and build class hierarchies, dynamic states and functional/process models. Both approaches can build better information systems and both cover similar aspects of information systems definition. However, both use different sequences of effort and focus on different aspects of the project. Visible Analyst allows you to choose either approach or to combine the approaches to develop a comprehensive product definition, design and development mechanism.

There are five keys to using Visible Analyst, or any MDA tool. The first key is to develop the discipline to apply and follow the steps and procedures of the technique. The second key is to develop skills in conceptualizing the MDA models to represent the real world requirements. The third key is to be consistent in how you define and describe the real world. The fourth key is to strive to be complete in the definition of all of the major parts of a real world application. The fifth key is to progress from the conceptual to the operational specifications and construction of a working information systems process.

VISIBLE ANALYST OVERVIEW

Visible Analyst is a Microsoft® Windows® application. Versions 7.1 and higher of Visible Analyst work with Windows NT, 2000, 2003 Server and XP while VA2008 is also VISTA compatible. This section defines the overall structure of Visible Analyst and identifies some of its key operational characteristics.

Visible Analyst Architecture

The basic components of Visible Analyst are: a set of diagramming tools, a rules module, and a repository module. Diagramming tools are used to construct the ―blueprints of your target system. These lessons guide you in the creation of diagrams and provide you with basic information on the uses of the diagrams.

A system is designed and constructed according to rules, and the rules module manages the methodologies of Visible Analyst tools for you. Visible Analyst allows you to choose the rule set you prefer to use as a guideline for the development of your system. These rules are important in determining the appearance of your diagrams, as well as the entire structure of your system. For the purposes of the tutorial, you are introduced to the supported techniques and learn how to designate the rule set to use and the different symbol types used for each rules methodology.

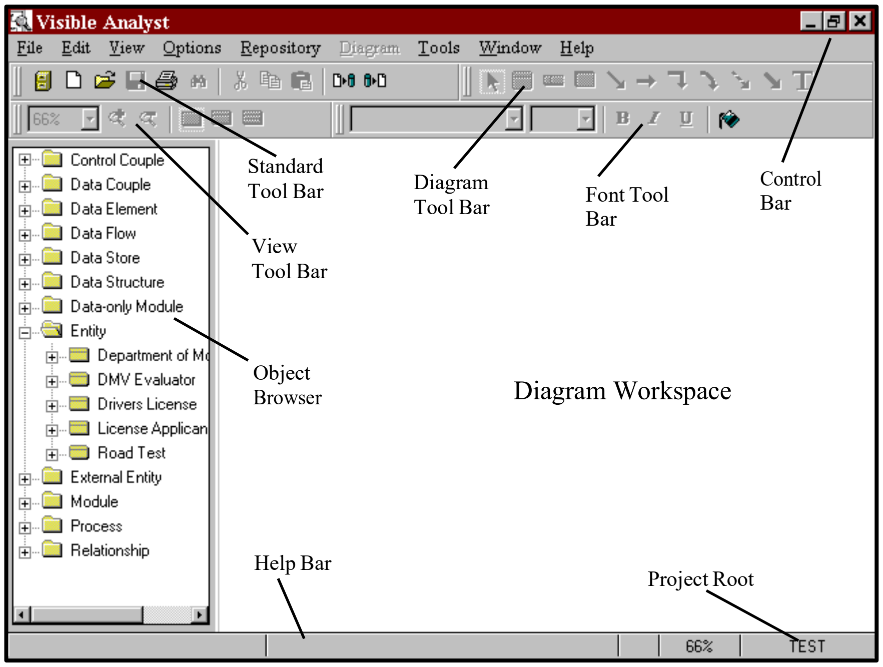


Figure 1-1 Visible Analyst Workspace

The repository module controls the individual repositories of each of your projects. A project's repository stores detailed information about objects that are used in developing a system. An object in the repository includes processes, entities, relationship lines, classes, etc. The type of information contained in the repository for each object includes description, composition, values and meanings, location references, and other very specific detail information (see Lesson 17 – Working with The Repository Functions for details). The repository makes Visible Analyst a very powerful systems development tool. Visible Analyst is much more than just a diagramming tool; its repository and rules set provide definition, documentation, and consistency capabilities for the entire system. Visible Analyst has advanced features enabling you to generate reports and code for your target system, using the information contained in a project repository.

Windows Version Features

This section highlights some of the Windows-specific features of Visible Analyst.

The Application Workspace

All work in Visible Analyst is done either in the main application workspace, shown in Figure 1-1, or in the repository, described in Lesson 16 - Repository Functions.

Windows Configuration

Visible Analyst configuration features controlled through Windows include the hardware configurations, desktop colors, available printer drivers, and available fonts. Changes or additions to these features can be made through Windows and are reflected in Visible Analyst.

Multiple Document Interface

The Windows Multiple Document Interface (MDI) allows multiple diagrams to be open at one time. Open diagrams can be of the same or different diagram types (data flow diagrams, entity relationship diagrams, etc.). Diagrams may be maximized, taking up the entire workspace, sized so that several diagram windows are visible, or minimized to icons appearing at the bottom of the application workspace. Any window larger than an icon is editable. You can cut, copy, and paste to and from the Windows Clipboard to move objects between diagrams and even between other Clipboard-aware applications. (See Figure 1-2.)

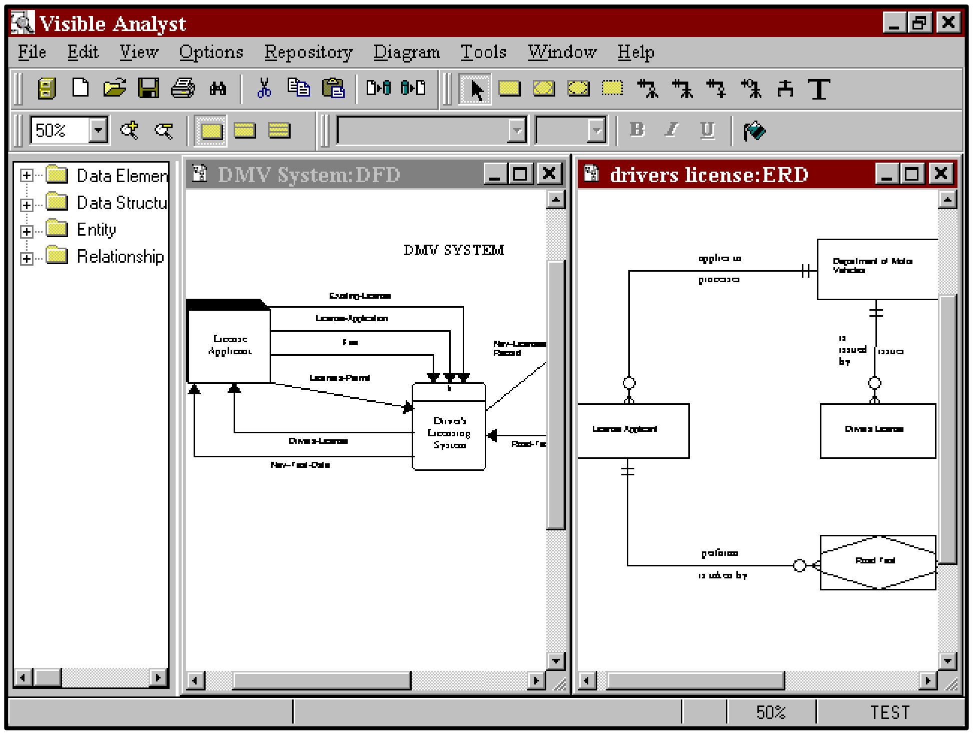


Figure 1-2 Visible Analyst Multiple Document Interface

Note

🗎 Users not familiar with MDI Windows programs should take note: there is a difference between the *diagram* Control menu button and the *Visible Analyst* Control menu button. The former is in the top left corner of the *diagram* window, or to the left of the File menu if the diagram is maximized. This Control menu contains functions that affect the diagram only, such Maximize, Close, etc. The latter is in the top left corner of the *Visible Analyst* window. The *Visible Analyst* Control menu affects the whole Visible Analyst window and program.

Selecting a Diagram Object

A diagram object is anything that appears on a diagram: symbol, line, text, or block. When you click on an object with a mouse button, it becomes the *current* or *selected object* and you can perform various operations on it. There are five different ways to select an object. The following paragraphs describe the effect of selecting an object with the left mouse button, the right mouse button, a double-click with the left mouse button, the TAB key and selecting a Block.

Left Mouse Button

Clicking on an object with the *left* mouse button selects it. The object changes color to show that it has been selected allowing you to make changes to the object or to move the object. When a symbol or line is selected, text labels for that object are automatically highlighted.

Right Mouse Button

Clicking on an object with the *right* mouse button also selects it. In addition, the Object menu appears containing all of the functions that can be performed on that object.

Notes

🗎 Unless stated to the contrary, instructions to *click* a mouse button refer to the *left* button. Instructions for the *right* button are explicitly mentioned.

🗎 Left-handed mouse users: if you use a mouse with the buttons reversed, you should reverse references to left and right mouse buttons in this text.

Double-Click

If you double-click on an object with the *left* mouse button, the repository entry for that object appears. If the object is unlabeled, a dialog box for labeling the object is displayed. Double-clicking is also used to indicate the end of a line.

TAB Key

To highlight only the text label for a selected symbol or line, press the TAB key until the appropriate item is highlighted. (If the label is located outside the symbol, you can click on it directly.) Continuing to press the TAB key sequentially selects each object on the diagram.

Selecting a Block

To select a block, meaning a group of objects, on a diagram, click and hold the left mouse button and drag the mouse to draw a box around the objects. All objects *completely* contained within that box change colors to show that they are selected. Once a block is selected, you can perform various functions on the block such as cut, paste, move, change text settings for contained objects, and other actions.

Deselecting Objects

To deselect any object or block, simply click the *left* mouse button on an empty area anywhere on the diagram workspace outside of the object or block. The items that had been selected return to their usual color. You can also use the Clear function on the Edit menu.

Shortcut Keys

Shortcut keys provide fast access to functions without using the menus. Some of the active shortcut keys used in Visible Analyst are standard Windows shortcut control key sequences, such as CTRL+P, which is the command for Print; others are specific to Visible Analyst. All available shortcut keys are listed here.

CTRL+A	Analyze	Analyzes a diagram or entire project.
CTRL+C	Copy	Copies to clipboard.
CTRL+D	Define	Accesses the repository.
CTRL+E	Connect	Draws lines between selected symbols.
CTRL+F	Find	Accesses the search mode.
CTRL+L	Lines	Sets the cursor to line drawing mode.
CTRL+N	New Diagram	Creates a new diagram.
CTRL+O	Open Diagram	Opens an existing diagram.
CTRL+P	Print	Prints the current diagram or queue contents.
CTRL+Q	Report Query	Generates a custom repository report.
CTRL+R	Reports	Generates a standard repository report.
CTRL+S	Save	Saves the current diagram.
CTRL+T	Text	Sets the cursor to text adding mode.
CTRL+U	Clear	Deselects diagram object or block.
CTRL+V	Paste	Pastes from Clipboard.
CTRL+T	Snap Symbols	Aligns selected symbols in a row.
CTRL+X	Cut	Cuts to Clipboard.
CTRL+Z	Undo	Erases partially drawn or undoes moved line.
DEL	Delete	Deletes object from diagram.
F1	Help	Displays context-sensitive help.
ALT+R	Delete Project	Deletes a project with no project files.

SHIFT+F1	Menu Help	Enters Help mode for menu items.
SHIFT+F10	Object Menu	Displays repository object menu.

Another standard Windows shortcut method for accessing a menu item without using the mouse is to press the ALT key followed by the underlined letter of the menu title or menu item that you would like to access. For example, to access the File menu, press the ALT key followed by the F key. It is not necessary to hold down the ALT key while pressing the F key.

Control Bar

The control bar, shown in Figure 1- 3, is located above the diagram workspace and gives you quick access to commonly used functions and types of objects that can be added to a diagram. The control bar can contain up to four tool bars.

- The standard tool bar contains basic buttons, such as Select Project, Open Diagram, etc., common to most Windows applications.
- The diagram tools tool bar contains the symbol, line, and text buttons appropriate for the current diagram.
- The view tool bar contains controls that change the zoom level and entity/class view level.
- The font tool bar contains controls that allow changing the current font characteristics, such as font type, font size, etc.

You can customize the control bar by selecting Control Bar from the Options menu to display the Customize Control Bar dialog box. Using this dialog box, you can select the tool bars to be displayed and select control bar options such as Show Tooltips, Large Buttons, Flat Buttons, and Hot Buttons. You can also right-click the control bar itself to display a properties menu that allows you to toggle the individual tool bars on or off or to select the Customize option. To change the size and position of the tool bars, click the left mouse button on the ―gripper (the two vertical bars ‖ at the beginning of each tool bar) and drag the tool bar to the desired position. From the Customize Control Bar dialog box, you can also ―undock ‖ the diagram tools tool bar so that it appears in its own floating window.

The ⇖ button (shown in Figure 1-3) is used to change into selection mode (also called editing mode). In selection mode, objects can be selected on the diagram to be changed or moved, or a box can be drawn around many objects on a diagram, for moving, cutting and pasting, or changing text settings for groups of objects. Click one of the drawing mode buttons, and you can add that type of item to the diagram. The object types include symbols, lines, couples, and caption text. When you choose one of the drawing mode items from the control bar to add to your diagram, the cursor automatically changes to indicate that you are either in symbol, line or couple adding mode, or caption text adding mode. Specifically, this means that while the cursor is positioned inside the diagram workspace and it is something other than an arrow, which indicates selection mode, clicking on the mouse adds an object to the diagram.

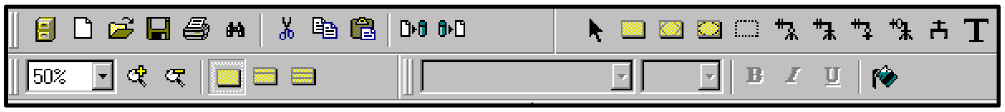


Figure 1-3 The Control Bar for Entity Relationship Diagrams with All Tool Bars Displayed

For example, when the diagram tools tool bar is displayed on the control bar, you can easily select the particular symbol you want to add to the diagram. A symbol is added to your diagram centered at the cursor location anytime you click on the diagram workspace while the cursor indicates symbol drawing mode.

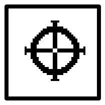


Figure 1-4: The Symbol Cursor

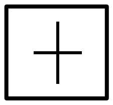


Figure 1-5: The Line Cursor

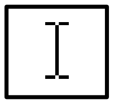


Figure 1-6: The Text Cursor

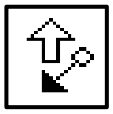


Figure 1-7: The Couple Cursor

Help Bar

As you move through the Visible Analyst menus, a line of text appears on the help bar at the bottom of the application workspace that briefly explains what that menu item does. The current zoom level, current project and current object are also displayed. You can toggle this feature off and on from the Options menu.

Object Browser

From the Options menu, you can choose to have the Visible Analyst object browser displayed on your screen. The object browser displays a list of all the objects in the repository in a resizable window. When there are no diagrams open, or the current window is the diagram list, all objects are displayed. When a diagram is open, only those objects that are valid for that diagram type are displayed. If an object appears on the open diagram, it is displayed in bold. Double-click on a folder in the list to expand or collapse it; double-click on an object in the list to display the Define dialog box. You can also click on an object in the list and drag it onto your diagram. To resize the object browser, click on the right margin of the browser and drag to the desired size.

Menus

The menus are arranged in nine groups for browsing and selecting the various features of Visible Analyst. (Refer to Figure 1-1.)

File Menu

The File menu contains the functions for accessing and creating projects and diagrams. This includes all of the functions that cause the opening of another diagram, such as Nest, Spawn, and Page. (These functions are explained under the specific diagram type where each is used.) It also includes a list of Recent Diagrams and Recent Projects. The Save, Print, and Exit functions are also found in the File menu. If you are using a network version, information about network activity and modifying the user list is contained in the File menu. If you purchased a copy of the Zachman Framework Edition, the framework can be opened and closed using the Zachman Framework option.

Edit Menu

The Edit menu contains the standard Windows editing functions including Cut, Copy, Paste, Find and Delete. There is also an Undo function for removing partially drawn lines and undoing a move line operation. The Strategic Planning options allow you to add a New Statement, Promote, Demote, Move Up, or Move Down, a strategic planning statement.

View Menu

The functions contained in the View menu allow you to change the appearance of the active diagram. There are functions to change the zoom level and to give you the ability to change the items displayed on a diagram, including Show Line Names, Show Symbol Names, Show Discriminators, Show Statement Types, Show Priority, Show Description, Class and Entity Display Options, Physical Schema, Events, and Messages. Also on the View menu are Grid and Ruler, functions that make it easier to position objects accurately on a diagram.

Options Menu

The Options menu contains functions that allow you to change default settings for Visible Analyst. For diagram drawing and manipulation settings, the functions include automatic labeling of symbols and lines, Line Settings defaults, Text Settings defaults and diagram Colors, as well as on/off settings for Security, the Help Bar, the Object Browser, and the Control Bar. The Options menu also includes settings for interaction diagrams, model balancing rules, SQL schema and shell code generation, DDS name translation, user-defined attribute and object definition, planning statement types, Zachman Framework cell settings and symbol template settings.

Repository Menu

All of the selections included in the Repository menu are functions performed on the information contained in a project's repository. These include Define, which allows repository access, schema and shell code generation, schema / model comparison, Key Analysis and Key Synchronization, Model Balancing, and repository Reports. The Divisions function is used with the Enterprise Copy feature and is explained in the on-line Help. The Divisions and Enterprise Copy feature are not available in the Visible Analyst Student edition.

Diagram Menu

The Diagram menu contains functions for selecting, manipulating, and analyzing diagram objects. These include functions for selecting Symbols, Lines, or Text to add to a diagram, as well as functions for changing or stylizing a selected item on a diagram. The function for analyzing the diagrams according to the selected rules methodology, modifying the diagram settings and the function for modifying an existing view are also contained in the Diagram menu.

Tools Menu

The Tools menu contains the various functions that can be performed on a project. These include Backup, Restore, Copy Project, Delete Project, Rename/Move, Import, Export, and copying information between projects. The utility for assigning user access to the multi-user version of Visible Analyst is also found in the Tools menu. The Enterprise Copy and Enterprise Tag Maintenance features are not available in the Visible Analyst Student Edition but are explained in the on-line Help.

Window Menu

The Window menu allows you to change the arrangement of the open diagrams. Diagrams can be automatically arranged in a Tile, Cascade, or minimized (icon) format. You can also switch between open and minimized diagrams.

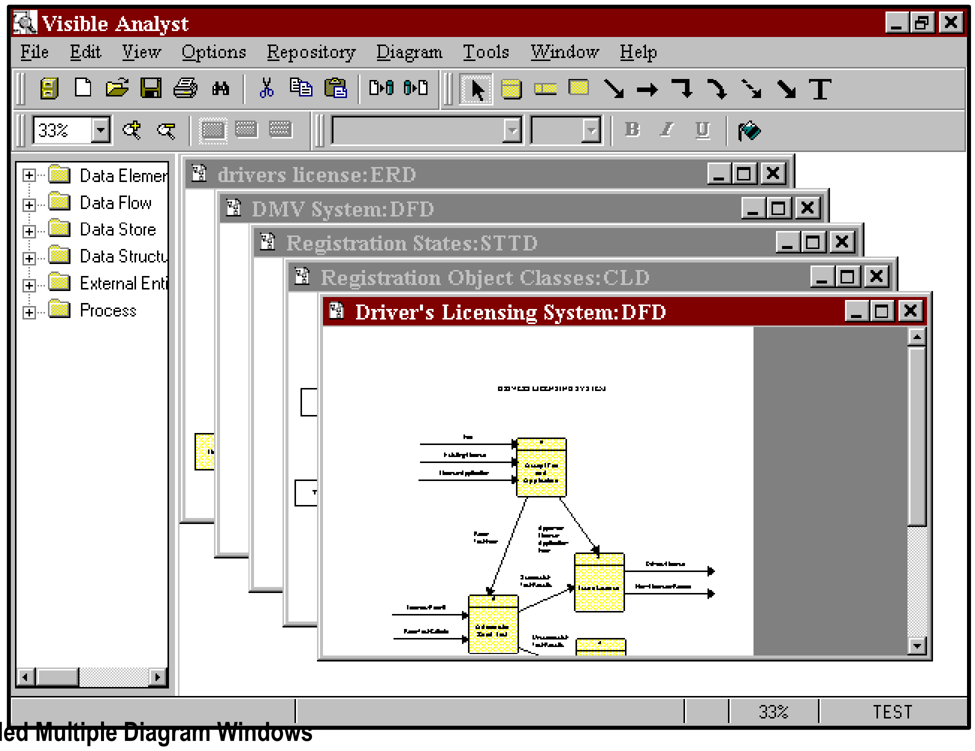


Figure 1-8 Cascaded Multiple Diagram Windows

Help Menu

The Help menu allows you to access the Help features, product and user information, and Visible Analyst on the Internet.

Note

🗎 Detailed information about each of the menu options can be found in the Visible

11-1 Completed State Transition Diagram

Activity Diagramming

OVERVIEW

The activity diagram describes the sequencing of activities, with support for both conditional and parallel behavior. An activity diagram is a special form of a state diagram in which the states represent the performance of actions and the transitions are triggered by the completion of the actions. The activity diagram can be attached to a class or to the implementation of an operation or a Use Case. The purpose of an activity diagram is to focus on flows driven by internal processing (as opposed to external events). Usually activity diagrams are constructed in situations where all or most of the events represent the completion of internally generated actions.

DEFINITIONS

The main components of an activity diagram include:

Activity	An activity is a state of doing something. It could be a task such as receiving a payment, or the execution of a software routine, such as a method on a class. It is represented by a rectangle with rounded corners.
Decision	A decision is used when more than one activity can be performed next, based on a certain condition. There is a single incoming transition and several guarded outgoing transactions. The guards are mutually exclusive and so only one of the outgoing transactions is followed. A diamond denotes the decision start and end.
Synchronization Bar	A synchronization bar is used to show parallel activities. It is represented by a black bar with one or more input transitions and one or more output transactions, that are all taken in parallel. This means that the sequence of the output transactions is irrelevant. In order to show that all the parallel activities need to be completed before the following activities, use a second synchronization bar that has multiple incoming transactions and a single outgoing transaction. The outgoing transaction is taken only when all the incoming transactions are completed.

Start	The start object designates the starting point of the activity diagram and is represented by a filled circle.
End	The ending point of the activity diagram is represented by a filled circle inside a hollow circle.
Swimlane	A swimlane is a way of designating responsibility for each action state. An activity diagram may be divided visually into "swimlanes"; each separated from neighboring swimlanes by vertical solid lines on both sides. Each action is assigned to one swimlane.
Transition	Represented by a solid line with a stick arrowhead, transitions may cross swim lanes. Transitions are implicitly triggered by the completion of the preceding them. The transitions may include guard conditions and actions. It is labeled by a transition string of the form _Event [guard]/Action'. All components of the transition string are optional

RELATIONSHIPS

The relationship structure in an activity diagram is directional arrows showing how the order in which the activities occur. The completion of one activity triggers the flow to move on to the next activity as dictated by the arrows.

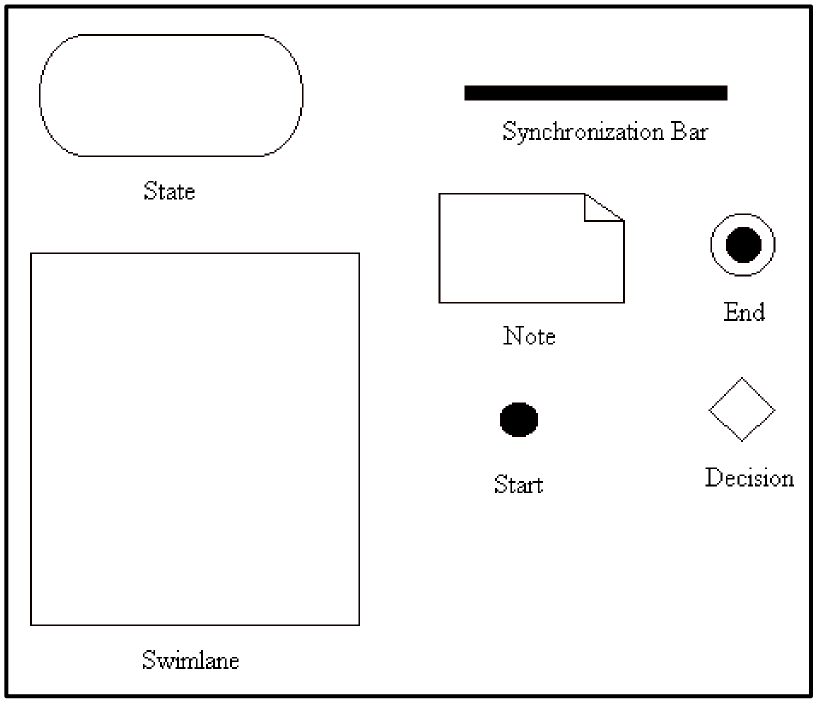


Figure 12-1 Activity Methodology Symbols

DEVELOPING YOUR ACTIVITY DIAGRAM

An activity diagram is a variety of activity states arranged in the sequence in which they must be performed. For our tutorial example, we look at the process of getting a driver‘s license.

Designating the Starting Point

For every activity diagram there has to be a designated starting point shown on the diagram with a filled circle.

Set the Zoom Level:	1	From the View menu, select 66% zoom so that you can see all of the needed workspace.

Create a New Diagram:	2	From the File menu, select New Diagram.
	3	Select the diagram type Activity.
	4	Select Standard Workspace.
	5	Click OK.
Add Start:	6	Click the fourth symbol button, the filled circle, on the control bar. This is the start object.
	7	Place the cursor in the top center of the workspace, and click the left mouse button. The starting point is drawn on the diagram.
Save the Diagram:	8	Save the activity diagram with the label —Driver‘s License Activity Diagram‖.

Adding A Synchronization Bar

A synchronization bar is helpful in depicting activities that are performed in parallel. For instance in our example, the receiving clerk can receive the application form and proof of insurance simultaneously. It is not important which order they are received in. Furthermore, the next activity of validating the applicant can be performed only after both application and insurance proof have been received. The synchronization bar denotes the starting and ending of activities performed in parallel.

Add Synchronization Bar:	1	Click the third symbol button, the bar, on the control bar. This is the synchronization bar that denotes forks and joins.
	2	Place the cursor under the start circle and click the left mouse button. A synchronization bar is drawn.
	3	Add the other bar as shown in the Figure 12-2.

Save the Diagram:	4	Save the activity diagram.

Adding Activities

Activities are the basic building blocks of the activity diagram. By determining what activities need to be performed, and arranging them in the order in which they are performed, with support for conditional and parallel behavior, the activity diagram is complete. Activities are represented as rectangles with the activity described inside the rectangle.

Add Activities:	1	Click the first symbol button, the rectangle, on the control bar. This is the activity (state) object.
	2	Place the cursor under the start circle and click the left mouse button. An activity is drawn.
	3	Label this activity —Receive Road Test Form/Learner‘s Permit‖.
	4	Add the rest of the activities as shown in the Figure 12-2.
Save the Diagram:	5	Save the activity diagram.

Adding Decisions To A View

In a process, some activities may occur only if a certain condition is met; otherwise certain other activities are carried out. The decision diamond marks the beginning and end of conditional behavior. In our example, only if an applicant is deemed valid does the testing procedure continues; otherwise the applicant is informed why his/her application was deemed invalid and the process ends there.

Add Decision:	1	Click the second symbol button, the diamond, on the control bar. This is the decision object.
	2	Place the cursor under the activity —Validate Applicant‖ and click the left

		mouse button. A decision symbol is drawn.
	3	Add the rest of the decisions as shown in the Figure 12-2.
Save the Diagram:	4	Save the activity diagram.

Adding Stopping To A View

The stopping point for a process is denoted with a filled circle inside a hollow circle.

Add Stop:	1	Click the fifth symbol button, the filled circle inside a hollow circle, on the control bar. This is the ending object.
	2	Place the cursor under the activity ―Validate Applicant‖ and click the left mouse button. An ending symbol is drawn.
	3	Add the rest of the decisions as shown in the Figure 12-2.
Save the Diagram:	4	Save the activity diagram

Adding Transitions To A View

Transition lines are arrows that communicate the order in which the activities need to be carried out. They can be labeled or left unlabeled.

Turn Off Auto Label Lines:	1	Select the Options menu and click Auto Label Lines, so that it is not checked anymore.
Add Transition:	2	Click the symbol button labeled _event', the horizontal arrow, on the control bar. This is the transition symbol.

	3	Place your cursor on the start object. Click the left mouse button and hold down as you drag the cursor down to the first synchronization bar.
	4	Add the rest of the transitions as shown in the Figure 12-2.
Save the Diagram:	5	Save the activity diagram.

Adding Labels to Transition Lines

Select the Transition to be labeled:	1	Click on the transition leading to the activity —Test Vehicle Knowledge‖.
Add Transition:	2	Click the right mouse button and choose Change Item.
	3	Type _valid' in the Event Name field and click OK.
	4	Add the rest of the labels as shown in the Figure 12-2.
Save the Diagram:	5	Save the activity diagram.

Adding Swimlanes To A View

Swimlanes depict responsibility. One can use swimlanes to depict which people or departments are responsible for which activities. In programming, this translates to assigning a class to each activity. In our example, we can identify two DMV departments that would be responsible for the activities in our diagram. The testing department would take care of performing the actual tests, and evaluating the test results; while DMV administration staff would perform the other duties such as accepting applications, validating applicants, issuing licenses to qualified applicants, etc.

Add Swimlane:	1	Click the sixth symbol button, the

rectangle, on the control bar. This is the swimlane object.

2 Name this swimlane ―DMV Administration‖.

3 Place the cursor above the start object and click the left mouse button. A swimlane symbol is drawn.

4 Click the arrow to activate select mode.

5 Select the swimlane you just drew and expand its size by a clicking and dragging at its ends. Make sure the activities that are DMV Administration‗s responsibility fall in this swimlane as shown in Figure 12-2.

6 Add the other swimlane as shown in the Figure10-2.

Save the Diagram: 7 Save the activity diagram.

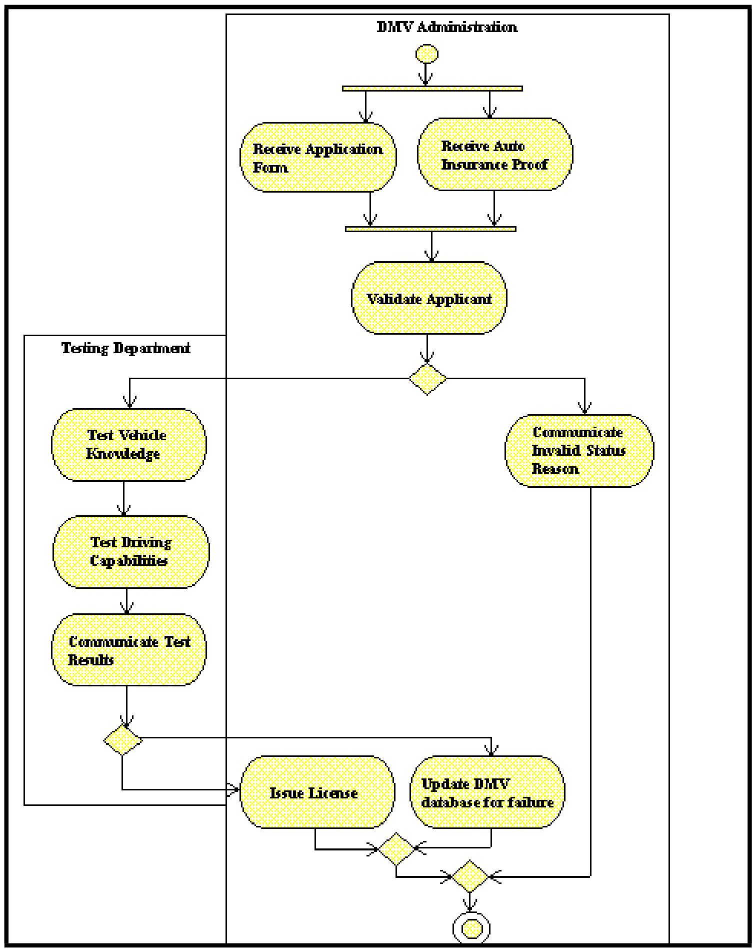


Figure 12-2 Activity Diagram

Working with the Repository Functions

OVERVIEW

This unit helps familiarize you with the operation of the Visible Analyst repository and shows you the power of an online, interactive database for systems analysis, design and data modeling. The TEST project used in the previous lessons is used as the basis for your exercises.

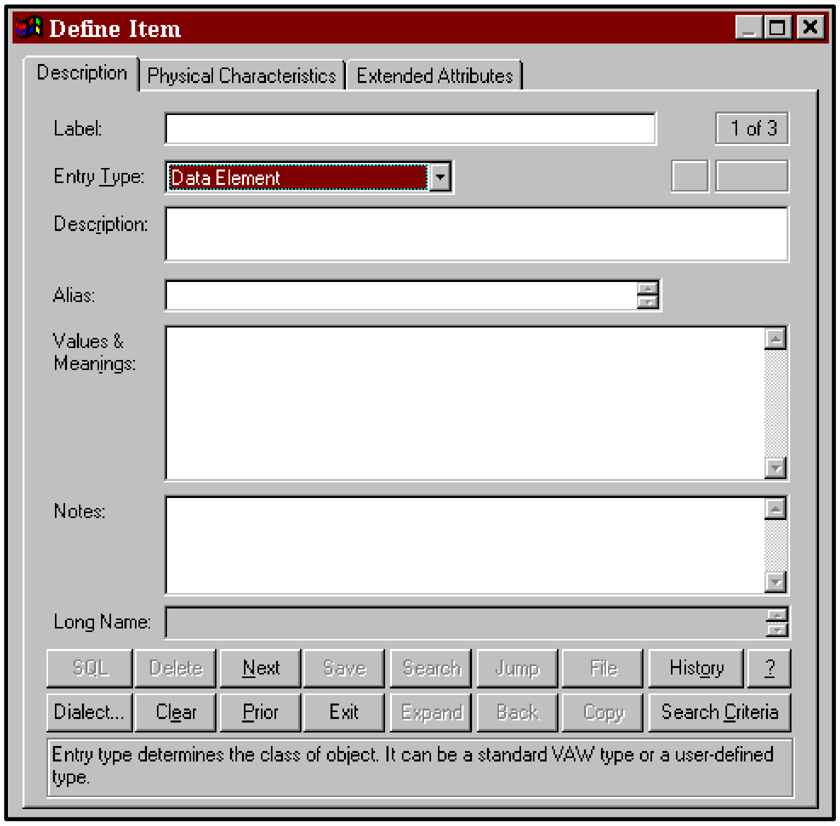
The repository is a powerful tool for creating and managing the narrative portions of a system‘s specification. A project repository is used to provide an entry location for all project documentation. Each graphical entry on your diagrams has an automatically created corresponding entry in the project repository, as do any items entered into a Composition or Alias field.

You have the ability to thoroughly define all of your graphical entries in the repository or to simply enter notes about them in the Notes field. As an integrated part of Visible Analyst, the repository operates in parallel with the diagramming functions to accomplish data decomposition logically. It contains powerful data management, text editing, import/export, and report facilities. By using it, meaning can be ascribed to diagrams and an asset of ever-increasing value can be created. After defining items, changing entries and entering notes, you can generate reports from this information in many different forms.

When you finish defining your data and processing, the repository also allows you to put it into an ASCII file and export it. The ASCII file can then be sorted to move data specifications to your database and process specifications to your text editor for writing code. (The Shell Code Generation utility can also be used for this purpose.)

Note

🗎 Users of the Educational and Demonstration versions of Visible Analyst cannot add items directly into the repository. First add the object to the diagram, and then edit it into the repository.



Label:

Entry Type: Data Element

Description:

Alias:

Values & Meanings:

Notes:

Long Name:

Entry type determines the class of object. It can be a standard VAW type or a user-defined type.

Figure 17-1 Blank Repository Dialog Box, Page One

REPOSITORY BASICS

Repository Control Buttons

The repository control buttons (see Figure 17-2) are always displayed at the bottom of the repository dialog box. Each of the button functions is accessed by clicking on the button or, as is customary in Windows, by using its keyboard shortcut by holding down the ALT key and pressing the underlined letter to execute the button function. Only the functions available to you at a given time are active; the others are grayed. The button functions are:

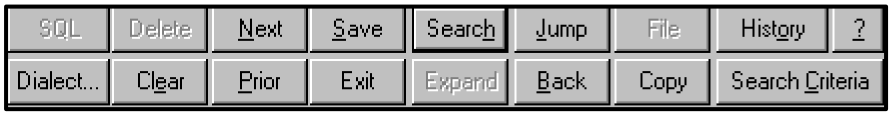


Figure 17-2 Repository Dialog Box Control Buttons

SQL	This button opens the Generated SQL for View dialog box. This dialog box displays the SQL generated for the current view object based on the view table and column specifications selected when creating the view, as well as the current SQL dialect. This button is active only when the entry type is View.
Dialect	This activates the RDBMS SQL Dialect dialog box. From there, you can change the current SQL dialect.
Delete	This deletes the current repository entry from the database. An entry can only be deleted when it has no location references, meaning that it does not appear on a diagram nor as an attribute of another repository item.
Clear	This clears the display of an entry and displays a blank repository dialog box. This allows you to Search for an existing entry or add a new entry. If you have made changes, you are prompted to save them before clearing. Your current location in the repository remains unchanged.
Next	This displays the next sequential repository entry that meets the repository search criteria (see below).
Prior	This displays the previous sequential repository entry that meets the repository search criteria.
Save	This saves all changes made to an entry.
Exit	This or the ESC key exits from the repository.
Search	This initiates a search for a particular entry in the repository. The procedure is explained in the section on Search Capabilities.
Expand *Contract*	This allows you to expand or contract the display size of some fields. The fields that normally display four lines can expand to display 15.

Jump	This allows you to jump immediately to another entry that is referred to in the current one. This feature is described in the section on Navigation Capabilities.
Back	This button provides a means of jumping to the previous repository entry. You can then continue to move backwards displaying previous repository entries.
File	This allows you to insert text from a DOS file at the cursor position or to copy highlighted text to a DOS file. It is explained in further detail in the Visible Analyst *Operation Manual* and in the online help system.
Copy	This button provides a means of copying the current object.
History	This provides a means of jumping back to a previously displayed repository object. A list is kept of every object definition that has been displayed. If you choose this button, the History dialog box appears and you can jump between entries by double-clicking on an entry. The maximum is 500 objects.
Help(?)	This displays context sensitive help about the repository. You can also press F1 to activate the help system.
Search Criteria	This allows you to specify how the repository is to be searched. It is explained in the section on Search Capabilities.

Other buttons that may be displayed on the Define dialog box are:

Primary Key	If the current object being examined is an entity type, the primary key button is displayed to the left of the Composition/Attributes field.
Attributes Details	This button provides a means of populating the composition of a repository entry with components and physical information. This button is displayed to the left of the Composition/Attributes field.
	When the Entry Type is a Class, or when the Classic User Interface is turned off, the Add button displayed beneath the Attributes Details button is active. You can use this button to add details. When you begin typing in the field next to the Add button, the

button is enabled. Click the Add button to add the attributes to the Attributes field.

Editing Keys

Because the Edit menu is not accessible from the repository, you can use the right-click menu that is available when an object (text) is highlighted and you click the right mouse button. Using the right-click menu, you can Cut, Copy, Paste, or Delete the selected object.

Field Types

The data repository of a Visible Analyst project is displayed using Define dialog box variations corresponding to different diagram objects. You see and work with some of these variations during the course of this lesson. The basic dialog box, shown in Figure 17-1, is for data elements, aliases, miscellaneous objects and external entities or source/sinks. Other objects, such as data stores, processes, functions, entities, relationships, modules, data flows information clusters, etc., have variations in individual fields and tabs of the Repository dialog box to accommodate the specific needs of those items. Some of these differences are seen later in the lesson.

Label Field

This is the name of the repository item. The names of items drawn on diagrams are automatically entered here.

Entry Type Field

This tells Visible Analyst what kind of object the item is: process, data flow, entity, etc. The entry type can be entered manually, or you can select the type from the scroll box accessed by clicking the down arrow at the end of the entry type field.

Note

🗎 You can edit the Entry Type and Label fields of data elements and data structures that do not appear on diagrams. The entry type for a data element cannot be changed if physical information for that element has been entered.

Description Field

The Description field is a two-line field that provides a convenient place to enter a somewhat more extensive descriptive title of the object than the Label field allows. The contents of this field are used for the Comment on Column (data elements) and Comment on Table (entities) when SQL DDL is generated if the selected SQL dialect supports this syntax.

Alias Field

The Alias field contains 10 lines of 128 characters each. It allows for the entry of alternative labels to the one used as the object label. This is most commonly used for indicating the cryptic abbreviations that are sometimes used in the actual coding of a software program, as opposed to the plain English names that are desirable for reference. The Alias field is an intelligent field. Data names entered into it establish new repository entries for these aliases.

Attributes Field

The purpose of the Attributes field is to accumulate the collection of data elements that you wish to define as constituting a data flow, entity, data store, etc. The Attributes field is an intelligent field. Data names entered into it establish new data element repository entries or update existing ones. These new data elements can then be used for further definition. Data flows, data structures and couples can also appear in some Attributes fields.

When you click the Attributes Details button, the Add Attributes dialog box appears. Using this dialog box, you can define up to 12 components and some of their properties. As you enter items, the dialog box automatically scrolls as necessary to allow you to enter more items until you reach 12. When you complete the entries, click OK to add them to the Attributes field. If you need to add more than 12 components, click the Attributes Details button again; and a new dialog box opens so that you can add additional attributes.

Use the Add button at the bottom of the Attributes field to add components one at a time. When you begin typing in the field next to it, the Add button becomes active. Complete your entry, and then click Add to enter the component in the Attributes field.

Values & Meanings Field

The Values & Meanings field allows an unlimited number of lines. The maximum number of characters that can be contained in the field is 64K. This field allows the entry of specific information about the value(s) the item can take.

Discriminator Values & Meanings Field

If the current object is a data element that is used as a discriminator, this field contains a list of values to identify the subtype entities. For each subtype, a value can be entered that will uniquely identify it. By default, these values are numbers starting with 0 for the supertype. To change the value, click the value until an edit control appears, make your changes, then press ENTER.

Notes Field

The Notes field is also a field that allows you to enter any pertinent information about the object. The maximum number of characters that can be contained in the field is 64K. This is the logical field to use when creating hyperlinks to external documents, web pages or other application files.

Location Field

This field displays two types of usage information. The field can contain the diagram name (and, for DFDs, the diagram number) of every diagram where the item appears. The field can also tell you if the item appears as an attribute of another item. This second kind of location entry has the entry type of the parent item, followed by an arrow and the name of the parent item.

Other Pages and Fields

Other pages of the Define dialog box contain additional information. For example, pages 2 and 3 of the basic repository form provide location and relationship information and specifications for PowerBuilder/VISION extended attributes. These two pages are similar for most entry types. For some entry types, additional pages will be displayed:

- When the entry type is an entity, the next five pages contain keys, foreign keys, triggers, check constraints, and physical information.
- For views, the next five pages provide table, column, join, clause, and option information.
- When the entry type is a relationship, there are additional pages that contain foreign key and cardinality information.
- When the entry type is a tablespace, an additional page contains property information.

A full list and complete descriptions of pages and fields can be found in the *Operation Manual* and in the online help.

Object Repository

The Visible Analyst repository provides several additional forms and data input components for supporting the object-oriented concepts. The object repository components are detailed below.

Attributes

The Attributes field replaces the Values & Meanings field whenever the Repository dialog box displays a class. The field contains a list of the data members for the class showing the local data element and type. To add, change, or remove local data elements, click the Attributes Details button or select Add/Change from the Repository Object menu. For each attribute, the following information can be defined:

- **Name**. The name of the attribute. Each attribute of a class has a separate entry in the repository with a type of local data element. This is an optional field. The search button can be used to find other local data elements in the repository.
- **Type**. The attribute type can be a class, data element, or data structure. If the type does not exist in the repository, a new class is created. The location field of the attribute type contains a reference to the current class. This is a mandatory field. The Search button can be used to display a list of valid types. If the attribute type is a data element or elemental

class, its physical characteristics are displayed. Entries added to the Type field are saved as data elements for an entity or data flow, and class/subtype element when the object is a class.

- **Limit**. The number of occurrences of the attribute. If this field is blank, the attribute occurs once.
- **Reference**. A qualifier to indicate the access method for an attribute. *Value* indicates the object defined in the *Type* field is used; *Address* indicates a pointer to the object is to be used; and *Reference* indicates a reference to the object is to be used. The default is Value.
- **Visibility.** *Publi*c members have global visibility. *Private* members are only accessible to member functions and friends. *Protected* members are accessible to derived classes and friends. *Implementation* members are only accessible to the class itself. The default is Private.
- **Qualification**. *Constant* indicates a member's value cannot be changed. *Volatile* indicates the member can be modified by something other than the program, either the operating system or hardware. *Static* indicates there is only one instance of the member regardless of the number of times a class is instantiated. The default is None.
- **Physical Characteristics**. If the attribute type is elemental, the physical characteristics can be set.

For every item entered into the Type field, Visible Analyst creates a repository entry (if one with the same name does not already exist) and updates that entry's location field. If an item is removed, this field is updated to reflect this. These repository entries are generally created as classes unless a data element already exists with the same name or the physical characteristics are defined

As you enter items, the dialog box automatically scrolls as necessary to allow you to enter more items until you have finished. Insert is used to insert a new attribute into the list at the current position, while Delete removes the current attribute (the current position is indicated by ➢➢). When you have completed the entries, click OK to add them to the Attributes field.

Item names entered into this field may contain up to 128 characters each and may consist of any upper or lower case letters, numbers, spaces, periods, underscore characters and hyphens; but the first character must always be a letter.

Attached Entities/Classes

The attached entities/classes for the currently displayed relationship are listed in this field. When an inheritance relationship is displayed, the characteristics of that relationship can be changed (see changing Inheritance Characteristics later in this chapter). Otherwise, the information cannot be edited from within the repository; and all changes must be made on a diagram. The field lists the two entities or classes attached to this relationship. Below the second entity name is listed the reverse of the current relationship. If either direction of the relationship has not been named, the name of the relationship in the reverse direction is displayed as ―reverse of (opposite relationship name).‖ This field allows you to jump to the repository entries for any of these entities or relationships, as described above.

Relations

For an entity or class, the Relations field displays the relationship name followed by the name of the entity or class on the other end of this relationship for each relationship attached to this entry. These sets are ordered alphabetically by the opposite entry name. When an inheritance relationship is displayed, the characteristics of that relationship can be changed (see Changing Inheritance Characteristics later in this chapter); otherwise, the information cannot be edited from within the repository; and all changes must be made on a diagram.

This field allows you to jump to the repository entries for any of these entities, classes, or relationships by positioning the cursor on the line containing an entity, class, or relationship name and clicking the Jump button.

Long Name

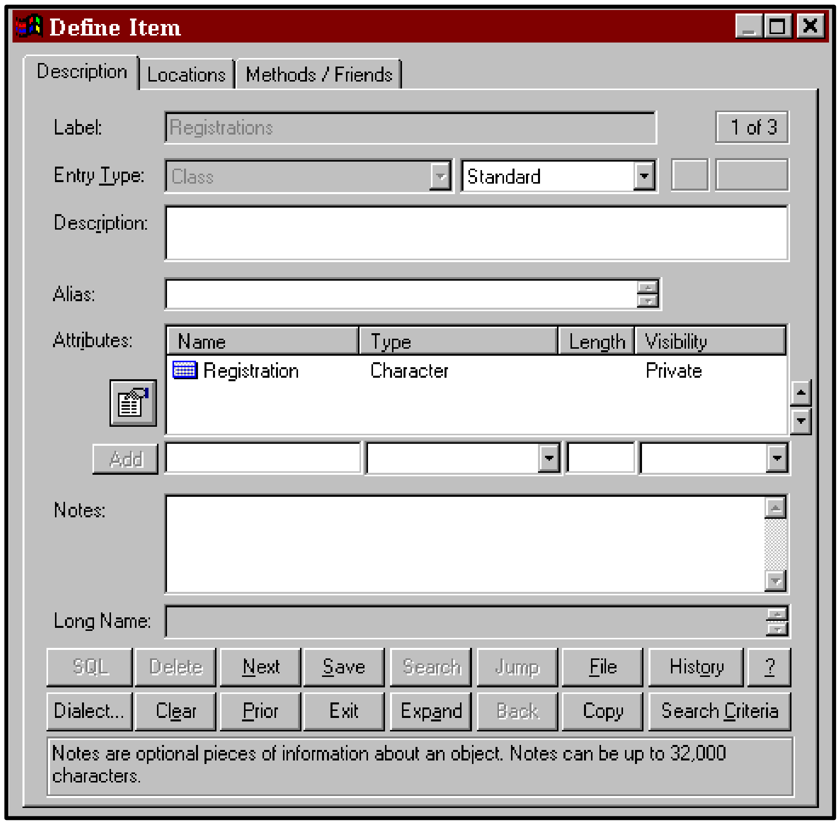
When a repository entry, either a local data element or a module, belongs to a class, the full name of the entry includes the class name. The Long Name field displays this name and, in the case of modules, includes the argument list (the argument list is required to differentiate overloaded member functions). If you want to change the argument list for a class method, click the right mouse button on the Long Name field and select Change (see the Methods section later in this chapter for details). If you want to change the class to which the method belongs, select Class from the Repository Object menu. To display the class definition, click the Jump button.

Class Characteristics

Concurrency, displayed on the Methods/Friends tab, is a class property that distinguishes an active object from inactive object. An *active* object may represent a separate thread of control. A *sequential* object is a passive object whose semantics are guaranteed only in the presence of a single thread of control. A *guarded* object is a passive object whose semantics are guaranteed in the presence of multiple threads of control.

A persistent class exists beyond the lifetime of an executable program. This means it must be stored on a non-transitory storage device. If the subtype of a class is set to either entity (associative or attributive) and the class is used on an entity relationship diagram, this field cannot be changed.

An abstract (or virtual) class cannot be instantiated because it contains pure virtual methods. If pure virtual methods exist for a class, Abstract is checked. If you attempt to uncheck this field, all pure virtual methods are reset to virtual. If you attempt to check it and virtual methods exist, they are converted to pure virtual methods.



Label: Registrations — 1 of 3

Entry Type: Class — Standard

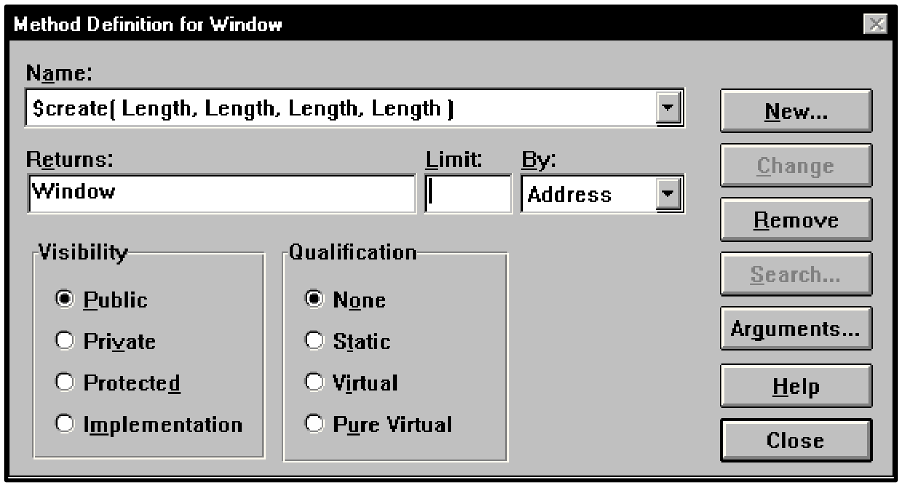
Name	Type	Length	Visibility
Registration	Character		Private

Notes are optional pieces of information about an object. Notes can be up to 32,000 characters.

Figure 17-3 Class Attributes

Methods

Methods (or Member Functions) are the operations that are defined for accessing a class. The Methods field contains a list of the functions for a class showing the name, return value, argument list, and flags to indicate its visibility. To add, change, or remove methods, click on the Methods field and click the Attributes Details button or select Add/Change from the Repository Object menu. To add a new method for a class, click the New button and type the name of method you wish to add. To search for methods that have already been defined in the repository, click the Search button. The list contains all modules that have previously been defined in the repository. If the module already belongs to a class, the class name is displayed. Note that when you select a module that already exists, the complete definition for that module is used including return value and argument list. Click OK to add the method name to the list of methods for the current class. For each method, the following information can be defined:



Name: $create(Length, Length, Length, Length)

Returns: Window

Limit:

By: Address

Visibility: [✓ Public] Private Protected Implementation

Qualification: [✓ None] Static Virtual Pure Virtual

Figure 17-4 Class Methods

- **Returns**. The return type can be a class or data element. If the type does not exist in the repository, a new class is created. The Location field of the attribute type contains a reference to the method. This is an optional field. Click the Search button to display a list of valid types.
- **Limit**. The number or size of the parameter. If this field is blank, it occurs once.
- **By.** A qualifier to indicate how the return value is passed. *Value* indicates a copy of the parameter is passed; *Address* indicates a pointer to the object is to be used; and *Reference* indicates a reference to an object is to be used.
- **Visibility**. *Public* methods have global visibility. *Private* methods are only accessible to other member functions within the same class and friends. *Protected* methods are accessible to derived classes and friends. *Implementation* methods are only accessible to the class itself. The default is Public.
- **Qualification**. *Static* indicates a method can be used without a specific instance of an object (it can only be used with static attributes (data members)). A *Virtual* method is one that you expect to be redefined in a derived class. A *Pure Virtual* method has no definition and must be defined in a derived class. A class with any pure virtual functions is an abstract (or virtual) class. The default is None.
- **Arguments**. A list of parameters to be used by the method. This is an optional field. If a method appears more than once with the same name within a class, it must have a different argument list for each definition. This is known as function overloading. See the next section for defining arguments.

When a method is added to a class definition, an entry of type module is created in the repository. The long name includes the class name and the argument list. The argument list is needed to differentiate between overloaded functions.

Note

🗎 Because the same name can be used for more than one method, there may be duplicate module entries in the repository, each belonging to a different class.

Arguments for Methods

When defining methods (member functions) for a class, the parameters to the function need to be specified. To add, change, or remove arguments, click the Arguments button on the Method Definition dialog box. For each argument, the following can be defined:

- **Name**. The name of the parameter. This is an optional field.
- **Type**. The parameter type can be a class or data element. If the type does not exist in the repository, a new class is created. This is a mandatory field. The Search button can be used to display a list of valid types. If the parameter type is a data element or elemental class, its physical characteristics are displayed.
- **Limit**. The number or size of the parameter. If this field is blank, it occurs once.
- **Pass By**. A qualifier to indicate the how the parameter is passed. *Value* indicates a copy of the parameter is passed; *Address* indicates a pointer to the object is to be used; and *Reference* indicates a reference to an object is to be used. The default is Value.
- **Qualification**. *Constant* indicates a parameter‘s value cannot be changed. *Volatile* indicates the parameter can be modified by something other than the program, either the operating system or hardware. The default is None.
- **Physical Characteristics**. If the parameter type is elemental, the physical characteristics can be set.

For every item entered into the Type field, Visible Analyst creates a repository entry (if one with the same name does not already exist). These repository entries are generally created as classes unless a data element already exists with the same name or the physical characteristics are defined.

As you enter items, the dialog box automatically scrolls as necessary to allow you to enter more items until you have finished. INSERT is used to insert a new parameter into the list at the current position, while the DELETE key removes the current parameter (the current position is indicated by ➢➢). When you have completed the entries, click OK to update the method name field. Item names entered into this field may contain up to 128 characters each and may consist of any upper or lower case letters, numbers, spaces, periods, underscore characters and hyphens; but the first character must always be a letter.

Friends

The Friends field displays a list of both friend classes and methods (or functions). A friend is allowed access to the private data members of a class. To add friends, click on the Friends field and click the Search button, select Add from the Repository Object menu, or double-click on the Friends field while pressing CTRL key. A list of classes and member functions is displayed in the Search list box. Locate each repository item you want to place in the Friends field and click the Search button; the item is added to the Select list box at the bottom. When you have found all of the entries you want, click the Select button and they are entered into the Friends field.

To remove a friend, highlight the desired item and press the DELETE key or select Cut or Delete from the Repository Object menu.

Navigation Capabilities

In this section, you change the displayed repository entry using Next, Prior, and Jump.

Note

🗎 The repository saves some internal settings for the duration of a Visible Analyst session. If these are set incorrectly, they may interfere with the smooth flow of this lesson. Therefore, we suggest that if you or another user worked in the repository during the current session, you should exit to Windows and restart Visible Analyst. In this way, you have a clean slate on which to run this lesson.

Open the Repository:	1	Access the repository using either Define from the Repository menu or CTRL+D. A blank Define dialog box is displayed.
Access an Entry:	2	Type ―Person Information‖ in the Label field and press ENTER twice. (Pressing ENTER once brings up the Search dialog box. Pressing it a second time displays the entry found. If you press ENTER twice quickly, you get the same result without displaying the search box.) The repository entry for Person Information displays with all of the information that has been entered into the repository for this entry.
Move Around:	3	Click Next. The next entry in alphabetical order is displayed.
	4	Click Prior. Person Information is again displayed.

Jump to Other Entries:

5 Click the element Name in the Attributes field. (It may be necessary to scroll the contents of the field to bring Name into view.) Click Jump. (Click Yes if you are asked if you want to save Person Information.) The repository entry for the data element Name is displayed.

6 Move to page two by clicking the Physical Information tab at the top of the dialog box. (The current page number is displayed in the upper right corner of the Define window.) This displays more information about the current entry, including the Location information that indicates where the current entry is used.

7 Click the line in the Location field containing Person Information. This highlights the line.

8 Click Jump. The entry for Person Information is once again displayed. The Locations tab (page 2) is currently displayed. An alternative to selecting Jump to switch to another repository entry is to double-click the entry name in the Location field or to click the Back button.

9 Move to page one by clicking the Description tab.

Search Capabilities

Searching for entries in the repository is an easy procedure. It can also be a very useful feature because you can set the Search Criteria to display only certain entry types as you move from one repository entry to the next. To search for an entry in the repository:

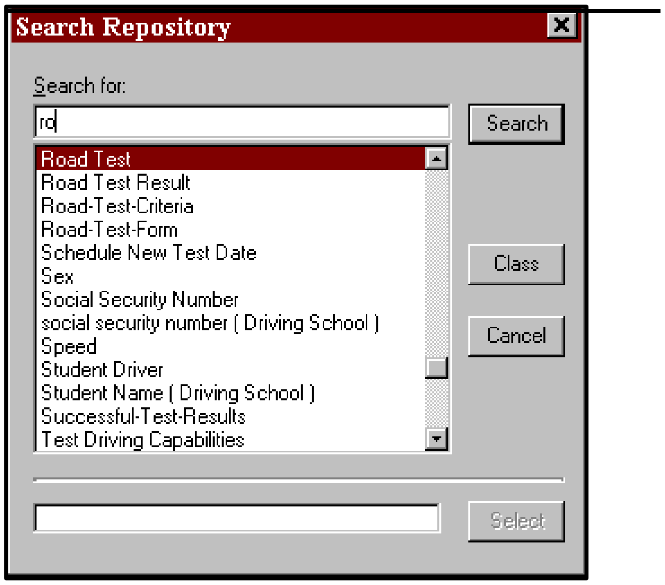
Access an Entry:

1 Click Clear. This clears the dialog box but does not delete the entry.

2 Type —Road Test‖ and press ENTER twice. The repository entry for Road Test is displayed with all of the information that has been entered into the repository for this entry. (This was done for you in the samples included with the TEST project.)

3 Click Clear.

Search for the Entry: 4 Click the Search button to open the search box to select from the repository. Type ―r‖ and entries that begin with ―r‖ appear in the list box. If you now type an ―o,‖ you see that the repository searches incrementally as you type, getting closer to the entry you want.



Road Test
Road Test Result
Road-Test-Criteria
Road-Test-Form
Schedule New Test Date
Sex
Social Security Number
social security number (Driving School)
Speed
Student Driver
Student Name (Driving School)
Successful-Test-Results
Test Driving Capabilities

Figure 17-5 Repository Search Dialog Box

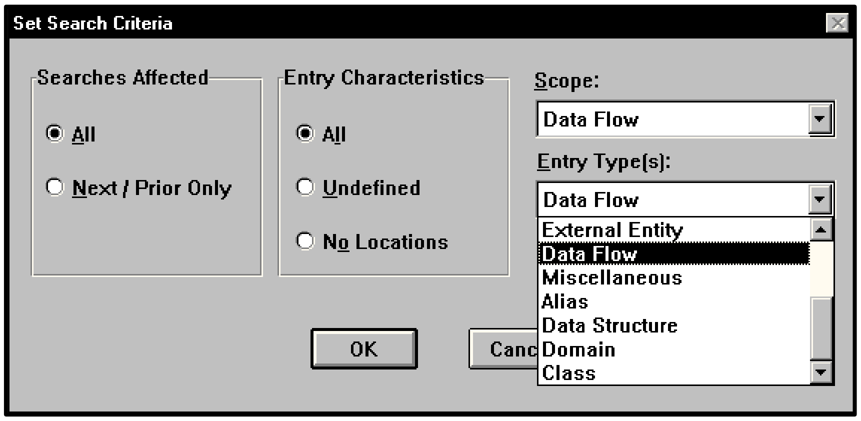
5 Click Road Test and then click Search. The repository entry for Road Test is displayed.

Setting the Search Criteria

Search criteria set the scope of the entries that are displayed as you search through the repository.

Clear the Dialog Box: 1 Click Clear to clear the dialog box.

Set the Criteria: 2 Click Search Criteria. You see a dialog box entitled Set Search Criteria, as shown in Figure 17-6.


Set Search Criteria

Searches Affected
- (●) All
- ( ) Next / Prior Only

Entry Characteristics
- (●) All
- ( ) Undefined
- ( ) No Locations

Scope: Data Flow

Entry Type(s): Data Flow

Figure 17-6 Setting Repository Search Criteria

3 In the box entitled Searches Affected, select All. This is the method used to limit the scope of the entries displayed when navigating the repository using Next and Prior, as well as the entries that are displayed when you select Search.

4 In the box labeled Entry Characteristics, select All. This tells Visible Analyst to search all items in the repository, rather than only those entries that are Undefined or entries that have No Locations. No Location entries are typically those that have been entered directly into the repository rather than added to the repository by being placed on a diagram.

5 Click the down arrow on the right side of the field marked Scope. This allows you to choose the diagram type to which you wish to limit your search. Select Data Flow.

6 Click the down arrow on the right side of the field marked Entry Type(s). This allows you to be very specific about the type of entry to which you wish to limit your search. You can choose individual types and some combination types.

7 Select Data Flow, then click OK.

Try Out the Criteria:	8	At the blank Define dialog box, click Search. Because your search criteria limits searches to data flows, the list displays only the entries in the repository of the type data flow. Select Road-Test-Criteria and then click Search.
	9	Now click Next. The next entry displayed is the next *data flow* in alphabetical order, rather than simply the next *entry* in alphabetical order. If you click Next a few more times, you notice that only data flow entries are displayed.
	10	Click Search Criteria again and set Scope back to Entire Repository. Be sure that Entry Type(s) is set to All. Click OK.

Using Search to Add Items to a Field

The Search feature can also be used to add repository entries to a field without retyping them. This option is very useful for adding multiple data elements to an Attributes field. Instead of typing the name into the field, you can select it using the Search function.

Clear the Dialog Box:	1	Click Clear.
Find an Entry:	2	Type ―V‖ in the Label field and click Search. Valid-Applicant should be the first entry on the list. Click on it and it appears in the Search For field. Click Search and the repository entry for Valid-Applicant appears.
Select Attributes:	3	Click on the Attributes field.
	4	Click the Search button. The available data elements are displayed. Double-click on Address, Birth Date, Name, and Social Security Number. All the selected elements are displayed at the bottom of the Search dialog box as shown in Figure 17-7.

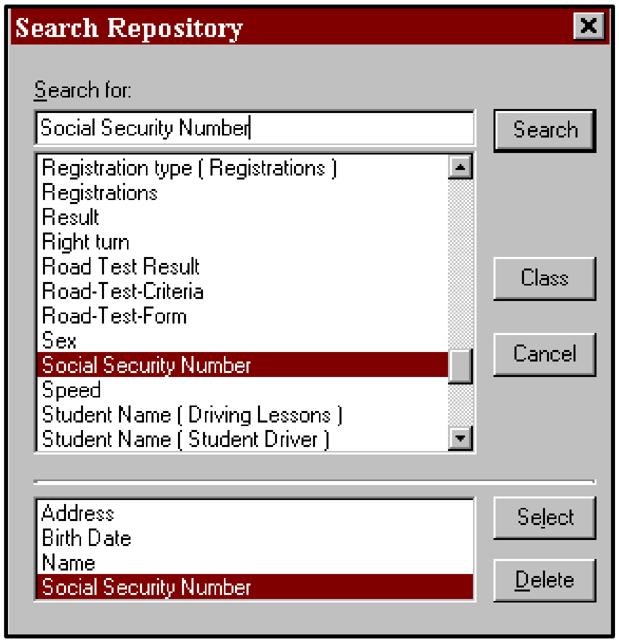


Figure 17-7 Add Information with Search

Add Attributes and Save:	5	Click Select. All the selected elements are added to the Attributes field. Click Save and then click Exit.

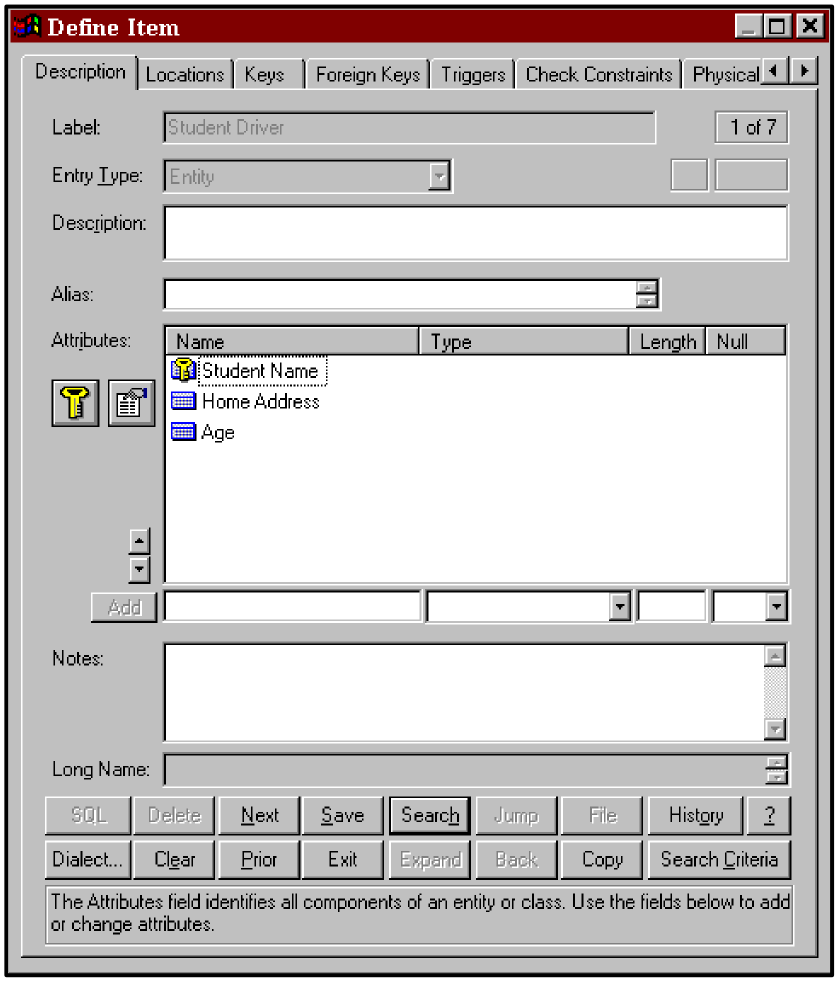
ADVANCED REPOSITORY FEATURES

Adding Information to the Repository

In this unit, you add attribute information to an entity; the attributes consist of the data elements that make up the entity. You also add the primary key information, so that you can demonstrate Key Analysis and Key Synchronization to migrate foreign keys across relationships automatically. All of the key information relates to the method for accessing tables in a database. We assume that each entity corresponds to one table.

Open a Diagram:	1	Open the entity relationship diagram —Driving School View‖.

Display a Repository Entry:	2	Click the ⬉ button on the control bar.
	3	With the *left* mouse button, double-click the entity Student Driver. Its repository entry is displayed.
Enter Attribute Data:	4	Place the text entry cursor in the field immediately to the right of the Add button under the Attributes field. Type —Student Name‖ and click Add. Add —Home Address‖ and —Age‖ in the same manner. Since the data elements you just added to the Attributes field are not already in the repository, entries for each are automatically added when you click Save.
Save the Entries:	5	Click Save to save the attributes you entered.
Enter Key Information:	6	Click the key button to display the Primary Key dialog box. Select Student Name to move it from the Columns in Table box to Columns in Key box. Click OK to return to the Define dialog box. The key notation by Student Name indicates that Student Name is the primary key for this entity.



The Attributes field identifies all components of an entity or class. Use the fields below to add or change attributes.

Figure 17-8 Student Driver Attribute Information

Clear the Dialog Box:	7	Click Clear. This clears the repository dialog box but does not delete the entry from the repository.
Access Another Entry:	8	Type ―Driving School‖ in the Label field and press ENTER.

Add Composition:	9	Click the Attributes Details button and type —Driving School Number‖ and —Driving School Name,‖ each on a separate line. Click OK.
Create Primary Key:	10	Click the Key button next to the Attributes field to display the Primary Key dialog box.
	12	Click Driving School Number in the Columns in Table box to move it to the Columns in Key box. Click OK to return to the Define dialog box.
Save:	12	Click Save to save your changes, then click Clear.
Access Another Entry:	13	Type —Driving Lessons‖ in the Label field and press ENTER.
Add Attributes:	14	Click the Attributes Details button, place the cursor in the Type field, and then click Search.
	16	Scroll the search box until Driving School Number appears. Click Driving School Number and then click Search to enter it on the Attributes dialog box. Move the cursor to Type field of the next line. Add Student Name in the same manner.
	17	Move the cursor to the Type field of the next line, and type —Lesson Number.‖ Click the cursor in the Limit field to enable the Physical Characteristics pane at the bottom of the dialog box. Select Integer as the Data Type.
	18	Click OK to add the attributes to the Define dialog box.
Create Primary Key:	19	Click the Key button next to the Attributes field to display the Primary Key dialog box. Click Lesson Number in the Columns in Table box to move it to the Columns in Key box. Click OK to return to the Define dialog box.
Save and Exit:	20	Click Save and then click Exit.

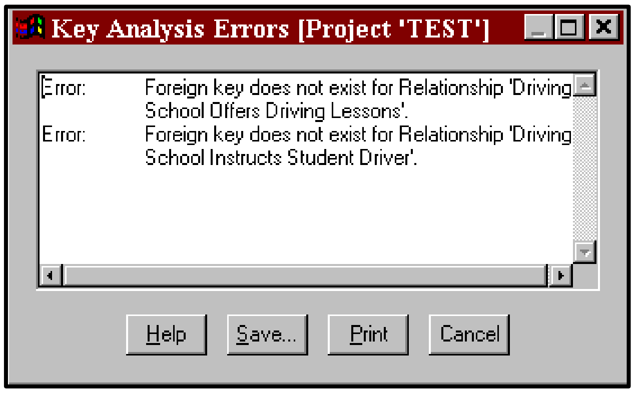
Key Analysis and Key Synchronization

The Key Analysis and Key Synchronization functions, found on the Repository menu, can help you set up a consistent relational database key structure. There are three types of keys used in a data model: primary, foreign, and alternate keys. All keys are designated in the Attributes field of an entity in the project repository. A primary key is one or more attributes or data elements that uniquely identify an entity. To designate a data element as a primary key, the yellow key notation is used in the Define dialog box. On the diagram, primary keys are displayed in the area immediately under the entity name when the primary key level is selected from the control bar or the View/Entity Display Options menu. A foreign key is a non-key attribute in one relation that appears as the primary key (or part of a compound primary key) in another relation. The gray key notation in the Attributes field of an entity designates a foreign key. The FK notation is shown when the entity on the diagram is displayed at the attribute view level.

Key Analysis verifies that the key structure for your data model is complete, checking that all key information is correctly identified for the data model. Key Synchronization analyzes the key structure and migrates data elements that you designate as keys, or parts of compound keys, across relationships to their associated entities, and creates the resulting foreign keys. Using associator element names in relationship repository entries makes this process work better. (Please check the Visible Analyst manual or online help system for an explanation of associator elements.)

Key Analysis and Key Synchronization both involve analyzing the primary key [PK] and foreign key [FK] designations in the TEST project repository. A primary key is an attribute or data element that uniquely identifies a record.

Run Key Analysis:	1	Select Key Analysis from the Repository menu. Visible Analyst scans the entire repository and indicates any errors it finds.



Error: Foreign key does not exist for Relationship 'Driving School Offers Driving Lessons'.
Error: Foreign key does not exist for Relationship 'Driving School Instructs Student Driver'.

Figure 17-9 Key Analysis Error Messages

View the Errors: 2 Click the Maximize button in the upper right corner of the errors dialog box. Scroll through the messages. You see that there are error messages indicating missing foreign keys for the entities on the current diagram.

Note

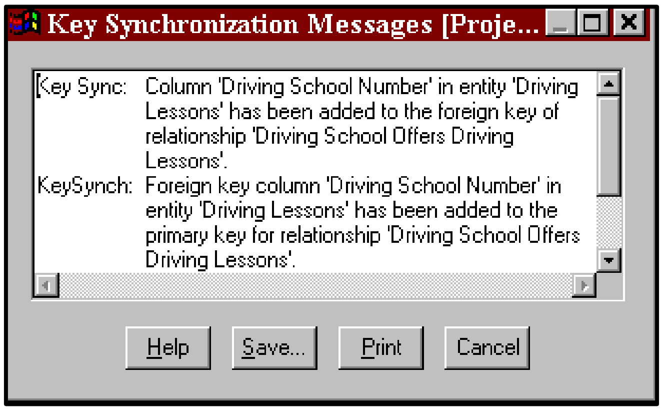
🗎 You can keep analysis error dialog boxes on the screen while you carry on various Visible Analyst activities. This is to make it easier for you to correct the errors found by Analyze. The same holds true for SQL Schema Generation, Shell Code Generation, etc.

.

3 Click Cancel.

4 Select Key Synchronization from the Repository menu. Visible Analyst first analyzes for key errors and then migrates the foreign keys across relationships.

5 Maximize the Key Synchronization Messages dialog box. Key Analysis messages appear first, followed by Key Synchronization messages. You should notice the Key Synchronization messages, indicating the foreign keys that have been migrated.


Key Synchronization Messages [Proje...

```
Key Sync:   Column 'Driving School Number' in entity 'Driving
            Lessons' has been added to the foreign key of
            relationship 'Driving School Offers Driving
            Lessons'.
KeySynch:   Foreign key column 'Driving School Number' in
            entity 'Driving Lessons' has been added to the
            primary key for relationship 'Driving School Offers
            Driving Lessons'.
```

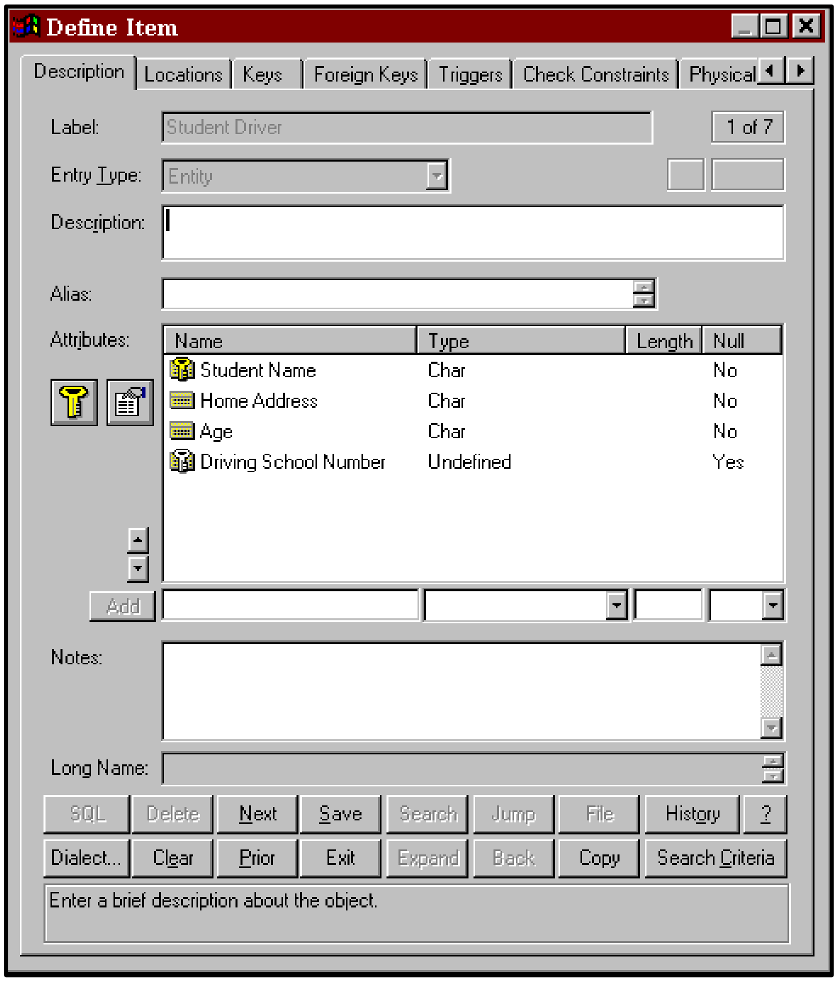
Figure 17-10 Key Synchronization Messages

6 Click Cancel.

Examine the Migrated Key:

7 Double-click Student Driver. Notice the foreign key Driving School Number that has been added. This was done by Key Synchronization. It saves you from migrating all of the foreign keys manually.

Note also that Analyze added text describing the key. All text following an asterisk is considered a comment and is ignored by the repository. (When the object interface is enabled, comments are not displayed.)



Label: Student Driver

1 of 7

Entry Type: Entity

Description:

Alias:

Attributes:

Name	Type	Length	Null
Student Name	Char		No
Home Address	Char		No
Age	Char		No
Driving School Number	Undefined		Yes

Notes:

Long Name:

Enter a brief description about the object.

Figure 17-11 New Foreign Key Information

8 Click Exit.

9 Deselect Student Driver on the diagram.

View Objects

Visible Analyst Corporate and Zachman Editions support the concept of an SQL view, which can be thought of as a derived or virtual entity. A view is similar to an entity in that it has a composition, but the items that appear in the composition of a view must belong to other entities or be expressions based on data elements used by another entity.

An SQL view is made up of two major components: a list of column names and a select statement that is used to filter information from the tables in the view. The select statement can contain not only the primary select clause, but also any number of sub-selects and union selects. When view is selected as the entry type, view-specific Define dialog box pages are displayed. Using these pages, you can select tables, columns, join relationships, clauses, and other options for the view. An expression builder is available to help you create the expressions to be used in the filter, group by, having, start with, connect by, or join expression controls.

Detailed information about views can be found in the *Operation Manual* and in the online help system.

Note

🗎 Views are not available in the Education Editions of Visible Analyst.

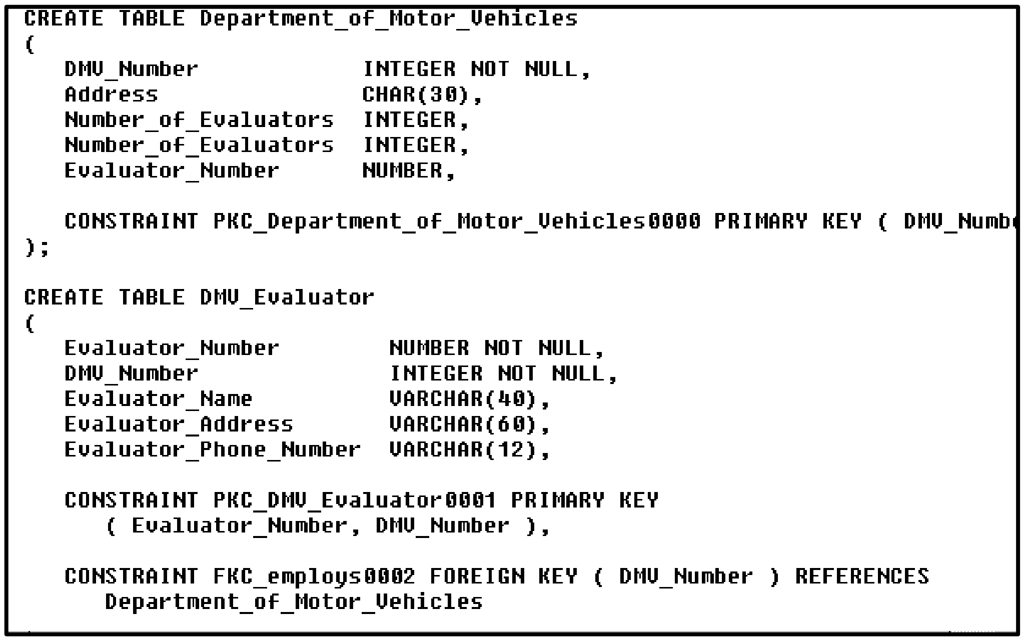
Generate Database Schema

The Corporate and Zachman Editions of Visible Analyst generates SQL DDL (Structured Query Language – Data Definition Language) schema from the information contained in the repository. In the Corporate and Zachman Editions, you can select from several different dialects of SQL, including a User Defined type, to allow the use of a dialect not currently supported by Visible Analyst. For more information on the custom feature, see the *Operation Manual* or the online help system. The statements that are supported include CREATE TABLE, CREATE INDEX, and COMMENT ON. More information is contained in the *Operation Manual* or in the online help system.

The Education Editions of Visible Analyst allow you to generate SQL for Microsoft Access and Oracle only. To generate SQL:

Choose Access Dialect:	1	Choose SQL Dialect from the Options menu, then choose Access.
Generate SQL Schema:	2	Select Generate Database Schema from the Repository menu to generate the schema. When the dialog box appears, click OK. (Refer to the *Operation Manual* or online help system for details of the SQL Schema Generation dialog box.) If errors are found, they along with the generated schema will be displayed.
View the Schema:	3	Maximize the SQL generation dialog box.

4 Click the Schema button to display the generated schema. See Figure 17-12. If Visible Analyst does not have the information to generate the schema, a list of errors is displayed; but no Select box is present. Click the Errors button to view any errors. (If too many errors are generated, the Schema button is not displayed.)



```
CREATE TABLE Department_of_Motor_Vehicles
(
   DMV_Number            INTEGER NOT NULL,
   Address               CHAR(30),
   Number_of_Evaluators  INTEGER,
   Number_of_Evaluators  INTEGER,
   Evaluator_Number      NUMBER,

   CONSTRAINT PKC_Department_of_Motor_Vehicles0000 PRIMARY KEY ( DMV_Numb
);

CREATE TABLE DMV_Evaluator
(
   Evaluator_Number        NUMBER NOT NULL,
   DMV_Number              INTEGER NOT NULL,
   Evaluator_Name          VARCHAR(40),
   Evaluator_Address       VARCHAR(60),
   Evaluator_Phone_Number  VARCHAR(12),

   CONSTRAINT PKC_DMV_Evaluator0001 PRIMARY KEY
      ( Evaluator_Number, DMV_Number ),

   CONSTRAINT FKC_employs0002 FOREIGN KEY ( DMV_Number ) REFERENCES
      Department_of_Motor_Vehicles
```

Figure 17-12 Generated SQL Schema

Shell Code Generation

The Corporate and Zachman Editions can generate C and COBOL shell code. The code that is generated encompasses the sequence of functions or paragraphs that make up a program, including global definitions, descriptive comments, function call/PERFORM statements, and passed parameters. Information entered in text fields in the repository entry for a program item or a structure chart module produces comments that describe these items within the generated code. Also, actual source code can be entered in the module description field of a module or macro, and this code is placed in-line with the function calls or PERFORM statements that are generated by invocations. Couples or ITRs used with invocation lines generate parameters for C code. There is also an option to customize the code to be generated. (See the online help for other generation options supported by Visible Analyst, such as AS/400 DDS, Visual Basic, PowerBuilder, etc.)

XML Generation

Visible Analyst can generate the XML Schema based on the W3C standard by selecting the Tools | Export | XML Schema (XSD) menu option. The XML file is generated for the entities and (optionally) classes developed in the project. The XML file is written to the Visible Analyst TRANS folder.

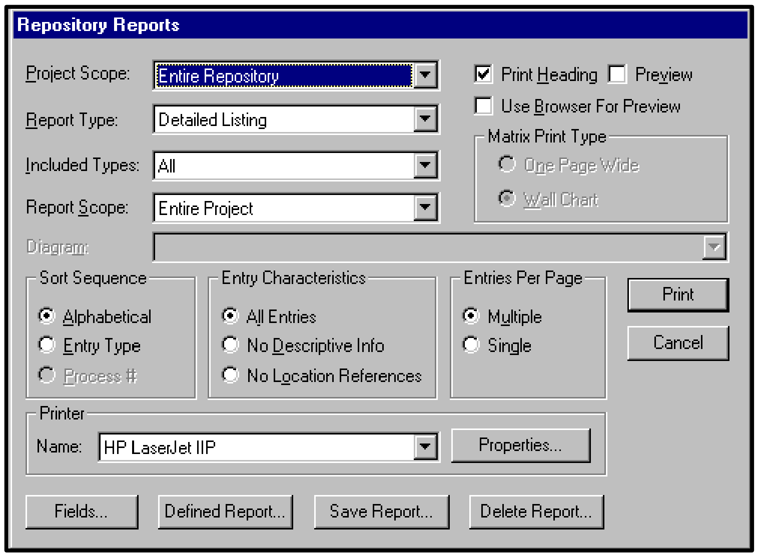
XML DCD code can also be generated based on the data models. This is similar to SQL schema generation. XML can be selected as the generation option when you select SQL Dialect from the Options menu. The procedure is similar to the SQL DDL generation. See the *Operation Manual* or the online help for more information.

Repository Reports

Now you practice generating a report on the data contained in the repository. This is a basic report containing a detailed listing of all entries contained in the repository. For detailed information about Reports and Report Queries (Custom Reports), see the *Operation Manual* or the online help system.

First set the font for the report you want to generate.

Set the Report Font:	1	From the Options menu select Text Settings.
	2	Under Text Type, select Report Body.
	3	Select a typeface and point size, and click OK.
Set the Report Criteria:	4	Select Reports from the Repository menu. The Repository Reports dialog box appears (see Figure 17-13).



Repository Reports

Project Scope: Entire Repository

Report Type: Detailed Listing

Included Types: All

Report Scope: Entire Project

Diagram:

[✓ Print Heading] Preview

Use Browser For Preview

Matrix Print Type: One Page Wide [✓ Wall Chart]

Sort Sequence: [✓ Alphabetical] Entry Type Process #

Entry Characteristics: [✓ All Entries] No Descriptive Info No Location References

Entries Per Page: [✓ Multiple] Single

Printer Name: HP LaserJet IIP

Figure 17-13 Repository Reports Dialog Box

5 Under Project Scope, select Entire Repository.

6 Under Report Type, select Detailed Listing.

7 Under Included Types, select All.

8 Under Report Scope, Entire Project is selected.

9 In the box labeled Sort Sequence, select Alphabetical. This determines the entry order in your report printout.

10 In the box labeled Entry Characteristics, select All Entries.

11 In the box entitled Entries Per Page, select Multiple Entries Per Page. You can select Single Entry Per Page to reorder the pages of your report once they have been printed.

Run the Report: 12 Click Print; the information is sent to the printer. Select Preview to view the report first.

Note

🗎 Reports can be generated in HTML format so that they may be viewed in a browser. When you select Preview, the Use Browser for Preview Option is enabled. If you select this option and you have a browser on your PC, the report is generated and displayed in your browser.

13 Click Cancel when printing is complete to exit the Repository Reports dialog box.

Where To Go From Here

OVERVIEW

This concludes the Visible Analyst tutorial. To exit the program, select Exit from the File menu.

You have now completed exercises in many of the major elements of planning, structured analysis and design, data modeling and object modeling:

- Drawing diagrams to model a system.
- Using methodology rules to insure against inconsistencies.
- Adding written definition to the graphic model.
- Embellishing the model following the initial layout.
- Expanding the model through definition of repository elements.
- Generating reports.

All structured software and systems engineering involve these same basic operations.

REAL WORLD APPLICATION

The example project that you created was a simple one. The real power of MDA is in the application to systems too complex to keep in your head at one time, too large to be reviewed by inspection, too widespread to have only one person working on the whole job. These types of projects include nearly every system designed today. That power shows itself in four areas:

- The assurance of accuracy and completeness, that no elements are left dangling or unaccounted for.
- The prompting and reminders that error checks and repository output provide, to focus attention in the midst of a dauntingly complex assignment.
- The word processing-like ease with which changes and modifications can be accomplished, while ripple effects are flagged and accounted for.
- The convenience of thorough documentation that is produced concurrently with the design, not as a drudgery-filled after-effort.

The power of MDA further multiplies into substantial gains in productivity, communication and quality when applied to team effort. The Corporate and Zachman Editions of Visible Analyst have significant capabilities for exporting and importing information, either through portable media or through participation in a local area network. This allows Visible Analyst to become an integral part of any development environment, sharing information and expanding the value of labor. Many groups thus can benefit from another group‘s hard work.

Finally, the application of the power of MDA productivity enhancement is not limited to software but can be applied toward analyzing and designing any system, such as:

Manufacturing	Medical Diagnostic Analysis
Planning	Command and Control Operations
Processing	Administrative Procedures
Legal/Judicial	Inventory Control
Audits	

Visible Analyst is designed to be a natural extension of the way you think, create, and analyze. Our goal is to —Put MDA Within Everybody‘s Reach‖ and to make MDA tools and the methodologies of structured analysis, design, data modeling and object modeling a natural, seamless, integrated part of your everyday work, rather than an arcane ritual to be occasionally endured by specialists. The integration and flexibility to use the components and elements that you deem necessary for your application provide you with a customizable tool set that can be adapted to support how you choose to work on the design and development of information systems.

WHAT TO DO NEXT?

There are still many things for you to do to become comfortable and committed to the use of a MDA tool. We suggest the following:

- Study and review the logic concepts introduced in the tutorial (consider reviewing the referenced materials).
- Review the tutorial steps and practice any areas that were unclear.
- Select a personal real-world project to do using Visible Analyst. Make it a modest-sized effort to give you some time to explore and experiment with the tool.
- Select the parts and components that you want to use in your software development practice.
- Practice, practice, practice. Make adjustments where needed.
- Make the tool a regular part of all of your projects.
- Define standards and procedures for library components.
- Build disciplines and skills with concepts and Visible Analyst.
- Use technical support as needed - don‘t get stuck, don‘t become frustrated.

- Stop and evaluate what you have done, show others, review the work of others - find ways to improve the content and the processes.
- Practice reusability wherever possible.
- Stay up to date with the tool and the evolving methodologies.
- Build a library of materials for use in future projects.

CONCLUSION

We hope that these lessons have helped to make you feel comfortable with the planning, structured systems analysis and design, and data modeling tools, and their implementations in our product.

For more information or if you have any questions about MDA or structured analysis, please contact the Visible Systems technical support staff:

Telephone	(617) 902-0767
FAX	(508) 302-2400

https://www.visiblesystemscorp.com

Index