

Visible Analyst®

Tutorial



Visible
Systems Corporation

Visible Analyst® Tutorial

A Model Driven Approach To Enterprise Architecture Planning, Analysis, Design and Development

Visible
Systems Corporation

This tutorial was designed to work with the following versions of the Visible Analyst:

- Visible Analyst – Project and Team Editions
- Visible Analyst – Professional Edition
- Visible Analyst – University Edition
- Visible Analyst – Community Editions
- Visible Analyst – Student Edition

Information in this document is subject to change without notice and does not represent a commitment on the part of Visible Systems Corporation. The software described in this document is furnished under a license agreement or non-disclosure agreement. The software may be used or copied only in accordance with the terms of this agreement. It is against the law to copy the software onto any medium except as specifically allowed in the license or non-disclosure agreement.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or otherwise, including photocopying, reprinting, or recording, for any purpose without the express written permission of Visible Systems Corporation. Visible Systems Corporation makes no representations or warranties with respect to the contents or use of this manual, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Names, dates, and information used in examples in this manual are fictitious and only for examples.

Copyright 2023 by Visible Systems Corporation, All rights

reserved. Printed and bound in the United States of

America.

This manual was prepared using Microsoft Word for Windows.

Visible Analyst

Tutorial on Structured Methods, Repository Management and The Zachman

Framework Visible® is a registered trademark of Visible Systems

Corporation.

The Zachman Framework illustration on the cover page of this tutorial was printed and used with the permission of the Intervista Institute © 2004 (www.intervista-institute.com). Microsoft and Windows are registered trademarks of Microsoft Corporation. Other product and company names are either trademarks or registered trademarks of their respective owners.

**Visible Systems
Corporation
24 School Street
2nd floor
Boston, MA 02108**

Technical Support: +1 617-902-0767

E-mail support@visiblesystemscorp.com

Internet: <https://www.visiblesystemscorp.com>

E-mail: sales@visiblesystemscorp.com

Dear Colleagues:

Thank you for your time in selecting our product, the Zachman Framework Edition of 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. And, we feel that we offer the very best in product support that often differentiates us from our competitors.

As you read through 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).

The following information outlines all you will need to know in order to get started in building your Enterprise Architecture. We hope that your first project will be a success.

The project TEST is automatically installed and is used in conjunction with the tutorial file "tutor.pdf" written to the installation directory and this tutorial book. Use the File | Select Project menu item to select this project. Included is a backup file set of the Zachman project and a copy of the document "Visible Analyst framework.doc" describing the project. This project and document explain which diagram or repository entry is used as the cell artifact.

Perform this procedure to restore the project to the Visible Analyst.

- * Open the Visible Analyst and choose the Tools | Restore menu item.*
- * At the first restore screen, click the Browse button next to the "Backup File Name" field.*
- * Point and click to the file "ZACHMANBACK.VSC" located in the VA\Zachman folder on the CD.*
- * Click on the file so that it is highlighted and click OK.*
- * The name of the project is displayed in the Name field on the restore dialog, so click the OK button.*
- * The second screen displays the path to the VA\Zachman folder, so click OK again to perform the restore.*
- * The project will be restored to the Visible Analyst.*

Use the File | Open Diagram menu item to access the diagrams directly, or use the File | Zachman Framework to display the framework. Click on a framework cell to view the artifact types associated with the cell. Double clicking on an item will open the diagram or display the artifact's repository entry.

Best Regards,

*Mike Cesino
President & CEO
Visible Systems Corporation*

Lesson

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), entity life history, activity, use case, sequence, collaboration, component and structure chart (product) models for an information system. The Business Process Modeling Notation (BPMN) in 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 proceeds 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 <http://www.visible-systems.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.

Component Model - Component diagrams allow you to show the structural relationships between system components.

Entity Life History Model - The Entity Life History models show how events in a system affect data entities.

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

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 rule's methodology.

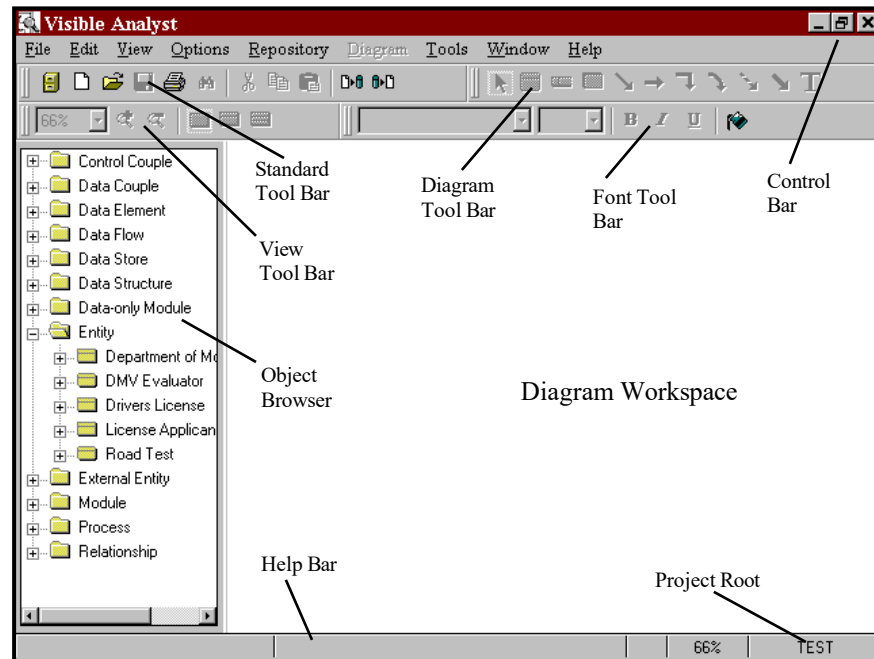


Figure 1-1 Visible Analyst Workspace

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 – 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.

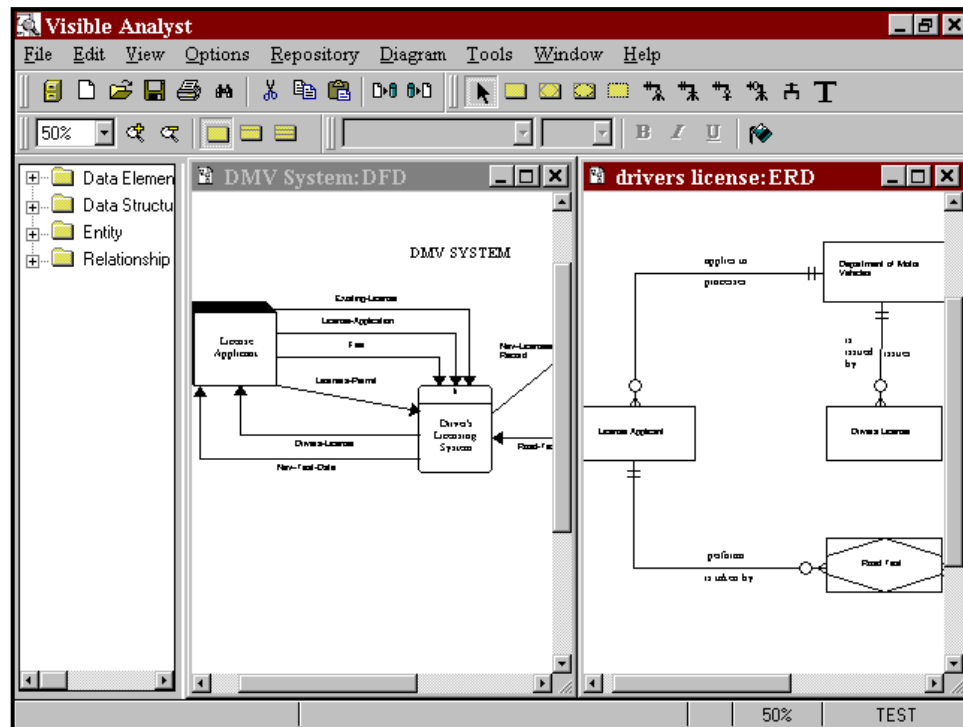


Figure 1-2 Visible Analyst Multiple Document Interface

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+Y	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 **F**ile 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

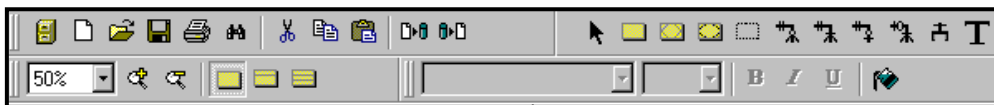
The control bar, shown in Figure 1- 3, is located above the diagram workspace and gives you quick access to commonly used functions and types of objects that can be added to a diagram. The control bar can contain up to four tool bars.

- 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

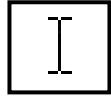


Figure 1-6
The Text Cursor

Figure 1-5
The Line 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.

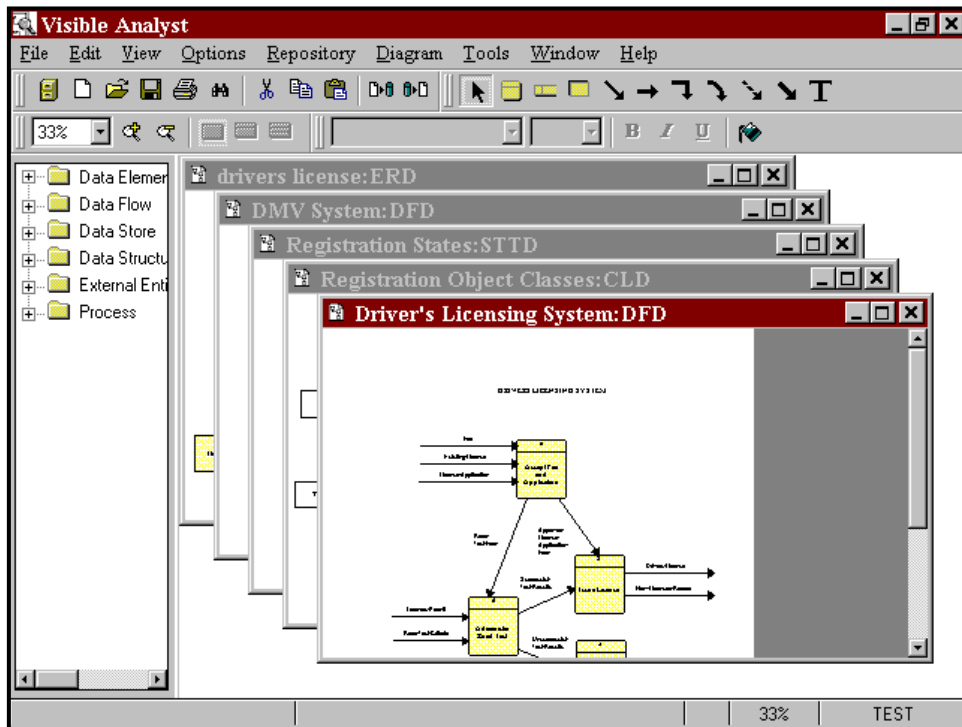


Figure 1-8 Cascaded Multiple Diagram Windows

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 Analyst *Operation Manual* and the online help system (accessed by pressing F1).

Lesson

Structured Modeling Techniques

OVERVIEW

The techniques for planning, process modeling, data modeling, object modeling, state transition modeling and structured design assist in the creation of systematically correct and consistent diagrams and documentation. Using structured and object techniques forces a standardization of logic throughout the system under analysis. The benefits of this approach are obvious:

- Large systems can be partitioned into component subsystems or sub-functions for further analysis.
- Specifications for individual components are easier, faster, and more accurate to define than the total system.
- The interaction between the parts can be planned, designed, evaluated, and implemented to reflect improved information flows and controls.
- More than one person can work with the same system in the network edition.
- Standardized format and grammar enhance and simplify communication and maintenance.

STRUCTURED PLANNING

Planning uses a structured technique based on functional decomposition for describing interrelationships among broad organizational areas, specific organization functions, and the systems that support those functions. Structured planning establishes organization responsibilities at function levels and then defines the process responsibilities within functions.

The objective of structured planning is threefold:

- To identify the specific business or organization function, including roles, goals, and objectives, to be automated or reengineered.
- To identify the existing system processes that support that function.
- To provide a focus for requirements analysis in support of identified goals and objectives.

For example, functions or functional areas in an organization that could be decomposed could be Finance, Sales, and Research. A function is usually designated by a *noun*. These functional

areas could then be subdivided into processes that are groups of activities necessary in running the organization. The processes are usually defined in active state *verbs*. For example, the Sales function could be decomposed into the Customer Relations, Selling, and Management processes. These processes could then be further decomposed using a data flow diagram. If a process is labeled as a noun, it is a signal that the process should be further decomposed into more processes.

Because of the high-level functional nature of this type of modeling, the technique specifically applies to functions and not to the data that those functions use. Since functional decomposition modeling is viewed as the highest level of business planning, it is probably the place to begin when you wish to define the overall functioning of an enterprise. There is no rule that you must begin here, but other things are easier if you do. For designing individual projects, it may be just as effective to start with a process model or a data model (or both at once), for you might consider that the project does not have the breadth to warrant planning at the FDD level¹. You might also choose to focus on the definition of objects, beginning with the object/class model.

STRUCTURED DESIGN

Structured design is the partitioning of a system into a hierarchy of modules that performs the activities internal to your system. It is a technique used for representing the internal structure of a program or system and its components. Structured design is a discipline that is complementary to structured analysis and implements another stage in the software life cycle. If data flow diagramming is the “what” of your system, structured design is the “how.” To be most effective, it should be based upon specifications derived using structured analysis. The capability to integrate analysis and design verifies that your designs reflect the reality of your specifications.

The modeling technique used in structured design is the structure chart. It is a tree or hierarchical diagram that defines the overall architecture of a program or system by showing the program modules and their interrelationships. Visible Analyst uses the structural information contained in the system model in the code generation process to create the precise infrastructure of your system. This includes the passing of control and parameters between program modules, as well as the specific order in which the modules are arranged in your code.

A module represents a collection of program statements with four basic attributes: input and output, function, mechanics, and internal data. It could also be referred to as a program, a procedure, a function, a subroutine, or any other similar concept. A structure chart shows the interrelationships of the modules in a system by arranging the modules in hierarchical levels and connecting the levels by invocation arrows designating flow of control. Data couples and control couples, designated by arrows, show the passing of data or control flags from one module to the next. This is equivalent to passing parameters between functions or procedures in an actual program.

The Visible rules implementation of the Yourdon/Constantine structured design methodology is intended to maintain as much design freedom as possible for you, while guarding against known poor design practices. The error and warning messages generated are intended to be used as guidelines rather than rules.⁵

OBJECT-ORIENTED MODELING

Object-oriented modeling concentrates on developing a collection of discrete objects that incorporate both data structure and behavior. The objects perform or are impacted by operations that represent the action between objects. The focus is on building object definitions that can be organized into a class hierarchy with high-level abstractions of a class of like objects that provide inheritance of characteristics to subclasses and eventually to individual instances or a unique occurrence. Objects can be brought together into groups called aggregations, and they can have relationships and attributes (called properties) similar to those found in the entity relationship model. In fact, the data model (ERD) is the basis for object-oriented concepts with its entities and attributes.

OBJECT CONCEPTS

The object model is used to define and build the classes and subclasses of objects and the data characteristics that uniquely define object groups. By developing a clear picture of object structures and operations needed to support a business process, the designer can build reusable object components and save time and effort in the development and testing phases of

⁵For more detailed information on the Yourdon/Constantine structured design technique you can refer to the following books (the Page-Jones book is the better choice for beginners):

Page-Jones, M. *The Practical Guide to Structured Systems Design*. Englewood Cliffs: Prentice-Hall, 1988.

Yourdon, E.N. and Constantine, L.L. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Englewood Cliffs: Prentice-Hall, 1986.

the project. The object model is a static model in that it defines all of the objects that are found in the application and the general and specific characteristics of each object.

The object model shows a static snapshot of the hierarchy and packaging of the objects. The data model is a static snapshot of the data components of the application and the relationships between data components. The data flow diagram (process model) shows the flow and sequence of operations of the application. The state model shows the dynamic changes that occur within the applications and to the objects over time. The structure chart (physical model) defines how the application is assembled and built.

STATE TRANSITION (DYNAMIC) MODELING

The state transition model focuses on the changing conditions and states of an object. As an object such as a Customer Order progresses through an organization, it changes its state from a pending order to a shipped order to a paid order. The movement of the order from one state to another changes some of the object's properties and is usually caused by an event or a method being applied to the object.

The dynamic model is built after the object model is defined. It provides a sequence of states of the objects as they change over time. Thus the object model is static and complete, and the state (dynamic) model is continuously changing with different events and triggers. The state model is closest to reality and supports the programming design mechanics. If the programs cover all of the state transitions of the objects, then the system should fit to reality.

The object and the state-transition model are linked to the functional model that describes the data transformations of the system. The functional model can be represented by data flow diagrams with processes and data flows showing how objects are serviced through their time sequence transitions.⁶

OBJECT MODELING AND PROCESS MODELING

From the static snapshot of the objects, to the dynamic changes of states, to the sequence of operations in the data flow, to the build specifications of the structure chart, object-oriented methodologies provide a complete mechanism for defining, designing and building information systems. Object concepts provide an alternative and a complement to the structured design methodologies. Both approaches define the data components of the application and provide a view of how the application needs to act to provide service and

⁶For a detailed discussion, refer to:

Rumbaugh, Blaha, Premerlani, Eddy and Lorensen. *Object-Oriented Modeling and Design*. Englewood Cliffs: Prentice-Hall, 1991.

support to the application users. The differences are mainly in the focus on components, the order of occurrence and the formats of packaging.

Both techniques and methodologies link to the happenings of the real world. They must both produce working information systems. They also share many similar concepts, such as reusability, modularity and hierarchical structures.

Visible Analyst supports both approaches to systems design and development. They can be used separately, together, or in any combination that suits you. Through the integrated Visible Analyst repository and the independence of the diagrams, you can maintain maximum flexibility and still take full advantage of the engineering practices for designing and developing better information systems.

DATA AND OBJECT RELATIONSHIPS

There is considerable similarity between entity and object models. Both focus on defining physical components: in the entity model the only elements are data or data-oriented components; in the object model, the focus is on real components that can be data, physical units, goods, materials, etc.

The general consideration is that the object model follows the design of the data model, but has made the application more worldly and generic.

LIBRARY MODEL

The library model contains the recorded information about all the pieces, parts, components, actions and conditions of the project. As objects are placed on diagrams and labeled, the label creates an entry in the library database for the proper data logic to support the type of graphic object. The library model is dynamic and evolutionary and is used to describe all the known factors and facets of the application and the systems development project. The Visible Analyst repository is the implementation of a comprehensive library model. It contains all of the labeled parts of the diagrams, and it provides a facility for expanding the details and definition of many components of the project. The Visible Analyst repository can support the building of data elements, database keys, pseudo code, test data and other specifications of the application. Free-form notes and description fields allow recording extensive comments, findings, important information, and other relevant factors about project components. Detailed reports and the generation of database schema, shell code, and other useful project outputs are derived from the library model.

The library model can serve the project design and development process, and it can be a useful reference source for maintenance and operation of the system as well as a key resource when changes need to be made to the system.

Lesson

Diagramming and Repository Basics

INTRODUCTION

This lesson introduces you to the diagramming tools. You learn the basic techniques for creating and modifying any type of diagram in Visible Analyst. We use the unstructured diagram format that does not require you to “follow the structured rules.” This allows you to concentrate on the basics of the drawing process without worrying about rules and the repository. Sometimes you just want to draw a diagram, but not as a part of an analysis or design project. (A number of examples are shown the *Operation Manual*.) Also, some diagrams created from standard diagram types, such as cluster diagrams from entity relationship diagrams (ERDs) and process decomposition diagrams from data flow diagrams (DFDs), are always unstructured. You should know how to access them.

The basic techniques of drawing diagrams are valid for unstructured and structured diagrams. We could just as well use the DFD type or the ERD type to teach basic diagramming techniques, but not all users have a Visible Analyst version that contains all diagram types. However, all Visible Analyst packages have unstructured diagram capability. The diagram drawn in this lesson has no meaning other than as an exercise and is not part of any other lesson.

CREATING A NEW PROJECT

Each project that you create represents one complete system. One project could also be used to depict one unit in a very large system. By maintaining the entire system in one project, Visible Analyst ensures that the entire system remains consistent throughout the entire development process rather than checking for global consistency once all of the units have been merged together. The LAN version of Visible Analyst allows multiple designers to work on diagrams in the same project.

Note

- Different types of lines are available for each type of diagram. You can select the line type for any one of the other available diagram types for use with an unstructured diagram. This selection must be made before a diagram is created.

- Open the Menu:*
- 1 Click on the **File** menu with the *left* mouse button.
 - 2 Select **New Project**. A dialog box like that in Figure 5-1 is displayed.

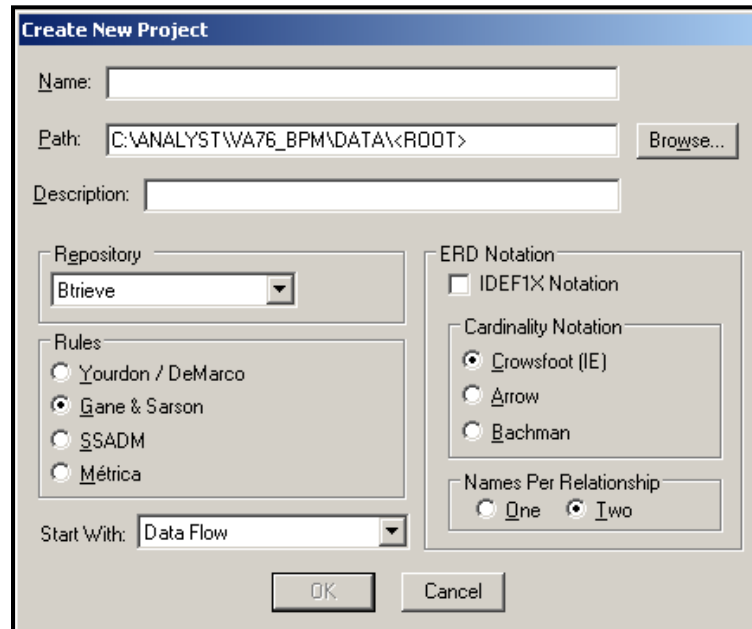


Figure 5-1 New Project Dialog Box

- Name the Project:*
- 3 Type TUTOR into the Project Name field. A project name or “root” can be up to 128 characters long. It must begin with a letter and can be composed of letters and numbers.
- Describe the Project:*
- 4 Click the cursor inside the field marked Description. Type “Tutorial Project.” Another common method for moving the cursor to other fields in a dialog box is to use the TAB key. Try pressing the TAB key a number of times. The highlighted selection changes as the cursor moves to a new field. Press the TAB key until the cursor returns to the Description field.

The next steps help familiarize you with the options available when creating a project. If a default is incorrect, you can click on the item to change the setting.

<i>Select the database Engine:</i>	5	Choose Btrieve as the database engine. Btrieve is included with Visible Analyst.
<i>Select the Rules to Apply:</i>	6	In the box entitled Rules, select Gane & Sarson. This is where you choose the rule set you want applied to your project. An unstructured diagram does not follow any rules, but it is necessary to select the type of rules to be applied to all of the diagrams that might be created for this project. (Rules are covered in more detail in Lesson 8 – Data Flow Diagrams.)
<i>Select ERD Notation Conventions:</i>	7	In the area entitled ERD Notation, the default notation is IDEF1X. This selects the type of relationship line notation you use on your Entity Relationship Diagrams. Crow'sfoot is selected as the alternate. (This is covered in more detail in Lesson 7 – Entity Relationship Diagrams.)
	8	In the box entitled Names Per Relationship, the default is Two. This refers to how relationship lines on ERDs are labeled.
<i>Choose a Window Type:</i>	9	Select the type of document window that will be opened automatically after the project is created. If you don't want a window opened (if you will be performing a Reverse Engineering operation for example), choose None.
<i>Activate the Project:</i>	10	Click OK to activate the project. When you do this, the New Diagram dialog box is automatically displayed.

Now you have created a project. The name of your project is displayed in the lower right-hand corner of the application workspace. If you turned off the help bar from the **Options** menu, the project name is not displayed. The next step is to create a diagram.

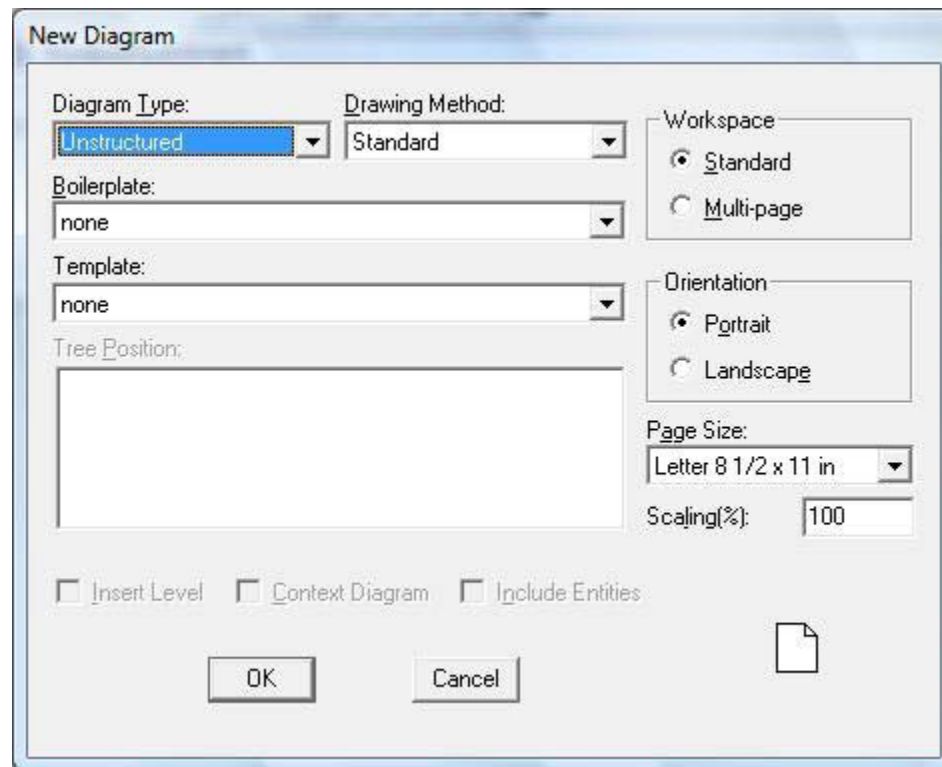


Figure 5-2 New Diagram Dialog Box


CREATING A NEW DIAGRAM

After creating a project and before creating a diagram, the screen should look just as it did when you started Visible Analyst, except that the name of your project appears in the lower right-hand corner. To create a new diagram, follow these steps:

- | | | |
|------------------------------|---|--|
| <i>Open the Menu:</i> | 1 | Click on the File menu (or click the New Diagram button on the standard tool bar). |
| | 2 | Select New Diagram . A dialog box like the one in Figure 5-2 is displayed. |
| <i>Set the Diagram Type:</i> | 3 | Open the selection box for the Diagram Type by clicking the arrow at the end of that field. Select Unstructured . The fields marked Boilerplate and Template should say None. |

A boilerplate is a template you can create to keep information such as diagram creation date, diagram created by information, and a diagram heading without rewriting it each time you create a new diagram. Templates allow you to add image files to an unstructured diagram template file, and create unstructured diagram using the images as the diagram symbols. (See the *Operation Manual* or the online help system for more information on boilerplates and templates. Boilerplates and templates are not enabled in the Education Editions of Visible Analyst.)

- | | | |
|----------------------------|---|---|
| <i>Select a Workspace:</i> | 4 | In the area entitled Workspace, select Standard. This sets your diagram size to one page. Multi-page allows you to spread large diagrams over a workspace of 90 x 88 inches. You can go to larger pages as needed later, or select them now if you know you are going to work on a large diagram. (Multi-page is not enabled in the Education Editions of Visible Analyst.) |
| <i>Select Orientation:</i> | 5 | In the Orientation area, select Landscape. |
| <i>Select Page Size:</i> | 6 | Open the Page Size drop-down list and select 8-1/2 x 11. |
| <i>Select Scaling:</i> | 7 | Accept the default scaling, 100%. |
| <i>Create the Diagram:</i> | 8 | Click OK to open a blank diagram. |

The control bar is located just above the diagram and below the menu. The  button is highlighted.

Above the menus, notice the title of your diagram. Since it has not been saved, it is marked Untitled: US. The US indicates that the window contains an unstructured diagram.

EDITING A DIAGRAM

Adding Symbols to a Diagram

Now add symbols to the diagram to become familiar with the different methods for doing this.

- | | | |
|------------------------------------|---|--|
| <i>Turn on Auto Label Symbols:</i> | 1 | Open the Options menu. There should be a check mark next to the selection Auto Label Symbols. This indicates |
|------------------------------------|---|--|

that you are automatically prompted to label a symbol as soon as it is drawn. If there is no check mark next to the selection, set the option by clicking on the selection.

<i>Change to Symbol-Adding Mode:</i>	2	Click the first symbol button in the control bar, then slowly move the cursor from button to button. As you move the mouse over each button, a brief description appears on the control bar describing its function. Icon buttons are added to the control bar for each type of symbol available to you for the current diagram type. Only certain symbols are available for most types of diagrams, but they are all available for an unstructured diagram. When you move the cursor back over the drawing area, it changes to indicate that you are in symbol adding mode
<i>Position the Symbol:</i>	3	Place the cursor where you would like the symbol to appear on the diagram and click the <i>left</i> mouse button. The symbol is drawn. Because Auto Label Symbols is turned on, a dialog box appears for labeling the symbol.
<i>Label the Symbol:</i>	4	Type “First” into the Text field.
	5	Click the OK button.
<i>Repeat for Another Symbol:</i>	6	Click the third symbol button and add it to the diagram as above.
	7	Type “Second” into the Text field and click OK. Note that the new symbol is now highlighted, indicating that it is the <i>current object</i> , and the previous symbol you added has returned to normal display.
<i>Save and Label the Diagram:</i>	8	From the File menu, select Save .
	9	Type the diagram label “Diagramming Technique.”
	10	Click OK. The diagram label appears in the window title bar. (See Figure 5-3.)

Note

- The only difference between saving a new diagram and saving an existing one is that you have to give the new diagram a name in the dialog box that is

displayed. The only restrictions on diagram labels are that they cannot exceed 128 characters and that they must be unique within the diagram type of the project. To change a diagram's name select Save With New Name from the File menu. After that, the process is identical to that described for a new diagram, above.

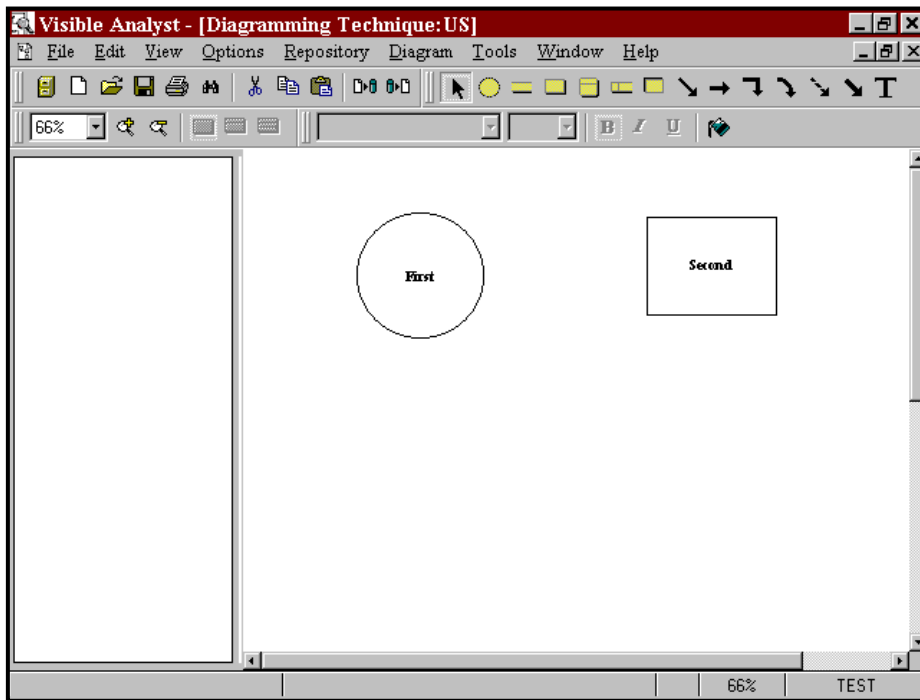



Figure 5-3 New Diagram with Symbols Added

Stylizing a Symbol

- | | | |
|------------------------------------|---|--|
| <i>Change into Selection Mode:</i> | 1 | Click the  button on the control bar or press the ESC key. This changes the cursor, indicating that Visible Analyst is now in selection mode. |
| <i>Use the Object Menu:</i> | 2 | Position the cursor over the symbol labeled First and click on it with the <i>right</i> mouse button. A menu appears with functions that can be performed on the symbol. |
| | 3 | Select Stylize. |

- Stylize the Symbol:*
- 4 In the dialog box, adjust the level of boldness by double-clicking the right-hand arrow on the scroll bar under the word Boldness.
 - 5 Click the Apply button. The symbol in the box indicates how your symbol looks. (See Figure 5-4.)

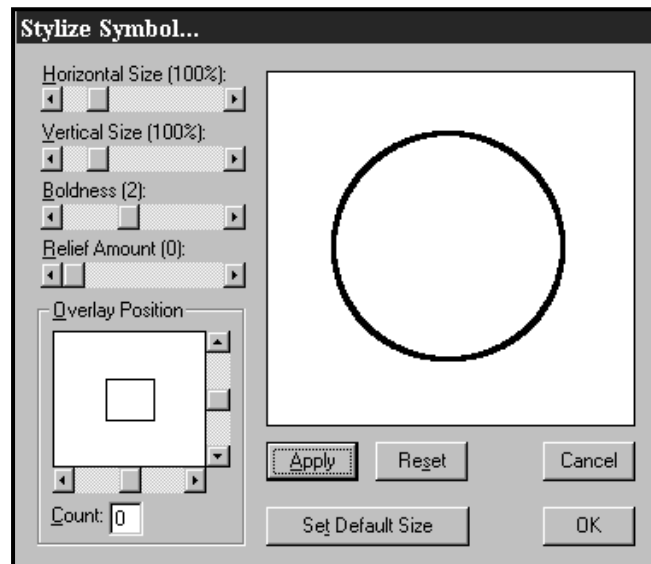


Figure 5-4 Stylize Symbol Dialog Box

- 6 Click the OK button and the stylization you selected is applied to the symbol on the diagram.

Moving, Cutting, and Pasting a Symbol

- Select the Symbol:*
- 1 Position the cursor inside the symbol Second and click the *left* mouse button. The symbol changes color to show that it is now the selected or current object.
- Move the Symbol:*
- 2 Position the cursor inside the symbol Second and click and hold the *left* mouse button. Move the symbol by dragging the box around. A rectangular outline, called the “bounding box,” appears in place of the symbol. Release the mouse button when your symbol is where you want it

or press the ESC key if you want to cancel the move operation.

Cut and Paste the Symbol:

- 3 While the symbol is selected, click on the **Edit** menu.
- 4 Select **Cut**. The symbol disappears from the diagram, but is saved on the Windows Clipboard.
- 5 Go back to the **Edit** menu and select **Paste**. The symbol is displayed surrounded by a dashed outline, indicating the symbol is the current object. Position the cursor within the outline of the symbol, hold the *left* mouse button down and drag it to the desired location on the diagram. Release the mouse button.

Deselect the Symbol:

- 6 Click on an empty space on the diagram with the *left* mouse button. This deselects the highlighted current object.

Notes

- ☐ You can use the Windows keyboard shortcuts for the editing functions to speed up these operations and to edit in dialog boxes.
- ☐ When selecting or changing line types and line terminator choices, Visible Analyst performs differently depending on what state it was in when the modifications were entered. When no diagram is selected and the line types are changed, the default choices are modified. When a diagram is selected but no line is highlighted, the choices remain in effect for the diagram. If a line is selected, the change only impacts the selected line.

Adding Lines to a Diagram

Now add a line to connect the two symbols you have drawn.

Turn on Auto Label Lines:

- 1 Open the **Options** menu. There should be a check mark next to the selection **Auto Label Lines**. This indicates that you are automatically prompted to label a line as soon as it is drawn. If there is no check mark next to the selection, set the option by clicking on the selection.

Set Line-Drawing Mode:

- 2 Click the first line button in the diagram tools tool bar to put Visible Analyst in line-drawing mode. The cursor changes to indicate this.

- Draw the Line:*
- 3 Position the cursor on the edge of the symbol labeled First that is nearest to the symbol labeled Second.
 - 4 Press and hold the *left* mouse button.
 - 5 Drag the line to the edge of symbol Second. The way the line stretches between the cursor and the start-point is sometimes called “rubber-banding.”
 - 6 Release the mouse button to signal the end of the line. If you release the mouse button within the symbol, the line is connected automatically to the edge of the symbol. When the line is completed, it changes color and *handles* appear at the endpoints. (See Figure 5-5.) A dialog box appears for labeling the line.

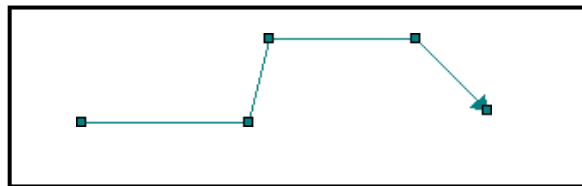


Figure 5-5 A Line With Its Handles


- Label the Line:*
- 7 Type “Flowname.”
 - 8 Click OK to draw the label next to the line on the diagram.

Now that you know how to add a line to a diagram, you can adjust the position and appearance of that line.

Note

- If you want to move the name of a line, select the name by positioning the cursor on the text and press and hold the *left* mouse button. Drag the label to the desired position and then release the mouse button.

Selecting and Adjusting Lines

- Return to Selection Mode:*
- 1 Click the  button on the control bar or press the ESC key.
- Select the Line:*
- 2 If the line is not currently highlighted, click on any point



along the line. When a line is selected, you can see its *handles*, little boxes at the end of each segment that allow you to move the segments by dragging the handles with the mouse.

- | | | |
|----------------------------------|---|---|
| <i>Set Line Characteristics:</i> | 3 | From the Options menu select Line Settings. |
| | 4 | Choose Single Dashed for Line Type. |
| | 5 | Click OK. The line is redrawn using the new type. |

Changing Line Settings as above allows you to adjust the line characteristics for the selected line. If no line is selected, you choose the characteristics for the next line you draw.

Adding Caption Text to a Diagram

You can add text in the form of a title or a paragraph. This text is used to enhance the definition of your diagram or its parts. When entering the text, press ENTER to continue the text on another line.

- | | | |
|-------------------------------------|---|---|
| <i>Set Caption Text Mode:</i> | 1 | Click on the large T (text) button on the control bar. |
| | 2 | Position your cursor at the top of the diagram and click with the <i>left</i> mouse button. |
| | 3 | Type "Unstructured Diagram #1." Then press ENTER to move the cursor down to the next line. Type "Diagram Drawing Techniques." |
| <i>Select the Caption Position:</i> | 4 | Click OK. |
| | 5 | Click the  button on the control bar or press the  key to return to editing mode. |
| | 6 | Click the right mouse button over the caption you just added to display the Object menu for the caption. |
| | 7 | Select Text Settings from the Object menu. |
| | 8 | Select Times New Roman in the box labeled Typeface. Refer to Figure 5-6. |
-
- | | | |
|--|---|--|
| <i>Change the Caption Characteristics:</i> | 6 | Click the right mouse button over the caption you just added to display the Object menu for the caption. |
| | 7 | Select Text Settings from the Object menu. |
| | 8 | Select Times New Roman in the box labeled Typeface. Refer to Figure 5-6. |

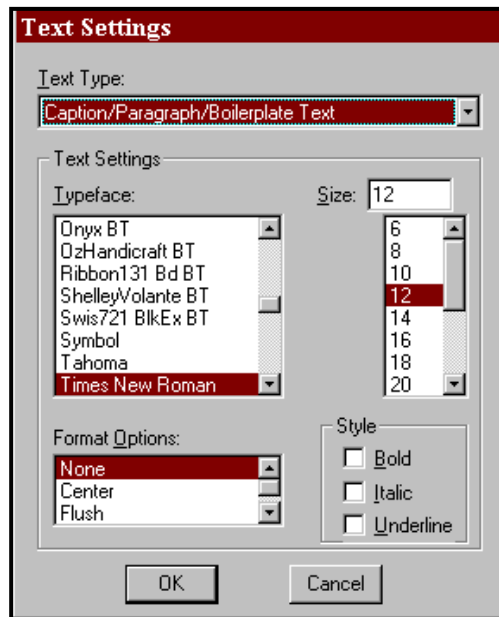


Figure 5-6 Text Settings Dialog Box

- 9 Change the Point Size to 16 in the Size box.
- 10 Select Bold in the box labeled Style.
- 11 Select Center in the box labeled Format Options.
- 12 Click the OK button and then deselect the text. The completed diagram should appear more or less like that shown in Figure 5-7.

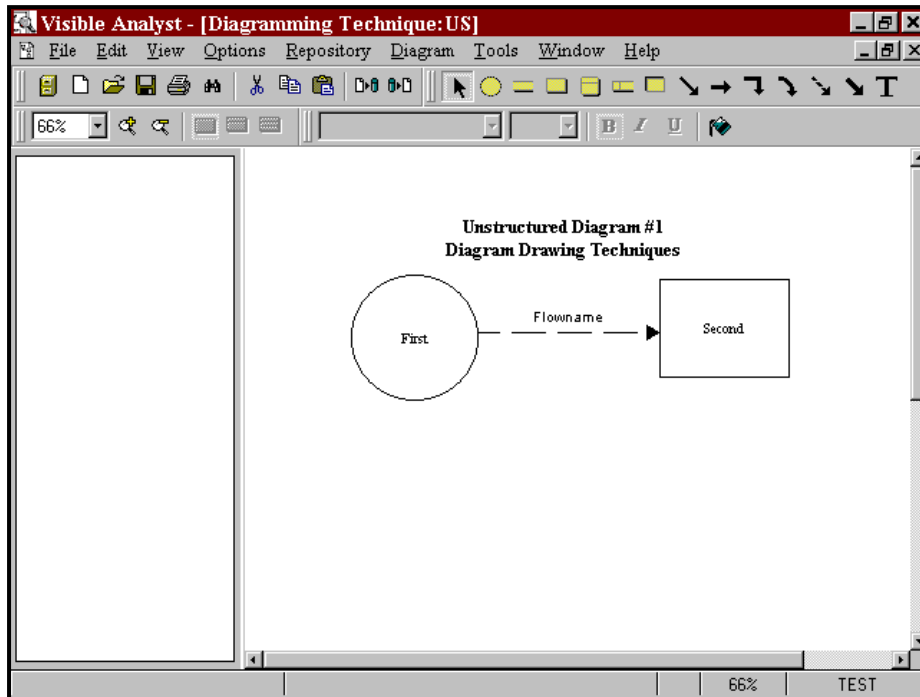


Figure 5-7 The Completed Diagram

Note

- The “T” icon text should not be used to define symbols or lines. Only symbol labels are entered into the repository for rules based components. To label an unlabeled diagram object, click the object with the *right* mouse button and choose Change Item from the object menu.

OTHER DIAGRAMMING FUNCTIONS

Now we take a look at some of the other functions available to help you create Visible Analyst diagrams.

Colors

Different screen objects displayed in different colors makes it easier to distinguish them on the screen. You have a number of choices available that you can experiment with to find a pleasing combination.

- | | | |
|--------------------------|---|--|
| <i>Open the Menu:</i> | 1 | To change the colors of your symbols, lines, and text select Colors from the Options menu. |
| <i>Change the Color:</i> | 2 | Under Object Type, select Symbol Color. Select a color by clicking on one of the color squares or by adjusting the slide bars. |
| | 3 | Click OK. If no objects are selected, the default colors are set. If objects are selected, only those items are changed. |

Displaying and Hiding Symbol Labels

It is sometimes easier to see the overall layout of objects on a diagram if there are no text labels distracting your attention from the structure the diagram represents. Visible Analyst allows you to hide the labels of symbols and lines if you wish to do so.

- | | | |
|------------------------------|---|--|
| <i>Hide the Labels:</i> | 1 | From the View menu, click Show Symbol Names . The symbol labels should disappear. (A check mark in front of this selection indicates that the symbol names are shown; otherwise they are not shown.) |
| <i>Redisplay the Labels:</i> | 2 | From the View menu, select Show Symbol Names again to reset the names to show. |

Note

- Turning line or symbol labels off is *not* the same as not labeling them. A line or symbol that has never been labeled *does not exist* as far as the repository is concerned.

Changing Text Characteristics for a Block of Diagram Objects

- | | | |
|-----------------------------------|---|--|
| <i>Select a Group of Objects:</i> | 1 | Draw a box around the symbols on the diagram. Place the cursor in the upper left corner of the diagram and hold the <i>left</i> mouse button down while you drag the mouse to the lower right corner of the diagram. A bounding box rectangle is created as you drag the mouse. After you release the button, all items <i>completely</i> inside the bounding box are highlighted. |
| <i>Change the Text:</i> | 2 | From the Options menu select Text Settings . |

- 3 Choose Symbol Labels in the box marked Text Type.
- 4 Choose a Typeface and Point Size.
- 5 Return to the Text Type box and choose Line Labels.
- 6 Choose a Typeface and Point Size.
- 7 Click OK.
- 8 Click in an empty area outside of the bounding box to deselect it.

The symbol labels and line labels for the items *completely* contained in the box change to the new text settings.

Note

- The text values of *all* items in the bounding box are set by this function. Those text types that you do not *explicitly* set revert to the default values shown in the dialog box.

CLOSING A DIAGRAM

To close a diagram:

Activate the Control Menu:

- 1 Click on the *diagram* **Control** menu button (not the *Analyst Control* menu button) in the top left corner of the *diagram* window, or to the left of the **File** menu when the diagram is maximized. There is also a Close Diagram function on the **File** menu.
- 2 Select **Close**. If your diagram has not been maximized, meaning that it occupies less than the entire Visible Analyst workspace, you can close the diagram by double-clicking on the **Control** menu button.
- 3 Visible Analyst prompts you to Save the diagram. Click Yes to close the diagram. Selecting No closes the diagram without saving any changes made since the last Save operation was performed.

THE TUTORIAL PROJECT

For the rest of this tutorial, you add diagrams to an existing project. We created the project to save you the time it would take to enter the repository information and create the diagrams that are necessary to demonstrate some of the more advanced features of Visible Analyst. To access the TEST project:

- 1 Choose **Select Project** from the **File** menu, or click the **File Cabinet** button on the control bar.
- 2 Select **TEST** from the list displayed on the **Select Project** dialog box and click **OK**.

TEST is now the current project. As when you created the first project, the lower right corner of the Visible Analyst screen displays the name of the current project.

CONCLUSION

Now that you understand the basic methods for drawing symbols, lines, and text in Visible Analyst, as well as how to change some of the optional settings, you are ready to build more significant diagrams.

We have provided diagrams to help demonstrate some of the structured modeling capabilities of Visible Analyst. The objects on the diagrams and entries in the repository have been filled in for you.

Lesson

COMPONENT DIAGRAMS

OVERVIEW

Component diagrams allow you to show the structural relationships between system components. According to the UML 2.x specification, components are "autonomous, encapsulated units within a system or subsystem that provide one or more interfaces".

Although the specification does not strictly state it, components are larger design units that represent things that will typically be implemented using replaceable modules. Components are strictly logical, design-time constructs that can easily be re-used. By providing a high level view of the system components, the component diagrams provide an easily understood overview of the system to developers, analysts and administrators of the system.

Component diagrams can contain components, classes, interfaces and relationships. You can describe the components you are modeling and the relationships between them by drawing them onto a diagram. Each diagram or view can show an arbitrarily large or small part of your component model. You can show multiple views of your component model by including different combinations of components in the repository.

All information you place on a component diagram is, of course, captured by the repository and is available to your class model, your process model (data flow diagrams), your structure charts, and your data model (entity relationship diagrams), where applicable. The Analyze function can assist you in determining any syntax or definition problems with your object model.

COMPONENT DIAGRAM SYMBOLS

The component symbol is the primary graphic used when creating a component model. When a component is added to a diagram, an entry is created in the repository so that additional information can be specified regarding the component, in order to complete its definition. The class symbol is also used on a component diagram.

Each component is represented by a rectangle with an icon in the upper right corner as shown below.

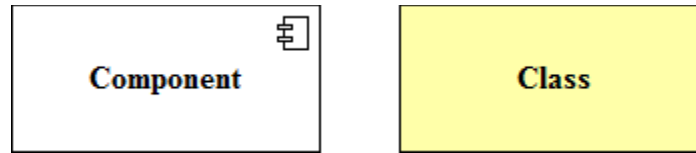


Figure 16-1 Component and Class Symbols

If you want to show the internal structure of a component, use the **Explode** function on the right mouse click object menu, or select **Nest** on the **File** menu. This allows you to create a new diagram with an outline of the component in which you can add the internal elements. These internal elements, consisting of classes with their methods and attributes expose the “black-box” properties and operational details of the component object.

Because component symbols are container objects, similar to Pools and System Boundaries, they have no default color assigned. Select a component symbol and choose **Colors** from the **Options** menu to choose a color for the component.

The Visible Analyst allows users to include Note symbols on a component diagram or within a component object. Notes do not maintain a repository entry and are only used as a textual reference on a diagram. Use the Note Link line type on the **Control Bar** to link the note to a component or class.

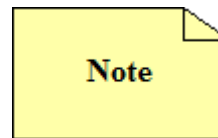


Figure 16-2 Note Symbol

INTERFACE LINES

An interface represents the formal contract of services a component provides to its consumers/clients. These interfaces are the basis for “wiring” the components together as explained in the UML 2.x specification, and Visible Analyst uses the UML 2.x notation for defining an interface. A line with a complete circle at its end (lollipop) represents an interface that the component provides. Interface lines with only a half circle at their end (socket) represent an interface that the component requires. In both cases, the name of the interface is placed above the interface line.

Interface lines are only displayed on a diagram and do not maintain an accessible repository entry. Interfaces that are attached to a component can be included when generating a component report.

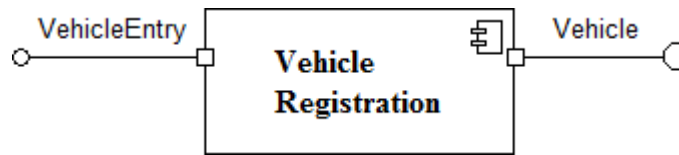


Figure 16-3 Component Symbol with Provided (left) and Required (right) Interface Lines

There are several ways to add an interface to a component:

- Right-click on a component and choose either *Add Provided Interface* or *Add Required Interface* from the context menu.
- Click on an interface icon on the control bar and then click within a component; the interface will be drawn automatically. For a provided interface click near the left side to add to the left or near the top to add to the top. For a required interface, click near the right side or bottom.
- Click on an interface icon on the control bar and then press and hold the left mouse button down where you want the interface to begin and then drag the mouse until it is inside the component symbol and release the button.

When labeling the interface, you can choose to show a port which provides a way to model how a component's provided/required interfaces relate to its internal parts. A port is displayed as a square at the component end of an interface.

You can connect the required interface of one component to the provided interface of another by pressing the left mouse button on the endpoint of the interface and dragging it until it connects with the endpoint of another interface.

The UML 2.x specification describes the assembly and a delegation connector types as follows:

Assembly Connections

An *assembly* connector is a connector between two components that defines that one component provides the services that another component requires. The assembly connector defines the connection *from* a required interface or port *to* a provided interface or port. This Assembly Connection is drawn with the Provided Interface line (with the lollipop) connected to a Required Interface line (with a socket).

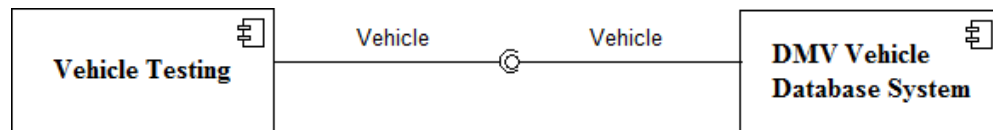


Figure 16-4 Example Assembly Connector between 2 Component Objects

Delegation Connection

A *delegation* connector is a connector that links the external contract of a component (as specified by its ports) to the internal realization of that behavior by the component's parts. It represents the forwarding of signals (operation requests and events): a signal that arrives at a port that has a delegation connector to a part or to another port will be passed on to that target for handling. The line is drawn from the port to the class or component as shown in Figure 16-5. Delegation lines are not normally named, but are recognized by the Visible Analyst Rules when the diagram is analyzed.

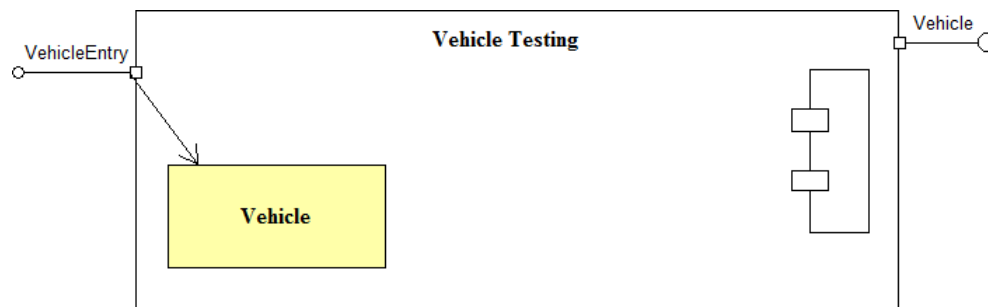


Figure 16-5 A Delegation Connector from the port of VehicleEntry to Vehicle

Dependency Line

A *dependency* relationship line is a dashed line with an open arrowhead as shown in figure 16-6. The dependency relationship is used to connect components or classes, and signifies that a single or a set of model elements requires other model elements for their specification or implementation. It is not a direction of a process but a direction of a relationship. Within a component object, use the dependency relationship to relate classes or components. In the example below, the Inspection Station relies on the information from the Vehicle class. While a dependency line may have a label associated with the line, no editable repository entry is created for these line types. The dependency lines are recognized by the Visible Analyst Rules when the diagram is analyzed.

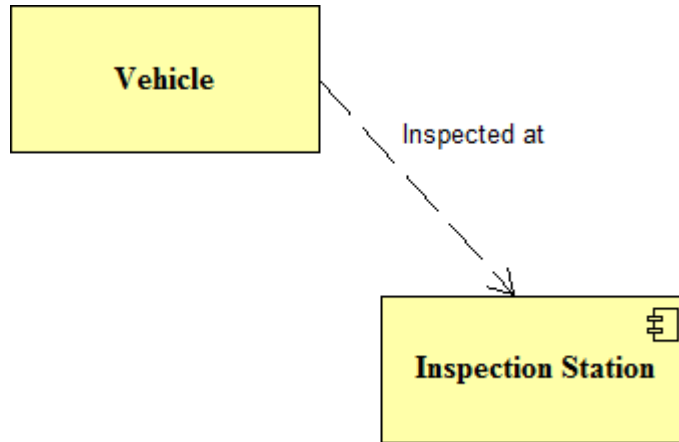


Figure 16-6 Dependency Relationship Line

COMPONENT INTERNAL STRUCTURE

The internal structure of a component or “white-box” view of the component as referenced in the UML 2.x specification can be drawn on a linked “child” diagram in the Visible Analyst. This parent-child linkage is similar to the diagram linkage when exploding a data flow process to create a child diagram. Each component can be “exploded” to create a new component diagram to display additional internal structure detail.

When a component symbol is exploded, all interfaces connected to the “parent” component are attached to the expanded component symbol on the child diagram, as shown in Figure 16-7. When the child diagram is saved, you are prompted to create a **Nest** relationship between the new interface component diagram and parent component symbol.

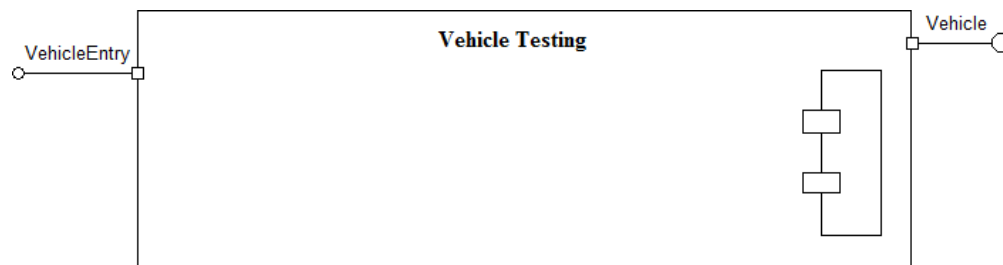


Figure 16-7 Exploded Component Symbol with Attached Connectors

THE DMV COMPONENT SCENARIO

Many states and countries periodically require vehicle inspections for safety and environmental reasons. The owner of the vehicle is required to bring the vehicle to a certified inspection station, where a certified inspector will evaluate the safety and environmental features of the vehicle. If any deficiencies are found in either test, the vehicle fails the inspection. In most cases, the owner is allowed to return for a follow up inspection after the deficiencies have been corrected. Once the vehicle passes the tests, an inspection sticker is issued for the vehicle indicating that the vehicle passed the inspection. The vehicle details and results of the inspection are then communicated to the Department of Motor Vehicles to be recorded in the DMV database after each vehicles inspection.

Drawing the Component Diagram

- | | | |
|---------------------------------|---|---|
| <i>Set the Zoom Level:</i> | 1 | Set the zoom level to 66% from the View menu. |
| <i>Open a new:</i> | 2 | Select New Diagram from the File menu, choose |
| <i>Component as the diagram</i> | | Type and click OK. |
| <i>Diagram:</i> | 3 | Click the first icon in the Control Bar, component, add a component symbol to the diagram, and label the component "Vehicle Testing". Add a second component symbol to the right of "Vehicle Testing" and label it "DMV Vehicle Database System". Click the Esc key to exit drawing mode. |

Add the Interfaces: 4 Right mouse click on the component “Vehicle Testing” and choose the **Add Provided Interface** option. The interface is automatically drawn attached to the left side of the component symbol. Label the interface “VehicleEntry”. Confirm that the **Show Port** option is checked.

Right mouse click on the “Vehicle Testing” symbol again, but choose **Add Required Interface** to draw the interface line attached to the right side of the symbol. Label the interface “Vehicles”.

Add a *provided* interface to the component object “DMV Vehicle Database System” and label the interface “Vehicles”

NOTE: You can select the line types from the **Control Bar** and manually draw the interface lines connected to the component symbol. The interface lines can be moved to any position on the component symbol by dragging the interface line to the selected position.

5 From the **File** menu, select **Save**, and save the diagram with the name **Vehicle Testing Components**”.

Create an Assembly Connection: 6

Left mouse click on the *required* interface connected to the “DMV Vehicle Database System” component so that the line handles, as explained in Chapter 5 (see figure5-5), are displayed. Left mouse click on the left line handle, hold down the mouse button, and drag the line so that the lollipop is within the socket, as shown in figure Y-4.

NOTE: While making the line longer, the line may not be drawn completely horizontal. While drawing the line, the line becomes dashed. While the line is dashed, click the “S” key on the keyboard to Snap the line and make it horizontal.

Analyze the Diagram: 7 Select **Analyze** from the **Diagram** menu to check the diagram for the correct syntax and click OK. The error message displayed points out that no required interface was attached to the component “DMV Vehicle Database System”.

To correct this error, right mouse click on the “DMV Vehicle Database System” component, choose **Add**

Required Interface, and label the interface “VehicleControl”.

Save the diagram and run the **Analyze** feature again to confirm that the diagram is correct.

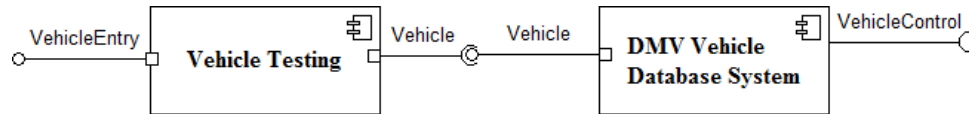


Figure 16-8 Completed Top-Level Component Diagram

The DMV Vehicle Testing system is composed of both safety and environmental components. These internal component objects may already exist as classes or other component objects in the project, or they can be added directly onto the new component diagram. The first action is to explode the “Vehicle Testing” component and create the child diagram.

- | | | |
|-----------------------------------|---|---|
| <i>Explode a Component:</i> | 1 | <p>Right mouse click on the component “Vehicle Testing” and choose</p> <p>Explode. Click the Create New Diagram button to create the new component diagram.</p> <p>NOTE: The Explode option is also available by selecting Nest from the File menu.</p> |
| <i>Add Internal Structures:</i> | 2 | <p>Click the first symbol icon and add the component “Safety Inspection” inside the “Vehicle Testing” symbol, adding the symbol towards the top of the component.</p> <p>Add a second component symbol below the first symbol, and label it “Environmental Inspection”.</p> <p>Add a Provider interface labeled “VehicleEntry” and a Required Interface labeled “Vehicles” to both component symbols.</p> |
| <i>Add Classes as Structures:</i> | 3 | <p>Click the class icon, the second symbol icon, and add a class symbol <i>between</i> the component symbols, labeled as “Vehicle”.</p> |
| <i>Save the Diagram:</i> | 4 | <p>From the File menu, select Save, and save the diagram with the name “Vehicle Testing Internal Components”.</p> |
-

- Add Delegate Lines:* 5 Click the Delegate line icon, the third line icon, and draw a delegate line from the “VehicleEntry” interface port to the component “Safety Inspection” provider interface lollipop. Draw a second line from the “VehicleEntry” interface port to the “Environment Inspection” interface lollipop.
- Draw delegate lines from the required interface sockets attached to the “Environmental Inspection” and Safety Inspections” components to the “Vehicle” required interfaces port.
- NOTE: Delegation lines indicate that messages and signals flow from the interface to the internal components.
- Add a Dependency Line* 6 Click the Dependency line icon, the fourth line icon, and draw two dependency lines from the class “Vehicle” to the component “Safety Inspection” “and to the component “Environmental Inspection”. Label both lines” VehicleDetails”.
- NOTE: These dependency lines indicate that the components depend on an element or group of elements listed in the class.
- Analyze the Diagram:* 6 Select **Analyze** from the **Diagram** menu to check the diagram for the correct syntax and click OK.
-

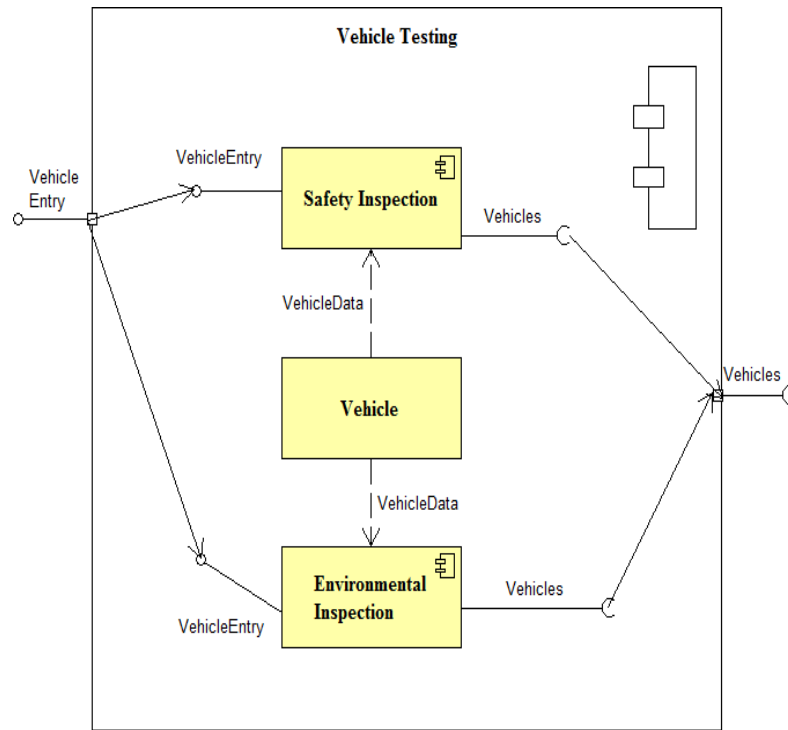


Figure 16-9 The Completed Internal Structure Component Diagram

