Visible® Analyst Tutorial

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

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

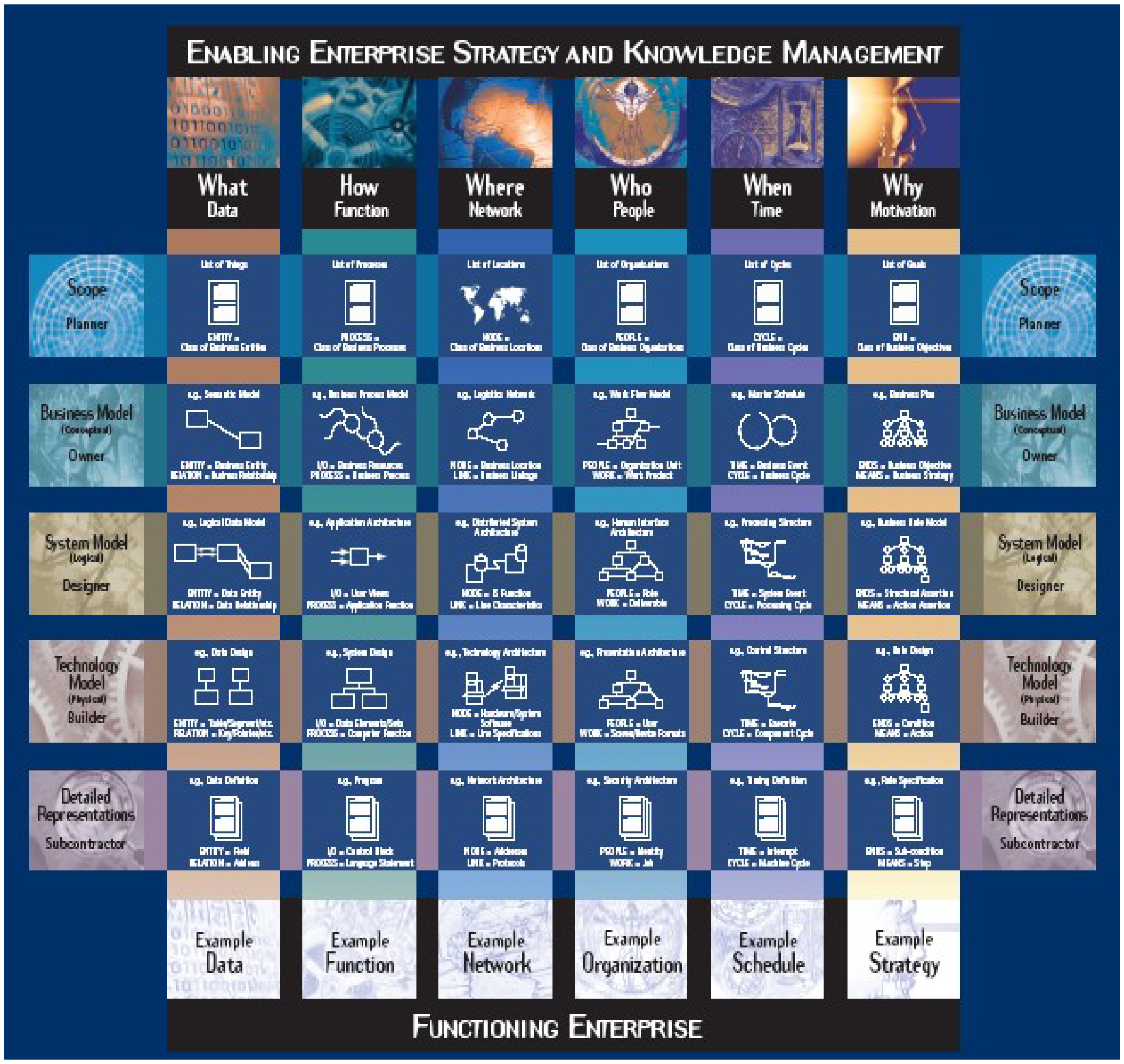
Email: contact@visiblesystemscorp.com

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

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. And, 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).

Collaboration Diagramming

OVERVIEW

UML includes specifications for two forms of interaction diagrams: sequence diagrams and collaboration diagrams. Both diagrams present the objects participating in a business scenario and show the messages sent and received.

The sequence diagram uses _lifelines,‘ parallel bars drawn below the objects, so that the sequence of messages sent and received can be understood by looking at the diagram from top to bottom. On the other hand, in a collaboration diagram the objects are arranged so that the basic relationships are highlighted; and sequence numbers are used to order the messages sent and received.

A collaboration diagram consists of a set of objects that together carry out a scenario, the links between the objects, and details concerning the messages sent and received. A collaboration diagram can be drawn at the class level or at the instance level.

- At the class level, it shows the associations or relationships between the different classes.
- At the instance level, it shows the links or messages that are passed between the instances.

DEFINITIONS

The important diagram constructs in collaboration diagrams include:

Object	An object appears in as a plain rectangle with an underlined title. It represents an entity with a well-defined boundary and identity that encapsulates state and behavior. A collaboration diagram may be populated with objects representing different classes, as well as objects representing specific instances in a class.
Note	A note appears as a rectangle with the top right corner folded over. A note is used to record descriptive text that appears on the diagram.

Object Link	An object link appears as a solid line connecting two objects and represents the fact that there is a relationship between the two objects. Visible Analyst automatically adds an object link (as part of the task) when a message is drawn between two objects.
Procedure Call	A procedure call is a message between two objects, appearing as a filled solid arrow. The target object (at the arrowhead end) must complete its task before the calling process can continue.
Flat Flow of Control	Flat flow of control is a message between two objects, appearing as a stick arrowhead and signifying the passing of control from the originating object to the target object.
Asynchronous Stimulus	Asynchronous stimulus is a message between two objects, appearing as a half-stick arrowhead, and used instead of a stick arrowhead to show an asynchronous communication between two objects in a procedural sequence.
Return	A return is a message between two objects, appearing as a dashed arrow with a stick arrowhead, and represents a return from a procedure call.
Self-Delegation	Self-delegation is a message from an object to itself, appearing as a recursive arrow.
Note Link	A note link appears as a dotted line connecting an object with a note.

DEVELOPING YOUR COLLABORATION DIAGRAM

Describing Scenarios using a Collaboration Diagram

The business analyst creates a collaboration diagram to explore the following questions:

- What objects are included in the scenario?
- What messages are sent and received?
- What is the sequence of the messages?

The objects that are included in a scenario are typically part of the enterprise model. For example, an object introduced to the repository using the class diagram can certainly be reused in a collaboration diagram. See Lesson 10, The Class Diagrams.

Object Instances Versus Object Classes

The objects appearing in a collaboration diagram may represent object instances or object classes. The way in which an object is identified determines if the object is an instance or a class. When the name is specified, it means this object represents a particular object instance. For example _John Smith:: Applicant‘ represents the fact that _John Smith‘ is a member of the _Applicant‘ class. If no object name is specified, the object represents the class.

Note the object class must always be specified. The object identifier is separated into two parts using a double colon (::); the first part specifies the name, the second part the class.

Object Methods

The messages sent to and from an object must _fit‘ the object, or correspond with its methods. Only methods from the derivation tree of the target (the one receiving the message) object's class can be used. All available methods are displayed in the drop-down list. If you want to create a new method, click the New Method button.

If the method has arguments, you can specify values for the arguments by clicking the Values button. By default the name and type for the method are displayed. If you want to change the argument list of the method, click the Change Arguments button.

Note The degree to which you can change method or message attributes depends on your rights to the target object's class and the interaction diagram settings.

Object Links

A Label Message dialog box appears when an object link is drawn between two objects. You must then supply details concerning the messages. If you are not ready to define these details (or wish to define them a later), you can delete the message icons, and retain the object link as a solid line.

Messages

Messages are added to the collaboration diagram to describe the way in which the objects will work together.

The following information is maintained for each message:

- **To and From.** The name of the target object and source object. This can be switched by clicking the reverse button.
- **Type.** The type of message, either asynchronous stimulus, flat flow of control, or procedure call.
- **Occurs Multiple Times.** Indicates the message will be called more than once. If this option is selected, an asterisk will appear next to the message name on the diagram.

- **Guard Condition.** Specify the guard condition that controls the firing of the message. This is a free-form text field. A guard condition is a logical expression that evaluates to TRUE or FALSE, and must be satisfied before the message can be sent.
- **Sequence Number.** Indicates the order of messages. This can be either a single numeric value such as 1, 2, or 3, or a decimal such has 1.2, or 1.1.4. This option is only available on collaboration diagrams, since sequence diagrams by their very nature indicate message ordering.

DEPARTMENT OF MOTOR VEHICLES SCENARIO

The collaboration diagram example is based on the following scenario:

- The registrar logs onto the system and selects the driver registration window.
- By selecting the option New Applicant, an Applicant object appears. Details concerning the applicant‘s name, address, and phone number are recorded.
- The registrar verifies that the applicant possesses an insurance certificate, and if yes, records the coverage limits. The registrar also verifies that the applicant has a learner‘s permit. Once these checks are made the applicant is considered _Valid‘ and is ready for the road test.
- If the applicant passes the road test, a driver‘s license is issued.

A completed collaboration diagram is shown in Figure 15-1.

Adding Objects to a View

The basic building block of the collaboration diagram is the _object‘. The following steps are taken to establish a new collaboration diagram and create the objects.

Set the Zoom Level:	1	From the View menu, select 66% zoom so 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 Collaboration Diagram.
	4	Select Standard Workspace and Portrait Orientation.
	5	Click OK.
Add Objects:	6	Click on the first symbol button, the rectangle, on the control bar. This represents an object.
	7	Place the cursor inside the diagram workspace and click the left mouse button.

	8	Label the object, leaving the name field blank, and the class field —Applicant‖.
	9	Add the rest of the objects as shown in Figure 15-1.
Save the Diagram:	10	Save the Collaboration Diagram, and give it the name _Driver Registration‘.

Adding Relationships to a Collaboration Model

The relationships in a collaboration diagram appear as object links. Messages to and from the objects are added to the object links.

This procedure was written with Auto Label Lines turned on; thus message details must be added when an object link is drawn.

Add Object Links:	1	Select the object link from the control bar and make a connection from _Driver Registration Window‘ to _Applicant‘.
Add Message to the Object Link:	2	Select the method that is associated with the target object; or if no method exists, add a method. Add the method —new‖.
	3	Select the message type _Flat Flow of Control‘.
	4	Leave the guard condition blank.
	5	Enter the sequence number _1.1‘.
	6	Continue adding messages until the model is complete, as shown in Figure 15-1.
Save:	7	Select Save from the File menu.

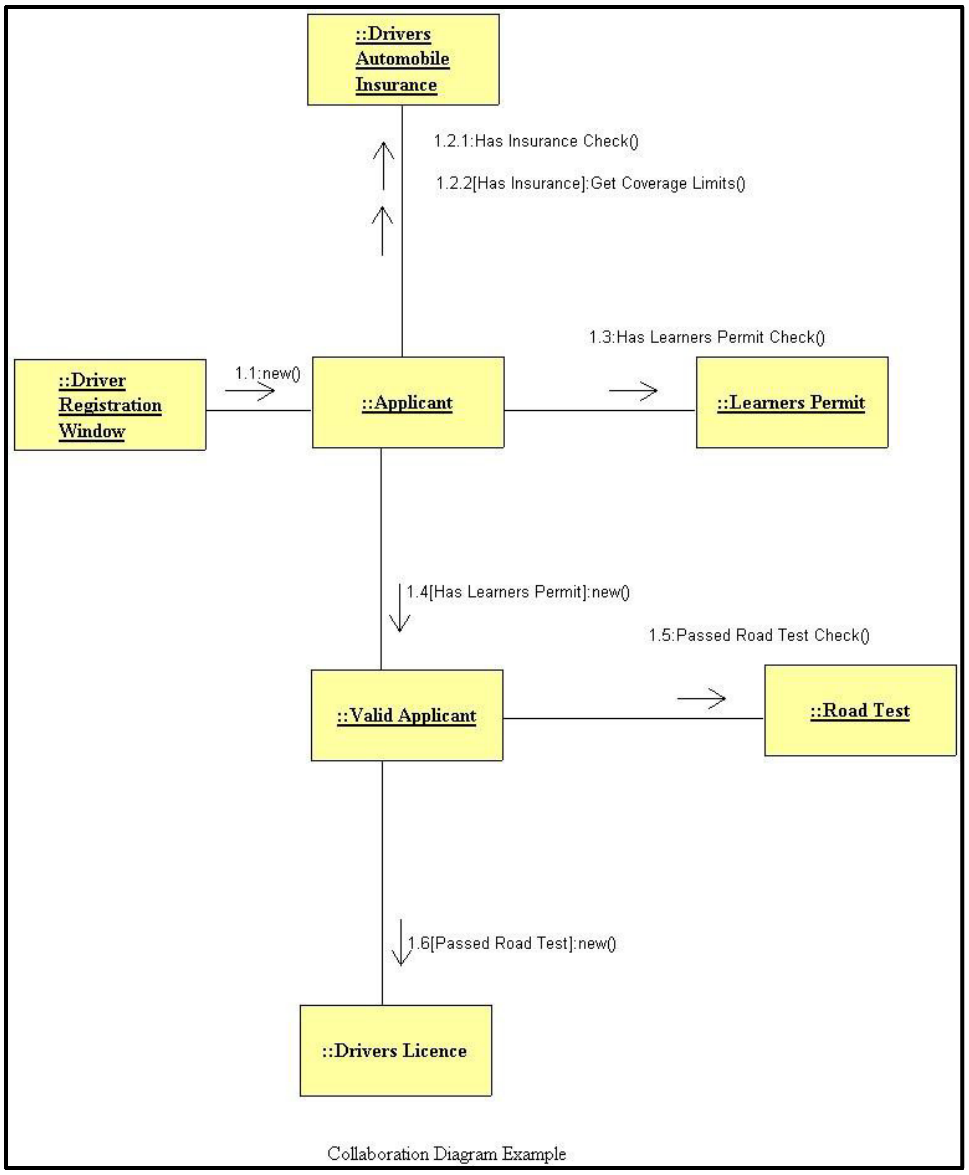

::Drivers Automobile Insurance
1.2.1:Has Insurance Check()
1.2.2[Has Insurance]:Get Coverage Limits()
1.3:Has Learners Permit Check()
::Driver Registration Window
1.1:new()
::Applicant
::Learners Permit
1.4[Has Learners Permit]:new()
1.5:Passed Road Test Check()
::Valid Applicant
::Road Test
1.6[Passed Road Test]:new()
::Drivers Licence
Collaboration Diagram Example

Figure 15-1 Collaboration Diagram Example