Visible® Analyst Tutorial

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

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

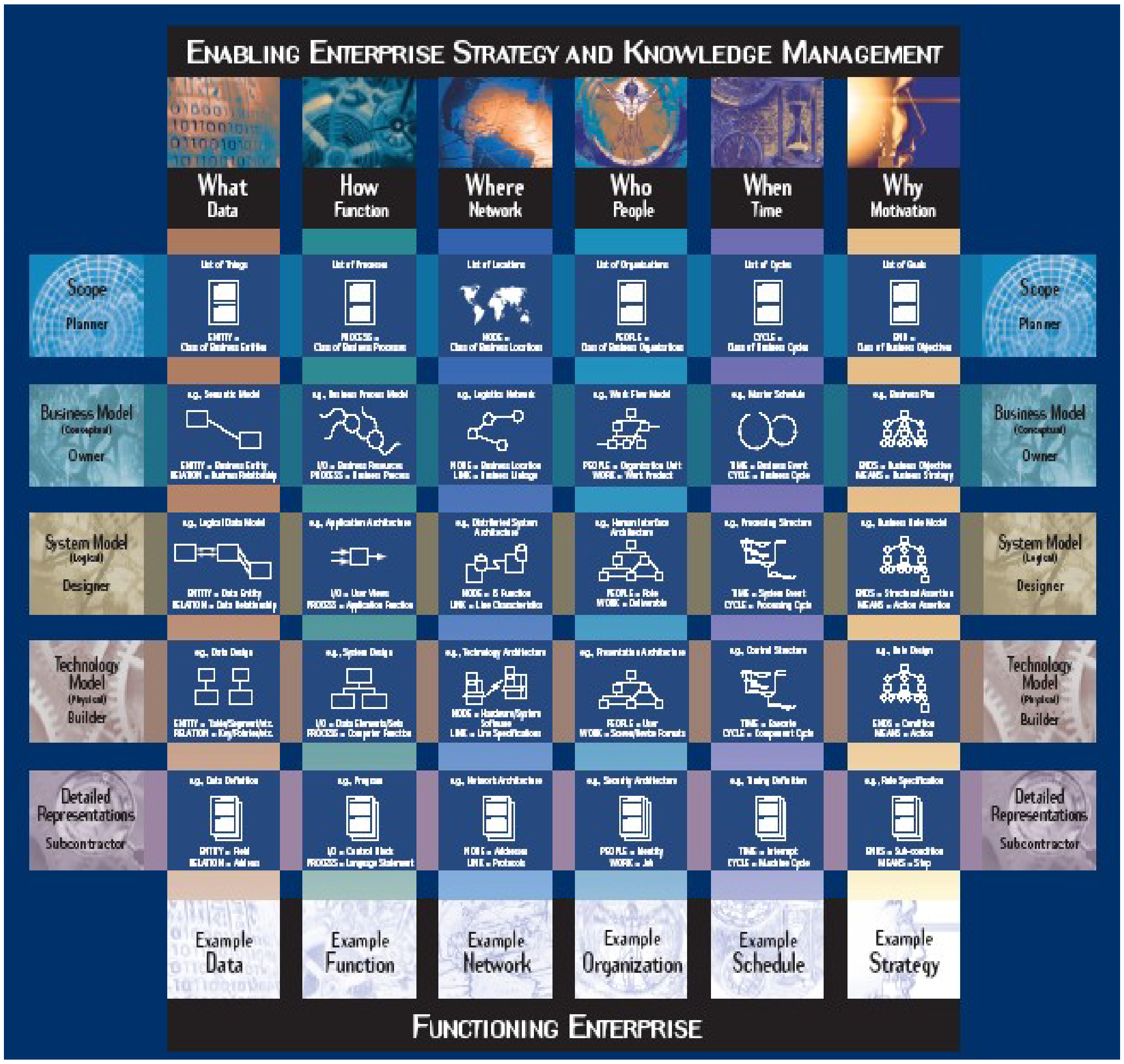
Email: contact@visiblesystemscorp.com

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).

Sequence Diagramming

OVERVIEW

A sequence diagram is a type of interaction diagram. Interaction diagrams describe how groups of objects interact and collaborate in performing a behavior. There are two types of interaction diagrams that basically model the same information: sequence diagrams and collaboration diagrams. In a sequence diagram, the interaction is modeled in time sequence. Generally, an interaction diagram captures the behavior of a single use case. The sequence diagram shows several objects participating in the interaction and the messages that are passed among these objects. The diagram shows the objects by their "lifelines" and the messages that they exchange arranged in time sequence. It does not show the associations among the objects. The associations can be obtained from the complementary collaboration diagram.

A sequence diagram has two dimensions: the vertical dimension represents time and the horizontal dimension represents different objects. Normally time proceeds down the page. (The dimensions may be reversed if desired.) Usually only time sequences are important; but in real-time applications, the time axis could be an actual metric. There is no significance to the horizontal ordering of the objects. Often call arrows are arranged to proceed in one direction across the page; but this is not always possible, and the ordering does not convey information.

DEFINITIONS

The components of the sequence diagramming process include:

Object	An object is defined as an instance of a class. It is drawn as a rectangle with the name of the object and class name inside the rectangle.
Class	A class is a group of objects with the same data structure (attributes) and behavior (operations). A class is an abstraction that describes properties that are important to an application.
Lifeline	The lifeline represents an object‘s life and existence during the time period of the interaction. A dashed vertical line is the symbol of a lifeline. An object symbol is drawn at the top of the lifeline.

Activation	An activation shows the time period during which an object is performing an action. It represents both the duration of the action in time and the control relationship between the activation and its callers. An activation is shown as a tall thin rectangle whose top is aligned with its initiation time and whose bottom is aligned with its completion time. Activation symbols are drawn on the top of an object's lifeline.
Message	A message is a communication from one object to another, usually communicating an order to perform an action. A message is represented by a horizontal solid arrow from the lifeline of one object to the lifeline of another object. The time order in which these messages occur is shown top to bottom on the page. Each message can be labeled with a message name, conditions, return arguments, etc.
Self Call	This is a message that an object sends to itself. It is represented by a message arrow originating at the object lifeline and looping around to end at the same lifeline.
Object Deletion	Objects that are deleted by a message or self-destruct during the time period of the interaction have a large X drawn at the bottom of their lifeline.
Return	A return is a message that is not a new message, but rather a return message from an object to which a new message was previously sent. It is labeled with a dashed line rather than a full line.
Condition	Some messages are sent only when a certain condition is true. In this case, you can label the message with the controlling condition.
Asynchronous Message	An asynchronous message is one that does not stop the caller object from continuing processing.

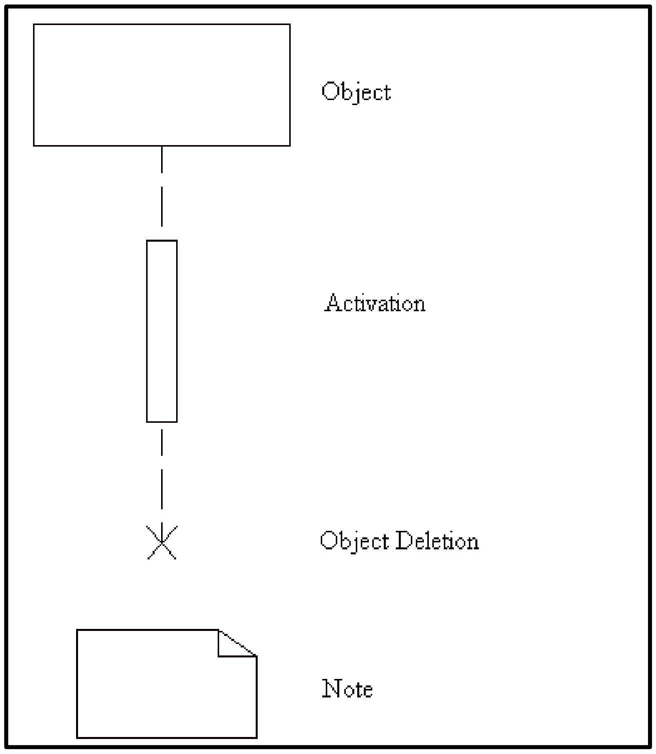


Figure 14-1 Sequence Diagram Symbols

DEVELOPING YOUR SEQUENCE DIAGRAM

The sequence diagram primarily is composed of a group of objects and the messages that are passed between them.

Adding Objects

Objects are the basic building blocks of the sequence diagram. The objects are usually placed horizontally across the page in no particular order, while the vertical axis denotes time sequence. Objects used on the sequence diagram may already exist in the repository, or they may be new objects created during the time period of the interaction.

Each object will have a dashed vertical line under it, representing its lifetime. If the object is created or destroyed during the period of time shown on the diagram, its lifeline starts or stops at the appropriate point. Otherwise, it goes from the top to the bottom of the diagram.

Set the Zoom Level:	1	From the View menu, select 66% zoom so that you can see all of the needed workspace.
Create a New Diagram:	2	From the File menu, select New Diagram.
	3	Select the diagram type Sequence.
	4	Select Standard Workspace.
	5	Click OK.
Add Object:	6	Click the first symbol button, the rectangle, on the control bar. This is the object symbol.
	7	Place the cursor in the top left of the workspace and left-click the mouse. The object is drawn and you are prompted for an object name and class name. Leave the object name field blank. Type —Application Entry Window‖ as the class name. Click OK.
	8	You are prompted to create a new class if it doesn‘t already exist. Click the Yes button.
	9	Add the rest of the objects as shown in the Figure 16-2.
Save the Diagram:	10	Save the sequence diagram with the label —DMV Sequence Diagram‖.

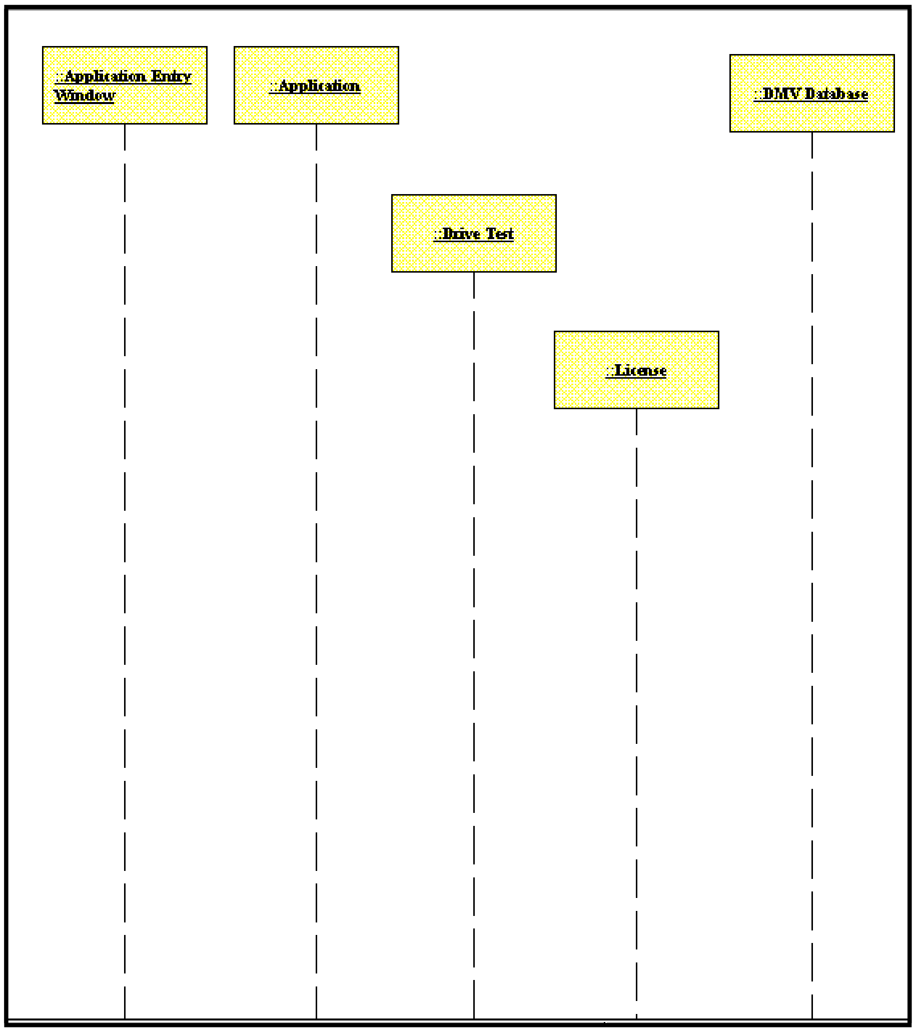


Figure 14-2 Sequence Diagram with Objects

Adding Activation Symbols

An activation represents the time period during which an object is performing an action. It represents both the duration of the action in time and the control relationship between the activation and its callers. An activation is shown as a tall thin rectangle whose top is aligned with its initiation time and whose bottom is aligned with its completion time. The incoming message may indicate the action. In procedural flow of control, the top of the activation symbol is at the tip of an incoming message (the one that initiates the action) and the base of the symbol is at the end of a return message.

In the case of a recursive call to an object with an existing activation, the second activation symbol is drawn slightly to the right of the first one so that they appear to "stack up" visually. Before drawing the second activation symbol, lengthen the size of the original activation symbol by grabbing the symbol handles when the object is highlighted. After drawing the second activation symbol on top of the first symbol, increase the width of the second symbol so that it visually —stacks up‖ over the first symbol.

Add Activation Symbol:	1	Click the second symbol button, the narrow vertical bar, on the control bar. This is the activation symbol.
	2	Place the cursor on the —Application Entry Window‖ object‘s lifeline, and click the left mouse button. An activation symbol is drawn on top of the object‘s lifeline.
	3	Add the other activation bars as shown in the Figure 16-3. You can size the length of the activation symbols by clicking on the rectangle and dragging the edges.
Save the Diagram:	4	Save the sequence diagram.

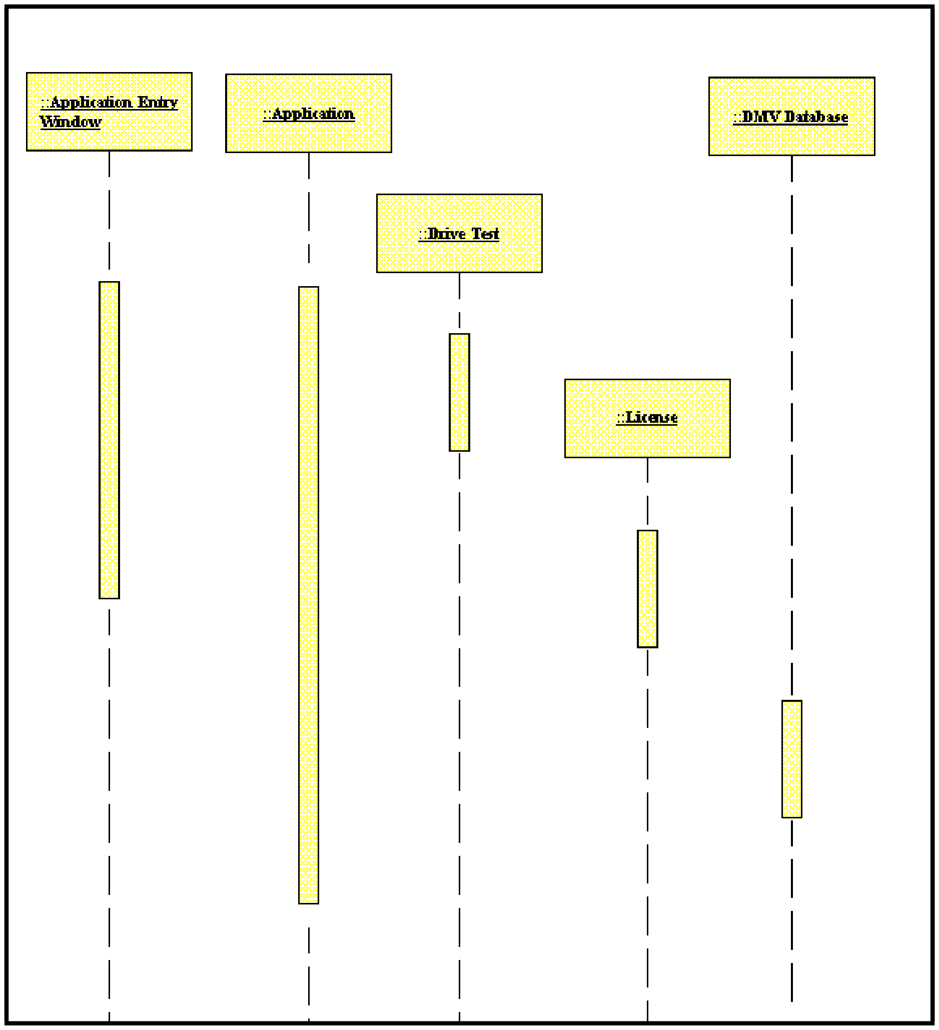


Figure 14-3 Sequence Diagram with Activation Symbols Added

Adding Object Deletion

Objects can be deleted during an interaction during the time represented on a sequence diagram. Objects that are deleted by a message or that self-destruct during the time period of the interaction have a large X drawn at the bottom of their lifeline. In our example, the object Application is created when a new application is received and is deleted after the application has been processed.

Add Object Deletion:	1	Click the third symbol button, the X, on the control bar. This is the object deletion symbol.
	2	Place the cursor under the —Application‖ object lifeline and click the left mouse button. An object deletion symbol is drawn. Add the other object deletion symbols as shown in Figure 16-4.
Save the Diagram:	3	Save the sequence diagram.

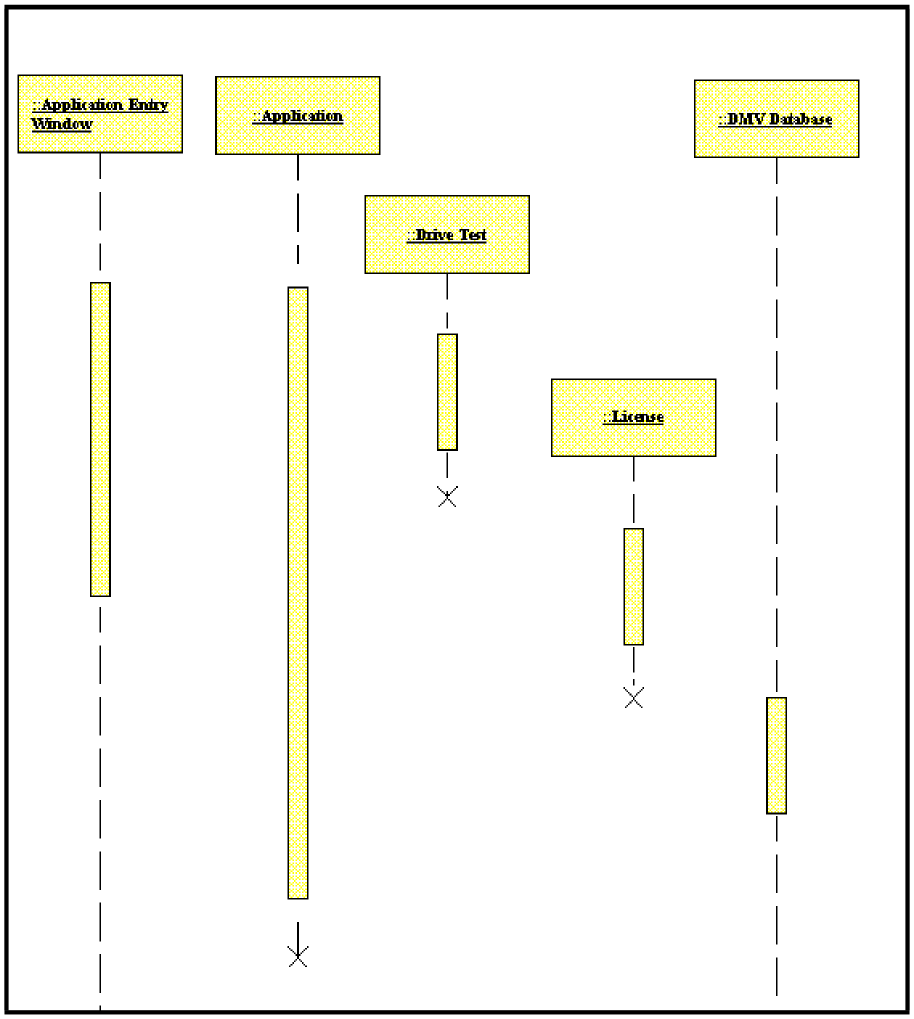


Figure 14-4 Sequence Diagram with Object Deletion Symbols Added

Adding Procedure Calls to the Diagram

Messages that are procedure calls can be passed from one object to another. It is basically a command for the receiving object to perform a certain action. When adding any message, there are certain pieces of information you are required to enter:

- **Name.** Name of the message.

- **Type.** A message can be of three types: procedure call, flat flow of control, or asynchronous stimulus.
- **Occurs Many Times.** This is a flag that indicates whether a message is sent many times to multiple receiver objects.
- **Guard Condition**. If the message is to be sent only if a condition is met, this field will contain that condition.

In our example, the object Application Entry Window submits an application to the object DMV Validation when a new application is accepted. This is an example of an unconditional procedure call. The Application object sends a message to the object License to create a new license. However, this message is sent only if the applicant passes the driving test. This is an example of a conditional procedure call.

Add Procedure Call:	1	Click the first arrow button, the bold full arrow on the control bar. This is the procedure call arrow.
	2	Place the cursor on the activation bar under the —Application Entry Window‖ object and click the left mouse button. Holding the left mouse button down, drag the cursor to the activation bar under the —‖Application‖ object and release the left mouse button. The Label Message dialog box appears.
	3	The Label Message dialog box appears. Click New Method and type —New‖ for the name of the method. Click OK to return to the Label Message window. Click OK again to exit this window.
Add Procedure Call with Condition:	4	Place the cursor on the activation bar under the —Application‖ object and click the left mouse button. Holding the left mouse button down, drag the cursor to the activation bar under the —DMV Database‖ object and release the left mouse button.
	5	The Label Message dialog box appears. Click New Method and type —Update Success‖ as the method name. Click OK to return to the Label Message dialog box. Type —passed‖ in the guard condition field and click OK.

6 Add the rest of the procedure calls as shown in Figure 14-5.

Save the Diagram: 7 Save the sequence diagram.

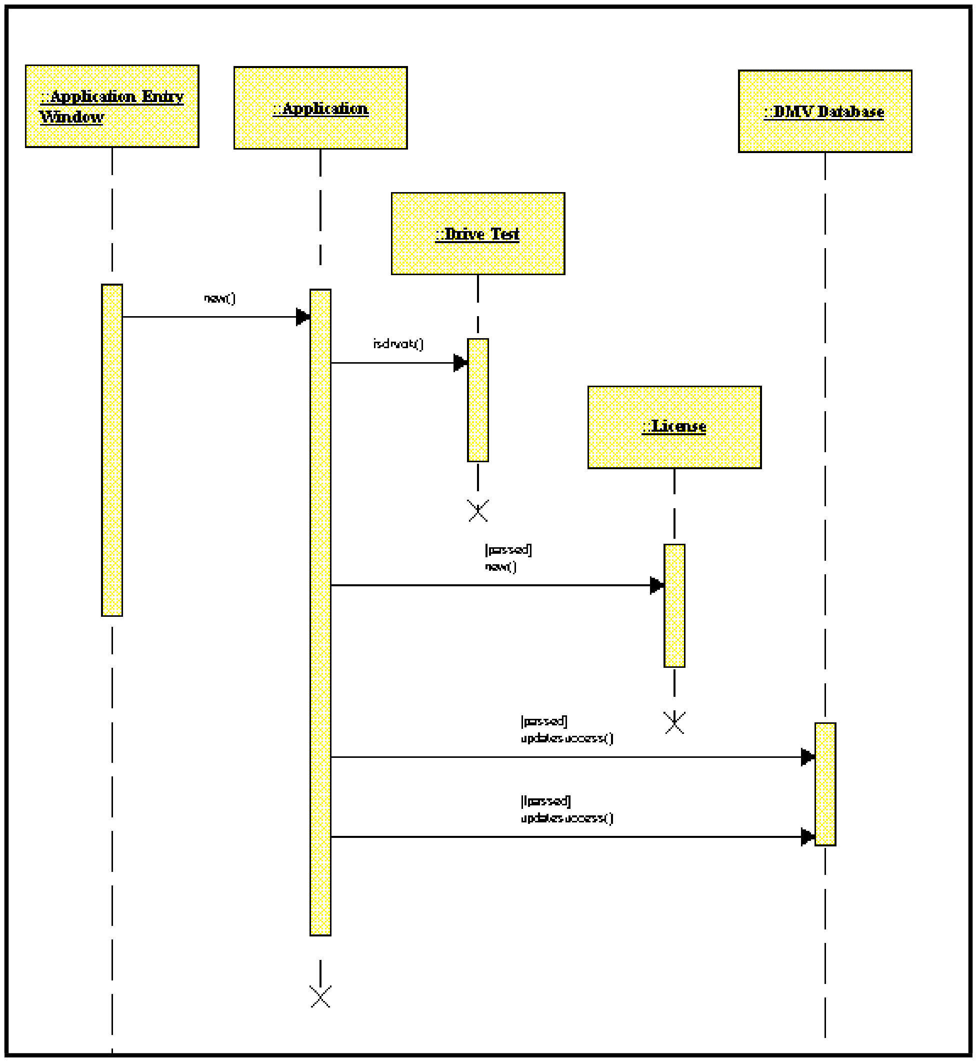


Figure 14-5 Sequence Diagram with Procedure Calls Added

Adding Return to the Diagram

An object can send a message in response to a message sent to it earlier. In our example, the application object returns the message sent to it by the application entry window object.

Add Return:	1	Click the fourth arrow button, the dashed arrow, on the control bar. This is the return symbol.
	2	Place the cursor on the activation bar under the —Drive Test‖ object and click the left mouse button. Drag the cursor to the activation bar under the —Application‖ and release the mouse button. A return arrow is drawn.
Save the Diagram:	3	Save the sequence diagram.

Adding Text Notes to the Diagram

Add Note:	1	Click the rightmost button on the toolbar, an uppercase T. This is the Add Text button.
	2	Place the cursor under the procedure call isdrvok() and click the left mouse button. The Add Text window appears. Type —passed = isdrvok()‖ and click OK. The note is added to the diagram.
Save the Diagram:	3	Save the sequence diagram.

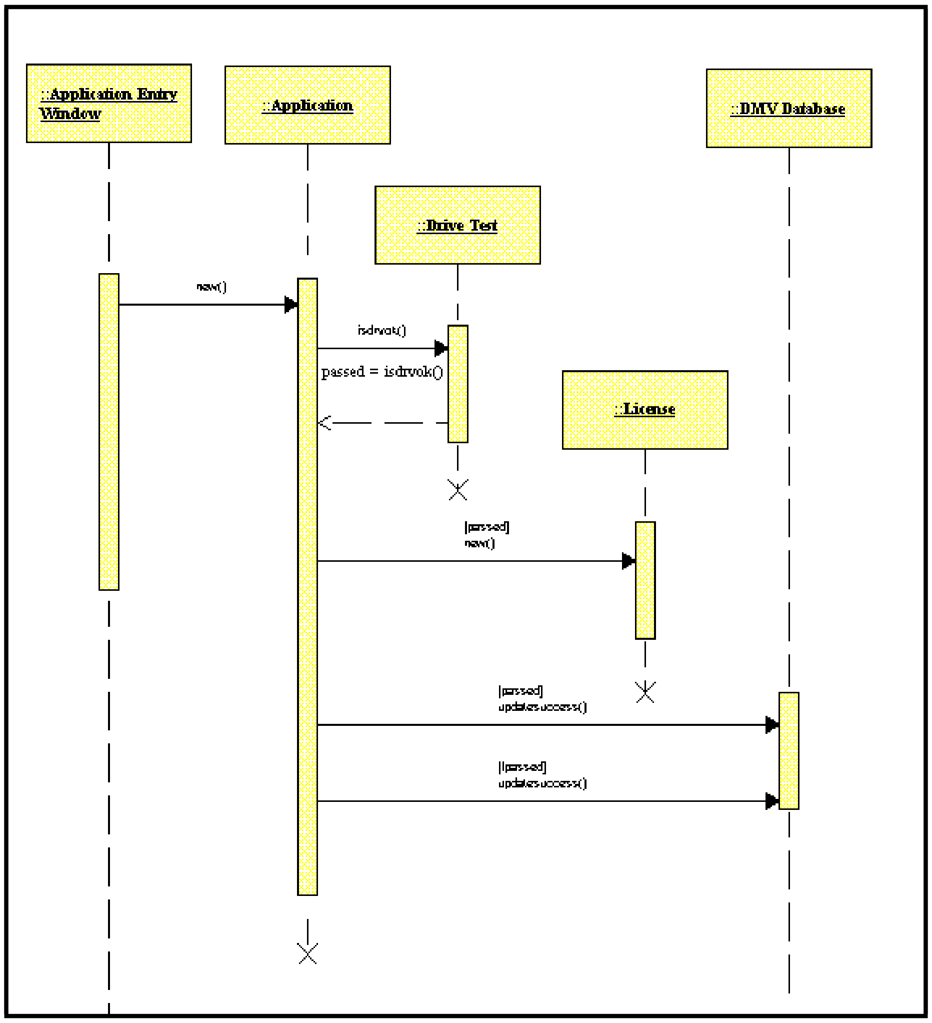


Figure 14-6 Completed Sequence Diagram

Purchase your UML Diagraming and Modeling software today for only $29 per month. It includes all UML notations.

Go to https://www.visiblesystemscorp.com/Subscriptions.htm