

# Using COM Objects For Data Access And Visualization

Ganesh Gopalan

COAS, Oregon State University

104 Ocean Administration Building

Corvallis OR-97333

**Abstract:** An Earth Science product could be the end result of several software tasks that process raw data. In this paper we describe how the tasks maybe mapped to different technologies to achieve the end result. The focus has been on building and using COM objects to accomplish specific tasks and in leveraging the functionality of existing applications such as MATLAB. Three applications are discussed; first the AVHRR Temperature Extractor that extracts sea surface temperatures from AVHRR images for a given drifter track and compares them with the temperatures collected by the drifter. User parameters are first submitted to the web-server. COM objects extract temperature values from the pixels in the image that correspond to the drifter's location (track). Drifter and image temperatures are returned to the web-server. The output can be viewed as XML, which is rendered using an XSL style-sheet. If the user now chooses to see a plot of the data, temperature values are extracted from the XML tree and passed to MATLAB using OLE automation. MATLAB can be launched both on the web-server and on the client and its methods invoked to generate a JPEG file as the final result. The technologies used in this application include DCOM, OLE automation and XML. The next application is the generation of volumetric slice plots from temperature and other data available at different depths. 3D slices can be applied along any of the latitude, longitude or depth axes. MATLAB is again accessed as an automation server to generate the plot. Finally we discuss the development of smart sensors. This project involves adding logic to an instrument such as a drifter to enable it to be controlled remotely or to provide near real-time access to data. A Windows CE device was selected to host the instrument. A DCOM client and server are used for accessing data on the device. Serial I/O is used to read parameters from the measuring instrument and supply it to the device.

## 1. INTRODUCTION

In this paper we describe applications where several software technologies have been used effectively to accomplish given Earth Science tasks.

The AVHRR Temperature Extractor retrieves temperatures from AVHRR images that match a given drifter's path and time frame.

“Volumetric Slice Plots” demonstrates 3D visualizations of ocean data at different depths, using MATLAB.

The “Smart Sensor” project involves connecting an oceanographic instrument, such as a drifter, to a Windows CE device that can host some programmable logic. The smart sensor is designed to provide near real-time updates of the data to clients.

In all these applications, Microsoft's Component Object Model (COM) is one of the key technologies used and is highly useful in inter - object communications.

## 2. AVHRR TEMPERATURE EXTRACTOR

The Temperature Extractor is a tool that allows the user to compare drifter temperatures with temperatures from AVHRR images that match the drifter's timeframe. Images with pixels that match the drifter's path are decompressed and the temperatures extracted from the pixel values.

### 2.1. USER INTERFACE

A list of drifter Ids is displayed from which the user can make a selection. The start and end dates are retrieved dynamically based on the drifter selected. These can be modified to select a shorter timeframe. The output format can be XML, HTML (Table) or text. To view a plot of the data, the user must select XML, as this allows the appropriate temperatures to be extracted from the XML tree and passed to MATLAB.

[Plot temperatures](#)

Using MATLAB on my machine.  
 Using MATLAB on COAS's remote server.

Number	DrifterCount	ImageCount	Instrument_ID	Std_File	Longitude	Latitude	Drifter_Temperature	Image_Temperature	Date_Local_Time	Date_Sampled
1	1	1	20137	<a href="#">sat9322923m.ebc</a>	-124.228	38.875	13.72	13.8	1993-08-17 17:13:55	1993-08-17 23:00:00
2	1	2	20137	<a href="#">sat9323003m.ebc</a>	-124.228	38.875	13.72	13.7	1993-08-17 17:13:55	1993-08-18 03:00:00
3	2	3	20137	<a href="#">sat9323123mn.ebc</a>	-124.267	38.746	13.56	13.1	1993-08-18 07:10:33	1993-08-19 23:00:00
4	3	3	20137	<a href="#">sat9323123mn.ebc</a>	-124.287	38.74	13.72	13.2	1993-08-18 08:48:28	1993-08-19 23:00:00
5	4	3	20137	<a href="#">sat9323123mn.ebc</a>	-124.286	38.74	13.88	13.2	1993-08-18 13:20:38	1993-08-19 23:00:00
6	5	3	20137	<a href="#">sat9323123mn.ebc</a>	-124.269	38.711	13.88	12.7	1993-08-18 16:52:19	1993-08-19 23:00:00
7	6	3	20137	<a href="#">sat9323123mn.ebc</a>	-124.275	38.648	13.56	13	1993-08-19 06:47:31	1993-08-19 23:00:00
8	7	3	20137	<a href="#">sat9323123mn.ebc</a>	-124.267	38.663	13.72	13	1993-08-19 08:21:07	1993-08-19 23:00:00
9	8	3	20137	<a href="#">sat9323123mn.ebc</a>	-124.219	38.655	14.04	12.8	1993-08-19 13:07:40	1993-08-19 23:00:00
10	9	3	20137	<a href="#">sat9323123mn.ebc</a>	-124.218	38.65	14.04	12.8	1993-08-19 14:48:28	1993-08-19 23:00:00

Fig. 1. Output for drifter 20137.

## 2.2. OUTPUT

Figure 1 shows output for a selected drifter (20137). The images whose dates are closest to that of the drifter for a

given date and time can be viewed by clicking on the links. The drifter position will be shown superimposed on the image. Also shown are the latitude, longitude, temperature and date and time when the drifter values were recorded, along with the image temperature and the image dates.

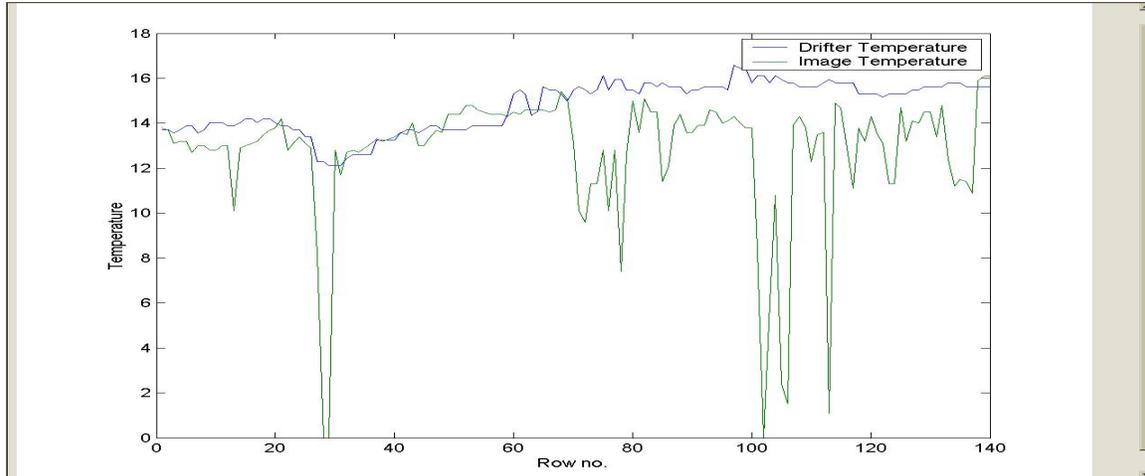


Fig. 3. A plot of drifter temperature and image temperature for comparison and analysis

Figure 3 shows a plot of drifter temperature versus image temperature for drifter 20137. The image temperatures were extracted from the GIF image by running a decompression algorithm and reading the pixel values.

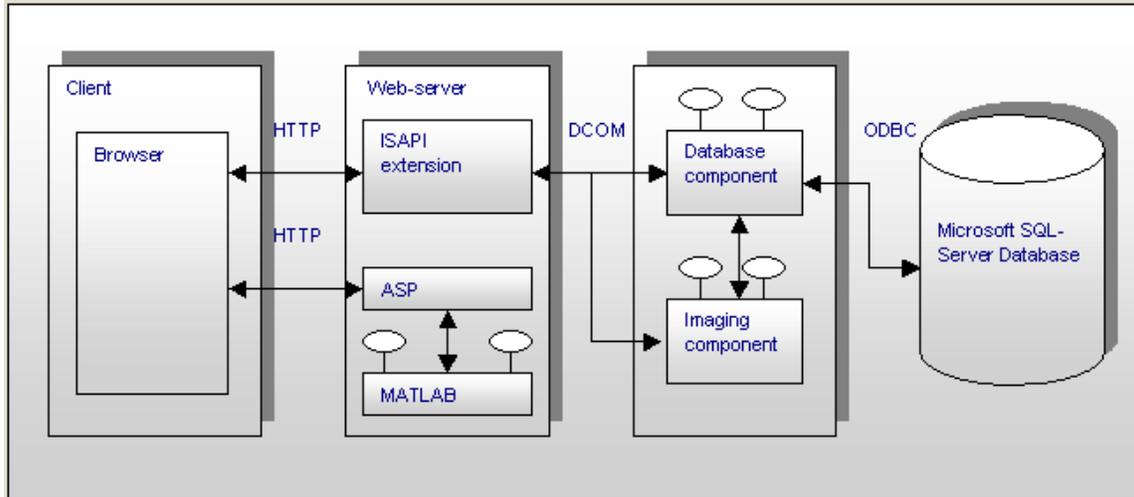


Fig. 4. The Implementation of the AVHRR Temperature Extractor

### 2.3. IMPLEMENTATION

The figure shows the application framework for the AVHRR Temperature Extractor. The backend architecture is built on a Windows NT/2000 intranet. The entry point into the system is through a Microsoft Windows 2000 workstation running Internet Information Server 5.0. Requests are handled by a piece of code called an ISAPI extension. ISAPI stands for Internet Server API. It is far more powerful than CGI in two key respects [CWT97]. Performance – well-designed ISAPI extensions can improve performance by an order of magnitude over a similar CGI application. Flexibility – ISAPI extensions enjoy a close integration with the server, which enables some actions to be executed more cheaply.

When a request is received by the web-server, appropriate validation is done before launching objects on remote machines to process the request. This is done using the Distributed Component Object Model (DCOM). DCOM supports distributed objects – that is, it allows application developers to split a single application into a number of different component objects, each of which can run on a different computer [EE98]. The ISAPI extension becomes a DCOM client and the remote objects are DCOM servers.

There are 2 DCOM components that run as remote servers in this application on a machine that serves as a “computational tier”. One is called the Database Component, which is responsible for making connections to the database and retrieving data. The other component, called the Imaging Component is responsible for extracting temperatures from the images. The Imaging Component contacts the Database Component rather than the database for data. The results are then sent back to the web-server (ISAPI extension), which then returns them to the client.

The objects expose COM interfaces (represented by the lines with circles at the end), so that their methods can be invoked locally or remotely. The ISAPI extension that is part of the multi-tiered system uses DCOM to invoke the database object and the Imaging component when processing a user request. Several such components can be built and integrated into the system.

Communication between the web-browser and web-server is limited to HTTP, which gives clients outside the local intranet access to the system without granting them the ability to run any code on the local system(s). This prevents any malicious use of system resources. The web-server communicates with remote objects on the computational tier using DCOM in a manner transparent to the clients. Also, non-DCOM clients can access the system in this manner.

The ISAPI extension uses static load balancing when processing requests. The DCOM objects are launched on 2 different machines for alternate requests. The framework allows more machines to be added to the system and the components can be installed and registered on these machines.

#### A. The Database Component

The Database Component has been implemented using MFC and ODBC for database functionality. MFC is Microsoft’s class libraries for software development while ODBC or Open Database Connectivity is a standard used in communicating with databases. However, the database functionality lies underneath a layer of COM that enables other database libraries to be used for data connectivity without affecting how the object is accessed by other COM clients. COM also supports versioning, so clients aware of

new interfaces (functionality) just use them while old clients continue to use old interfaces.

Methods supported include those that allow data to be retrieved from the Drifters and Images tables in the EOS database on the Microsoft SQL-Server. The client specifies a filter and provides pointers to memory locations that are to be filled with query results.

### B. The Imaging Component

The Imaging Component returns drifter and image temperatures.

The Imaging Component creates an instance of the Database Component that supplies it with GIF images for extracting the temperatures. The connection to the SQL Server is made by the Database Object, which then returns the results to the Imaging Component. The Imaging Component uses a GIF decompression algorithm to extract those pixel values that match the drifter's path.

Both the Database Object and the Imaging Component have been implemented as in-process, local and remote servers. In-process servers are those that can be run within the address space of the client. Their main benefit is that memory transfers between the server and client are extremely fast. Local servers are created in a process separate from the client but on the same machine. Remote servers are launched in a separate process and on a machine different from that of the client.

### C. Dynamic Invocation Of Remote Objects

Both components (DCOM servers) are capable of being launched dynamically as a user request comes in. In addition, the machine on which the objects are to be launched can be changed without having to recompile any of the back-end code. This is made possible by passing special parameters to the ISAPI extension in addition to the user's query.

Objects can be launched dynamically using COM only when required. This prevents CPU time from being wasted. The objects can also be shutdown when not in use, leading to better utilization of resources.

### D. Using MATLAB To Generate a Plot Of the Temperatures

MATLAB can be invoked from the web-server or browser using a technology called OLE automation. Automation is a concept through which an object exposes its methods to the outside and allows them to be invoked through scripting environments. Automation makes it easier for interpretive and macro languages to access COM components, and also makes it easier to write components in these languages

[Rog97a]. The MATLAB object is created and its methods invoked to load the temperature data into its environment and generate a JPEG file of the plot. Below is a code snippet in VBScript that illustrates this concept:

```
<%  
Sub Results()  
  
    data = Request.Form ( "data" )  
    filename = Request.Form ( "filename" )  
  
    set o = CreateObject  
        ( "Matlab.Application" )  
  
    o.Execute ("h = figure ('visible', 'off')")  
    o.Execute ("data=[" & data & "]" )  
    o.Execute ("x=data(:,1)")  
    o.Execute ("y=data(:,2)")  
    o.Execute ("z=data(:,3)")  
    o.Execute ("plot(x,y,x,z)")  
    o.Execute ("xlabel('Row no.')" )  
    o.Execute ("ylabel('Temperature')" )  
    o.Execute ("legend  
        ('Drifter Temperature',  
         'Image Temperature')" )  
    o.Execute ("saveas(h," &  
        "c:\InetPub\wwwroot\temp\  
        & filename & ".jpg")")  
  
    response.redirect  
    ("http://skor.oce.orst.edu/temp/"  
    & filename)  
  
End Sub  
%>
```

The code shows how the MATLAB object is created and initialized with data, and how the generated plot is saved to a JPEG file. The angular brackets <% and %> indicate that the code runs on the web-server.

## 3. VOLUMETRIC SLICE PLOTS

This application demonstrates the use of OLE automation to generate 3D slices of the ocean. Parameters available for plotting include temperature, oxygen, phosphate levels etc.

### 3.1. USER INTERFACE

The user can select a grid and a range of depths and any of the parameters such as temperature, phosphate etc. as shown in figure 5. When displayed as XML, the data can be plotted in the form of a 3D slice.

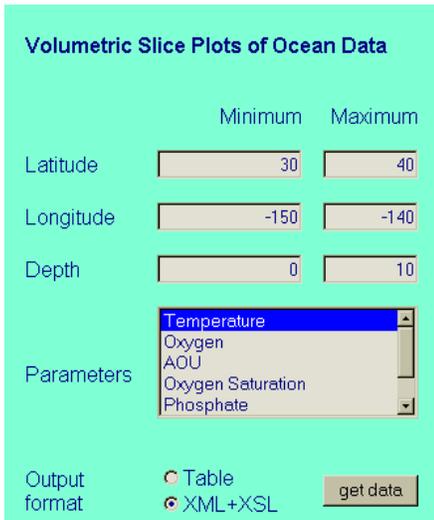


Fig. 5. User interface for generating volumetric slice plots.

### 3.2. OUTPUT

Figure 6 shows the output for a user query. For the returned data, the user can select points along the 3 axes (latitude, longitude and depth) to generate a 3D slice as shown in figure 7.

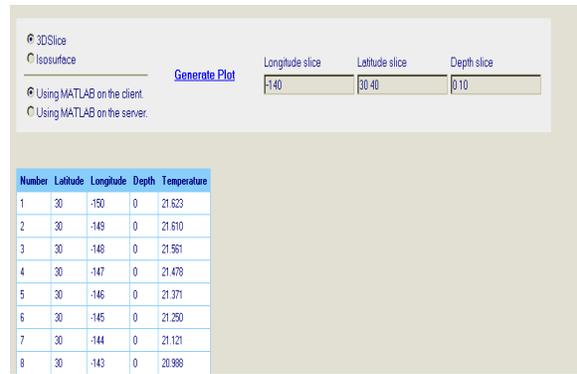


Fig. 6. The output for a user query of latitude, longitude, depth and temperature

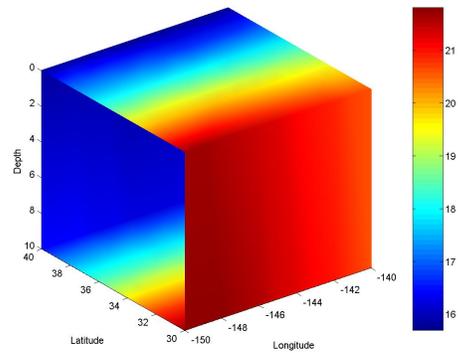


Fig. 7. A 3D slice of the data in figure 6

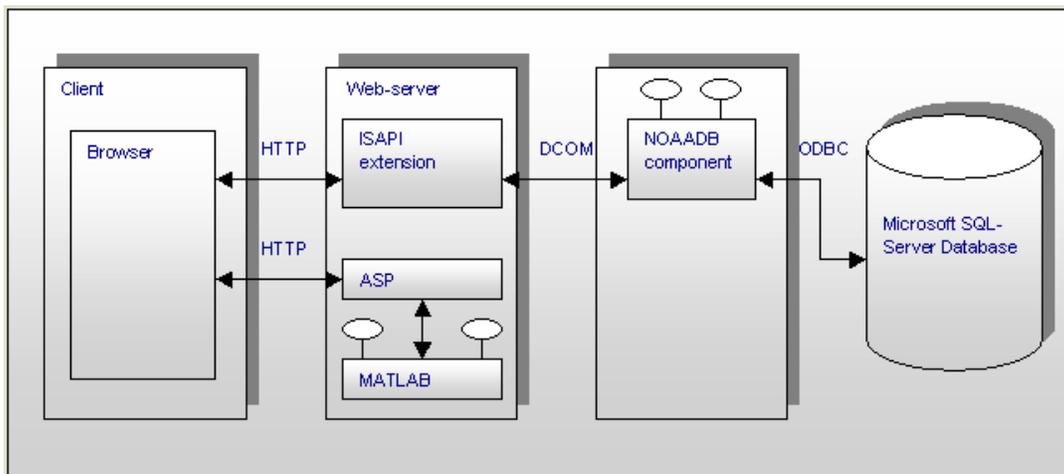


Fig. 8. The implementation of volumetric slice plots

### 3.3. IMPLEMENTATION

The implementation is identical to that of the Temperature Extractor except that a different COM object called the NOAADB component is used to retrieve 3D data. The Imaging Component is not used in this application.

The following code snippet illustrates how the 3D slices are generated from the volumetric data.

```
<%Results%>
<%
Sub Results()
    ....

    set o = CreateObject ( "Matlab.Application" )

    o.Execute ( "[xi,yi,zi]=meshgrid(min(x):max(x),
        min(y):max(y),min(z):10:max(z))" )
    o.Execute ( "vi=griddata3(x,y,z,v,xi,yi,zi)" )

    o.Execute ("slice(xi,yi,zi,vi,[' &
        slice_long & '], [' &
        slice_lat & '], [' &
        slice_depth & ']")" )

    o.Execute ("saveas(h,'" &
        "c:\InetPub\wwwroot\temp\"
        & filename & ".mat")" )

    response.redirect ("http://skor.oce.orst.edu/temp/"
        & filename)

End Sub
%>
```

The code shows how an instance of the MATLAB object is created through scripting and its methods invoked to create a 3D slice.

## 4. SMART SENSORS

This project demonstrates the advances in software and hardware technologies with regards to intelligent instrumentation. The goal is to show how the instrument itself can serve as a data source and provide near real-time updates to clients. The project is in the preliminary stages and is still under development. Currently the hardware configuration is complete. Serial I/O with an instrument is not yet implemented.

### 4.1. IMPLEMENTATION

We chose a Windows CE device made by Arcom Control Systems called the SBC-MediaGX-233 with 16MB of flash memory. The small footprint (a little more than 8" x 5") makes it suitable for attaching to a drifter. The device comes with about 32MB of RAM, a 10/100 Base T Ethernet port, 4 serial ports and a flat panel display (which is optional). It comes pre-installed with the Windows CE (version 3.0) operating system. It also has a built-in web-server that enables it to be accessed directly via a URL. The system being implemented is for proof of concept only, as during deployment the device would have to be connected to the network via a wireless card.

The communication between the server and client is through DCOM, while the measurements from the drifter will be read using serial I/O. Currently, we read data from a flat file, but in the final implementation, the data will be logged to this file by the instrument. Eventually we hope to support features such as the ability to dynamically change the sampling frequency on the drifter using the COM objects.

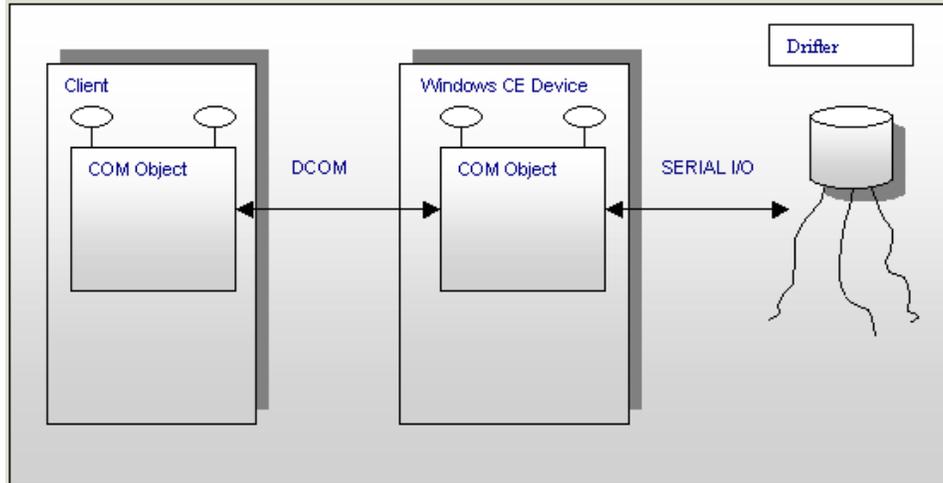


Fig. 9. The proposed Windows CE implementation of a smart sensor

The client was developed using Visual C++ 6.0 for Windows NT/2000 and the server was developed using Embedded Visual C++ 3.0 for Windows CE and the Active Template Library (ATL).

## 5. CONCLUSION

These projects demonstrate how COM and DCOM can be used in the development of software components for specific tasks. The components can be launched dynamically and the launch location (machine) can be changed at run-time. We have also been able to demonstrate that the COM components enable the development of thin clients since most of the computationally intensive work (such as GIF decompression) is done on the server by these components. The availability of DCOM on Windows CE facilitates data access through smart instruments. Finally, MATLAB itself

supports COM, which allows scripting environments such as VBScript or JScript to use its plotting capabilities.

## REFERENCES

- [CWT97] K. Clements, C. Wuestefeld, J. Trent and J. Clemens, *Inside ISAPI*. New Riders Publishing, 1997.
- [Rog97a] D. Rogerson , *Inside COM* . Microsoft Press, 1997.
- [EE98] G. Eddon G. and H. Eddon, *Inside DCOM*. Microsoft Press, 1998.