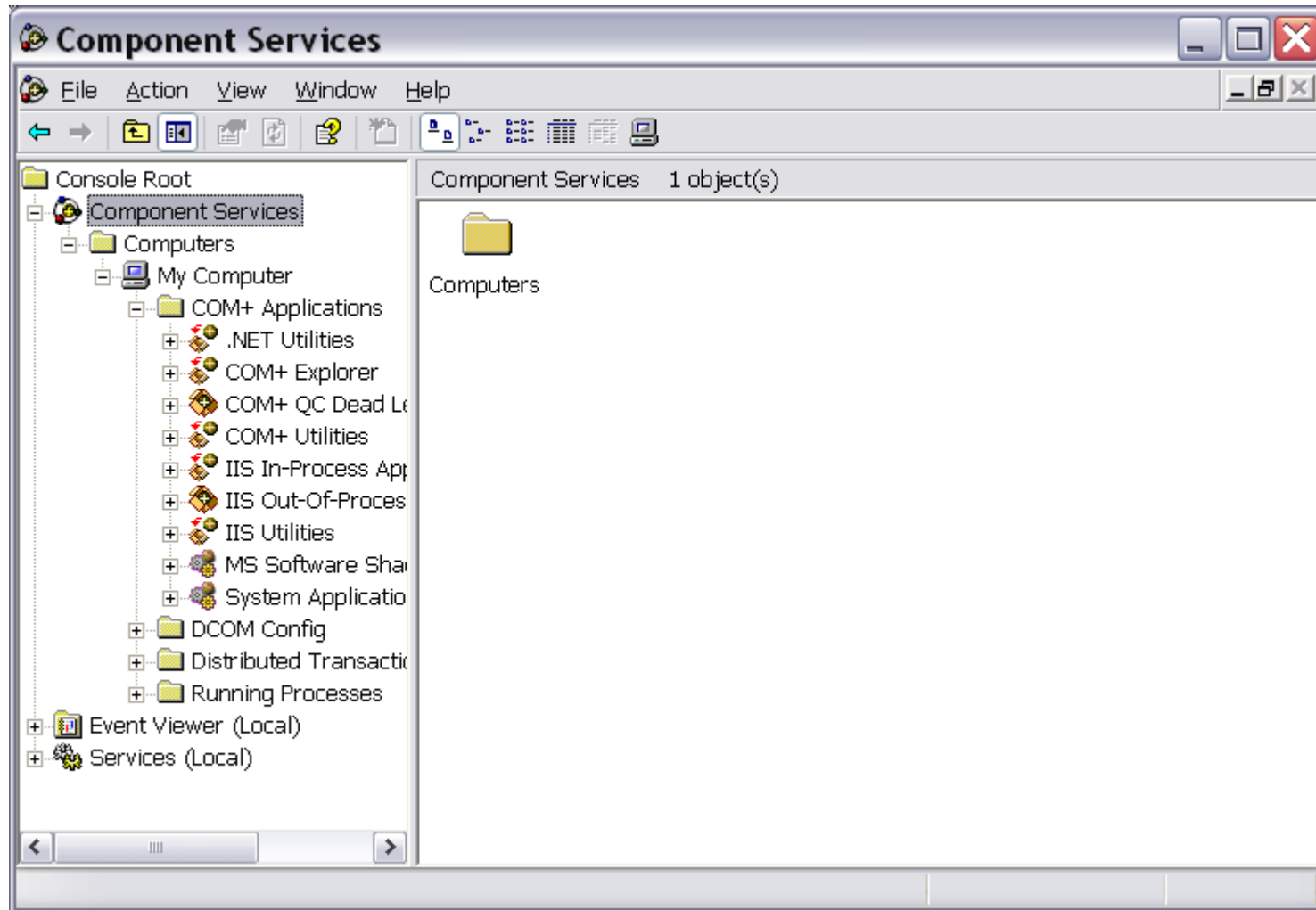
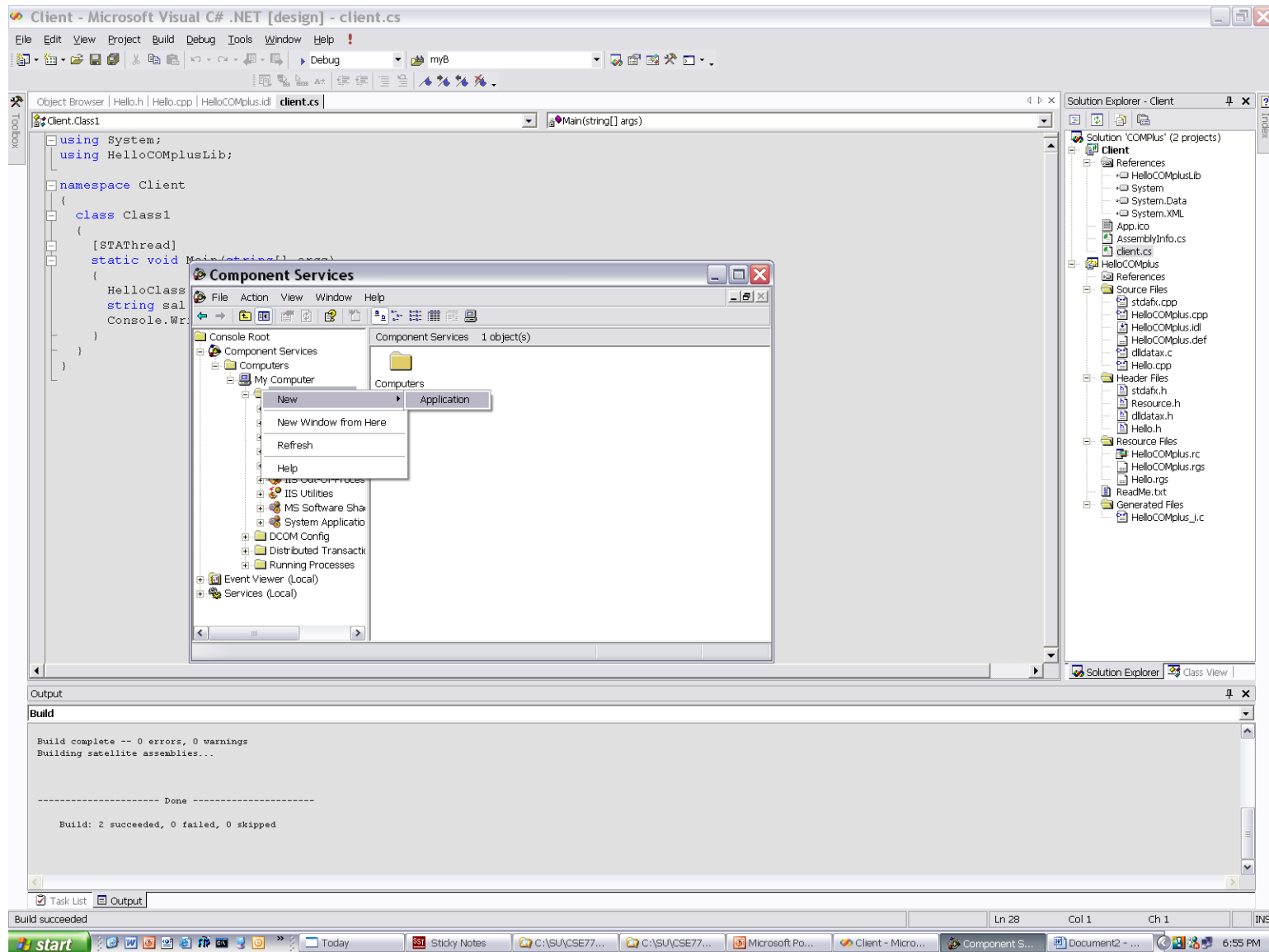


Creating a COM+ Application



Creating a COM+ Application



Creating a COM+ Application

The screenshot shows the Microsoft Visual Studio .NET environment. The main window displays the source code for a C# application named 'Client'. The code is as follows:

```
using System;
using HelloCOMplusLib;

namespace Client
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {
            HelloClass
            string sal
            Console.Wr
        }
    }
}
```

A 'Component Services' console window is open, showing the local computer's services. A 'Welcome to the COM+ Application Install Wizard' dialog box is in the foreground, offering to 'Install pre-built application(s)' or 'Create an empty application'. The 'Output' window at the bottom shows a successful build with 2 succeeded, 0 failed, and 0 skipped. The taskbar at the bottom shows the Start button and several open applications.

Creating a COM+ Application

Welcome to the COM+ Application Install Wizard 

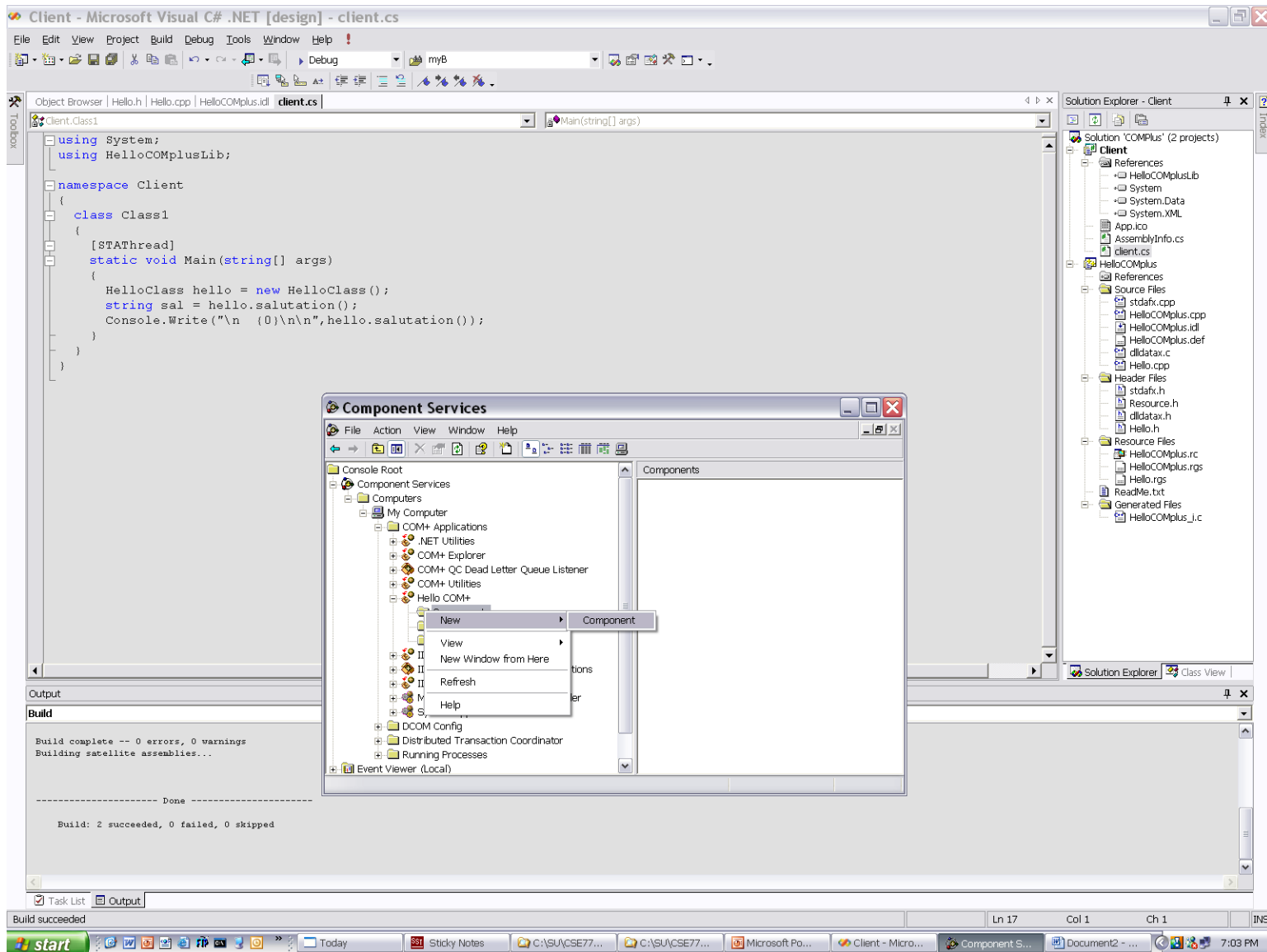
Create Empty Application 
Please specify the name of the new application.

Enter a name for the new application:

Activation Type

- Library application**
Components will be activated in the creator's process.
- Server application**
Components will be activated in a dedicated server process.

Creating a COM+ Application



Creating a COM+ Application

The screenshot displays the Microsoft Visual Studio IDE with a C# client application. The main window shows the code for `Client.cs` in the `Client` namespace, featuring a `Class1` with a `Main` method that interacts with `HelloCOMplusLib`. The `Output` window at the bottom indicates a successful build.

Overlaid on the IDE are two windows related to COM+ components:

- Component Services**: A tree view showing the hierarchy of components on the system, including `COM+ Applications`, `COM+ Utilities`, and `COM+ Roles`.
- Welcome to the COM+ Component Install Wizard**: A dialog box titled "Import or install a component" with the following options:
 - Install new component(s).
 - Import component(s) that are already registered. (WARNING: This will not register interface and method information.)
 - Install new event class(es).The wizard is configured for `Application: Hello COM+` and `Computer: My Computer`.

The `Solution Explorer` on the right shows the project structure for `Client`, including references to `HelloCOMplusLib`, `System`, `System.Data`, and `System.XML`, along with source files like `stdafx.cpp` and `HelloCOMplus.cpp`.

Creating a COM+ Application

The screenshot shows the Microsoft Visual Studio .NET IDE with the following components:

- Client - Microsoft Visual C# .NET [design] - client.cs**: The main code window showing the following C# code:

```
using System;
using HelloCOMplusLib;

namespace Client
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {
            HelloClass hello = new HelloClass();
            string sal = hello.salutation();
            Console.WriteLine("\n {0}\n\n", hello.salutation());
        }
    }
}
```
- Component Services**: A window showing the Component Services console tree. The 'Hello COM+' component is selected under 'My Computer' > 'COM+ Applications' > 'Hello COM+' > 'Components'.
- Welcome to the COM+ Component Install Wizard**: A dialog box with the 'Import or install a component' option selected. The application is identified as 'Hello COM+' on 'My Computer'.
- Select files to install**: A file selection dialog showing the 'Debug' folder containing 'HelloCOMplus.dll' and 'HelloCOMplus.tlb'. The 'File name' field is set to 'HelloCOMplus.dll' and the 'Files of type' is set to 'Component Files (*.dll;*.tlb)'.
- Output**: The bottom window showing the build output: 'Build complete -- 0 errors, 0 warnings. Building satellite assemblies... Done. Build: 2 succeeded, 0 failed, 0 skipped.'
- Task List**: Shows 'Build succeeded'.
- System Tray**: Shows the taskbar with the Start button, system clock (7:06 PM), and active windows including 'Sticky Notes', 'C:\SUA\CSE77...', 'Microsoft Po...', 'Client - Micro...', 'Component S...', and 'Document2 - ...'.

Creating a COM+ Application

Welcome to the COM+ Component Install Wizard 

Install new components 
Please specify the file(s) that contain the components you want to install.

Click Add to choose the file(s) that contain the components you want to install.

Files to install:

File	Contents
C:\SU\CSE775\CODE\COMPlus\HelloC...	components, typeLib

Components found:

Component	Properties	Interfaces	Installed
Hello	COM+	found	No

Details

Creating a COM+ Application

The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Client - Microsoft Visual C# .NET [design] - client.cs**: The main code window showing the following C# code:

```
using System;
using HelloCOMplusLib;

namespace Client
{
    class Class1
    {
        [STAThread]
        static void Main(string[] args)
        {
            HelloClass hello = new HelloClass();
            string sal = hello.salutation();
            Console.WriteLine("\n {0}\n\n", hello.salutation());
        }
    }
}
```
- Output Window**: Shows the build process and successful execution:

```
Build

The project is up-to-date.
Building satellite assemblies...

----- Done -----

Build: 2 succeeded, 0 failed, 0 skipped
```
- Console Window**: Shows the application's output:

```
C:\SU\CSE775\CODE\COMPlus\Client\bin\Debug\Client.exe
Hello from COM+
Press any key to continue
```
- Solution Explorer**: Shows the project structure for 'Client', including references to HelloCOMplusLib and various source files.
- Task List**: Shows 'Build succeeded'.
- Windows Taskbar**: Shows the system tray with the time 7:10 PM and the Start button.

Creating a COM+ Application

The screenshot displays the Microsoft Visual C++ IDE with the following components:

- Code Editor:** Shows the source code for `client.cpp`. The code includes headers for `atlbase.h` and `iostream`, and uses `CoInitialize`, `CoCreateInstance`, and `CComBSTR` to create and call a COM object.
- Console Window:** Displays the output of the application: "Hello from COM+" followed by "Press any key to continue...".
- Solution Explorer:** Shows the project structure for "HelloCOMplus", including source files, header files, and resource files.
- Output Window:** Shows the build process: "Build started: Project: HelloCOMplus, Configuration: Debug Win32", "HelloCOMplus - up-to-date.", and "Build: 3 succeeded, 0 failed, 0 skipped".

```
// client.cpp
#import "..\HelloCOMplus\debug\HelloCOMplus.dll" no_namespace named_guids
#include <atlbase.h>
#include <iostream>

void main()
{
    try
    {
        ::CoInitialize(NULL);

        HRESULT hr = S_OK;
        IHello* pSal = NULL;

        hr = ::CoCreateInstance(CLSID_Hello, NULL, CLSCTX_ALL, IID_IHello, (void**) &pSal);


        CComBSTR sal;
        std::wstring temp = pSal->salutation();
        std::wcout << L"\n " << temp.c_str() << L"\n\n";
        pSal->Release();


        ::CoUninitialize();
    }
    catch(exception& e)
    {
        std::cout << "\n " << e.what() << "\n\n";
    }
}
```

Output:

```
Build
----- Build started: Project: HelloCOMplus, Configuration: Debug Win32 -----
HelloCOMplus - up-to-date.
----- Done -----
Build: 3 succeeded, 0 failed, 0 skipped
```


Creating a COM+ Application

Welcome to the COM+ Application Export Wizard 

Application Export Information
Please enter information required to export this application. 

Enter the full path and filename for the application file to be created. Component files will be copied into the directory you specify for the application file.

Export as:

Server application - Install this application onto other machines

Export user identities with roles

Application proxy - Install on other machines to enable access to this machine

Save application in COM+ 1.0 format - some properties may be lost

Creating a COM+ Application

