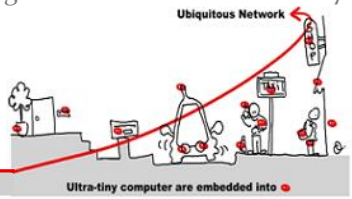
Web Service for Device (like UPnP) and Dynamic Composition (WComp)



1 Discovery of WComp

You can refer to the documentation available online as well as demonstration videos available for the installation and for taking the WComp middleware in hand:

https://www.wcomp.fr/download

You need to have a Windows® OS. Installing SharpDevelop and the WComp AddIn is mandatory and may be sufficient. At first run of SharpDevelop, you can choose your language in the menu "Outils / Options / Options de SharpDelevop/ Langue de l'utilisateur" in french, or "Tools / Options/ General / UI Language" in english.

Then, create a WComp Container:

- File / New → File…
- WComp.NET tab / C# Container item: creates a new file "Container1.cs" (tab at the top of the workspace)
- To manipulate the components, you must activate the graphical representation of the Container (WComp.NET tab at the bottom of the workspace).

Remember that you can only save your component assemblies using **Export…** in the WComp.NET menu.

2 Web Service for Device Composition

2.1 UPnP Tools Installation

Download the open source version of UPnP tools, formerly released as "Intel® Tools for UPnP Technologies":

http://opentools.homeip.net/dev-tools-for-upnp

Run the "*Device Spy*" tool, which is a Universal Control Point (UCP). This tool allows discovering UPnP devices, performing action invocations and event subscriptions/notifications on any of them. Then run the virtual UPnP device called "*Network Light*", verify it appeared in the UCP and test its functionalities. You can subscribe to events by right clicking on a *UPnP service* in the UCP.

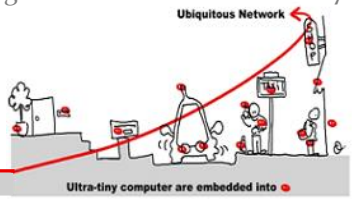
2.2 Integration of a UPnP Web Service for Device in WComp

We want to access and manage UPnP devices in WComp. To achieve this, we must generate proxy components for each discovered UPnP device. Let's generate it for the *Network Light*:

- File / New → File…
- WComp.NET / UPnP Device WebService Proxy
- Select the Light in the device list, all methods and state variables that you want to access via the proxy component (generally all of them). Click on Next and then on Finish. You've just generated a proxy component for this UPnP device.
- Reload the available components list to be able to access this newly generated component (using the menu entry WComp.NET / Reload Beans…)
- Find the component in the category "Beans: UPnP Device" (Tools tab) and instantiate it in the container.

We will now study how this component can be linked to others in order to interact with the UPnP device.

Web Service for Device (like UPnP) and Dynamic Composition (WComp)



3 LCA Model

3.1 Application Designing Using Components Assemblies

3.1.1 Simple Events

We will obtain the status of the virtual light in WComp by proceeding to the invocation of the "GetStatus" UPnP action. The proxy component was generated with an input port (a method) of the same name. We will use two components to proceed to this invocation and display its result: a button and a checkbox. They are available in the "Windows Forms" category. Connect the button "Click" event to the "GetStatus" input port of the *Network Light* proxy component, and the "GetStatus_Return" event of the proxy to the "set_Checked" input port of the checkbox.

3.1.2 Complex Events

We now want to control the status of the *Network Light* with a checkbox inside WComp.

Create a checkbox and connect its "CheckedChanged" event to the "SetTarget" **incompatible** method. Since the "SetTarget" method takes a boolean parameter and that the "CheckedChanged" has a different signature, they are not naturally compatible. The call has to be completed by providing a boolean information from the caller.

3.1.3 A more complete application

We will complete this application to become familiar with the components.
If you double click on your virtual light, it turns it on or off. However, your checkbox does not reflect these changes unless you click on the button to get its status.

- Find a way to reactively update the state of a new checkbox when the state of the UPnP device changes,
- We want to show the status of the *Network Light* in text in a label or textbox instead of in a checkbox (consider using the "ValueFormatter" component in the "Beans: Basic" category),
- Connect the components needed to vocalize the state of the light (using components "BoolFilter", "PrimitiveValueEmitter" and of course "TextToSpeech" in the "Beans: Services" category).

3.2 Creating a Component

You may need to create your own components if existing components do not meet your needs. The SharpDevelope environment offers the opportunity to create a component from a skeleton.

3.2.1 Create and use a simple component

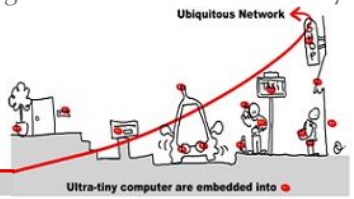
After quitting and restarting SharpWComp, you can create a new solution for creating the desired component:

- File / New → Solution…
- WComp.NET / Wcomp Bean Solution.

Choose a name and path for your solution, then add a new file to it (using the Projects tab):

- Right click on the project in the solution, and select Add → New Item…
- WComp.NET / C# Bean: creates a new file "Bean1.cs" or any name you chose,
- Update the "Beans" reference in the project, pointing it to the Beans.dll located in the AddIn installation directory, in SharpDevelop's files. You can now compile the bean template.

Web Service for Device (like UPnP) and Dynamic Composition (WComp)



We will make a component that adds two integers. This component will have two methods in entries: *intVal* and *intAdd* which allow you to specify an initial value and a second to add a new value. This component will emit an event which is the sum of both. We call this component *AddInt*.

Write the code, compile it and copy the created library to the component repository, located where you installed SharpDevelop, usually *C:\Program Files\SharpDevelop\2.2\Beans*. Reload beans repository. Your new component is now available in the specificed category, the default being "Beans: Basic".

Make an assembly to test your component with two textboxes, using components named "StringToInt" and "ValueFormatter". Display the result in a label.

3.3 The UPnP Wizard Designer

You will now see an example of designer that automatically instantiates UPnP proxy components when UPnP *devices* are discovered on the network: the UPnP Wizard Designer. It should already been installed and available with on shortcut on your desktop for more convenience (it is installed `in C:\Program Files (x86)\SharpDevelop\3.0\UPnPWizardDesigner`).

Launch the designer, and start the UPnP interfaces of a SharpWComp container. From this moment, if you launch a UPnP device like the "*Network Light"*, the proxy component will be automatically generated, compiled, loaded in the container, and instantiated with the correct URI property. The proxy component will also be removed when the device disappears.

You have to define with which container you want to connect this tool by selecting the right one in the Connect menu. You can also filter some device type to avoid to automatically create beans for such kind of devices.

Exercice 1 : Destroy all the beans in your container. Start the UPnPWizardDesigner, filter the unwanted type of UPnP device and start you UPnP Light. You should obtain a UPnP proxy bean corresponding to this UPnP device instanciated in your container.

3.4 Build your Own Application in 30 min.

UPnP Designer is only an example of the numerous protocols we can use in building an IAm application. Most of the time some Gateways allow to add dynamically proxy components to access to devices as soon as they are available (see demonstrations).

Moreover, because these technologies are over IP, software composition using them can use Device in a Wide area in a totally distributed approach.

Exercice 2 : After devices installation by the advisor on the local network, build your own IAm application using two, or more devices.