Software Methods and Tools

Fall 2015
Instructor: Yongjie Zheng

Lab #5: ArchStudio

Description:

In this lab, we will create a chatting application in ArchStudio. We will start from creating its architecture in Archipelago. Then we will use the mapping tool integrated in ArchStudio to automatically generate architecture-prescribed code. After that, we will modify user-defined code to complete application logics.

Please closely follow the instructions provided below.

Tasks:

1. Installing ArchStudio 4.

Open the Eclipse platform.

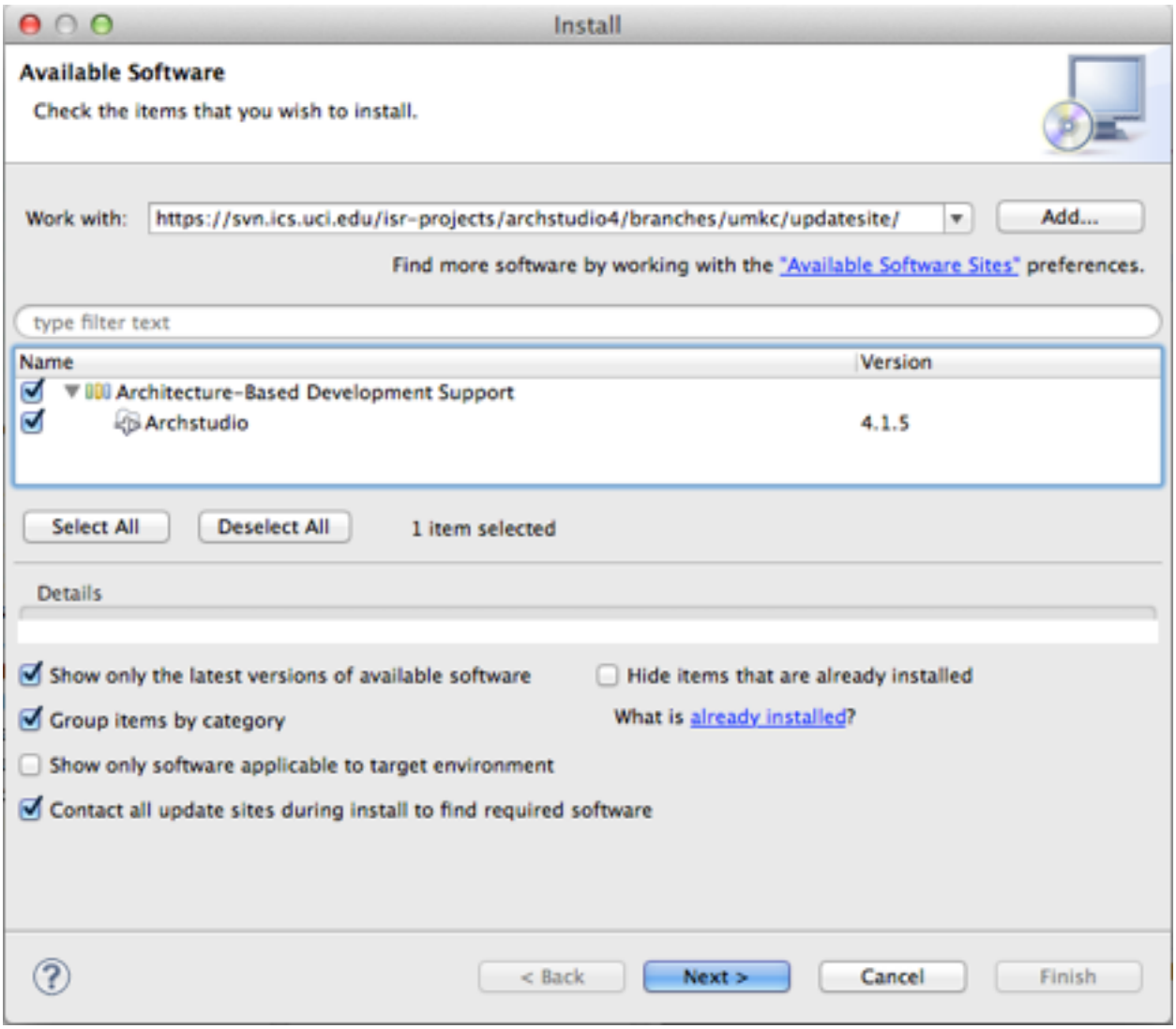
Go to Help > Install New Software, and then enter the following URL in the dialog panel that pops up.

https://svn.ics.uci.edu/isr-projects/archstudio4/branches/umkc/updatesite/

Click the Add button, and select all the components as illustrated in the following figure. Click Next.

Go through all the panels that follow. Once the installation is complete, you will be asked to restart Eclipse.

After Eclipse is restarted, you have successfully installed ArchStudio 4.



2. Creating the chatsys project and its architecture file.

Create an Eclipse plug-in project that does NOT contribute to UI. The name of the project is edu.umkc.chatsys

After the project is created, open its META-INF editor and add the "edu.uci.isr.myx.fw" project to its dependency list. Save it.

I am assuming you know how to do the above two steps.

Right click your project and select New > Other > ArchStudio 4 > ArchStudio Architecture Description. Click Next.

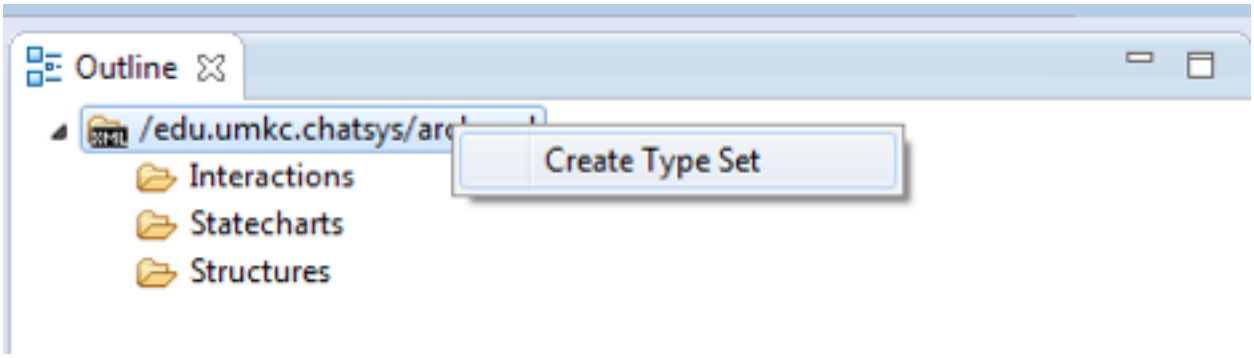
Give your architecture file a name (e.g. arch.xml) in the next panel. Click Finish.

Switch to the perspective of “ArchStudio 4”. If everything is fine, you should be able see the layout of ArchStudio as I showed you in class.

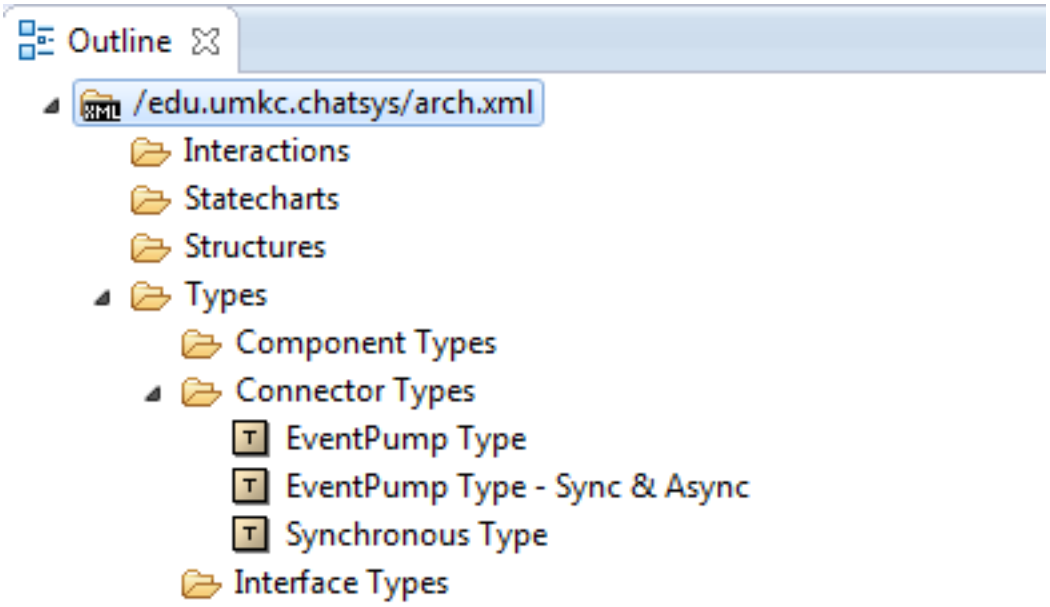
3. Creating the interface types and Java interface files.

Find the architecture file you just created in the Navigator view, and right click it. Select Open With > Archipelago.

Right click the architecture file in the Outline view and select “Create Type Set”, as shown in the figure blow.



Expand the Types entry, and it should look like the following figure.



Right click “Interface Types”, and select “New Interface Type”. Change the name of the new interface type to “chat type”.

Similarly, create another interface type named “chatevents type”.

Next, we are going to create a Java interface file for these two interfaces respectively.

Right click the project, and select New > Interface. Click Next.

Enter “edu.umkc.chatsys” as the package name, and “IChat” as the class name. Click Finish.

Repeat the above two steps, and create the second interface file with the same package name and “IChatListener” as the class name.

Open the two Java interface files you created, and modify them as follows.

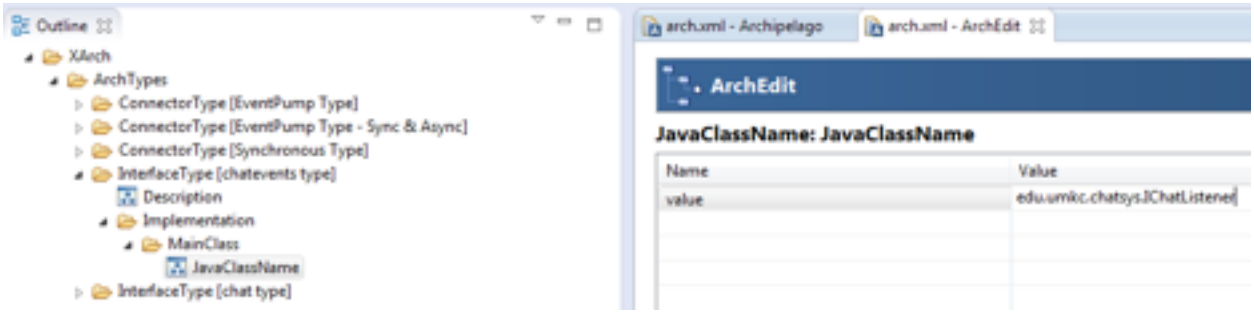
```
public interface IChat {
	public void sendMessage(String sender, String message);
}

public interface IChatListener {
	public void messageSent(String sender, String message);
}
```

Next, we will link the Java interface file to the corresponding interface type we created.

Right click your architecture file, and select Open > ArchEdit.

Go to the element XArch > ArchTypes > Interface Type [chatevents type] > Implementation > MainClass > JavaClassName. Enter “edu.umkc.chatsys.IChatListener” in the right panel, as shown below.



Repeat the above step and enter “edu.umkc.chatsys.IChat” for the “chat type” interface type.

Save all your changes.

4. Creating the structural architecture of the chatsys application.

Open the modified architecture file with Archipelago again.

Right click the element of “Structures”, and select “”New Structure”. Change the name of the created structure to “Chatsys”.

Double click the “Chatsys” element to open it in Archipelago editor.

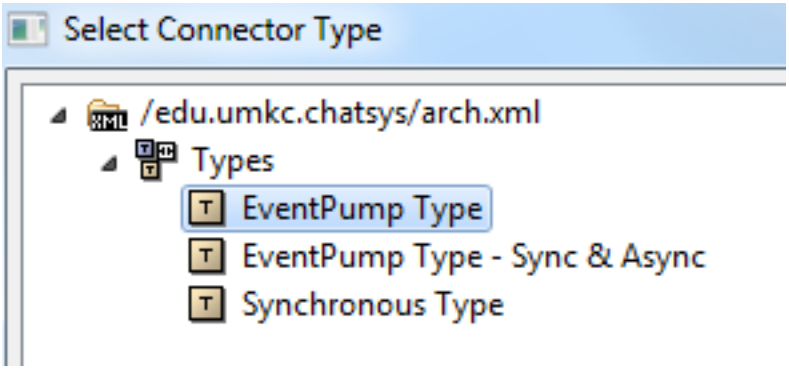
Create three components (right click in the editor and select “New Component”): one named “ChatServer” and the other two both named “ChatClient”.

For each component, create two interfaces respectively (Right click the component and select “New Interface”). Make one interface pointing out and the other one pointing in (Right click the interface icon and select “Edit Direction”).

Your current architecture should look like the following.

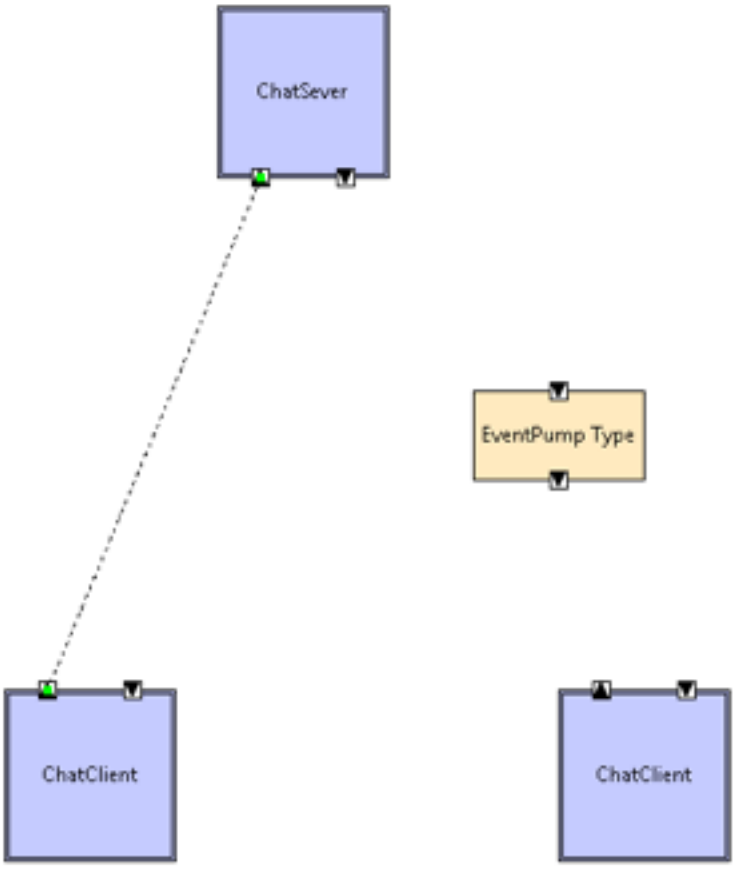
Next, we are going to create a connector and link all the components together.

Right click the editor and select “New Connector”. Select “EventPump Type” in the selection panel that pops up as shown below.

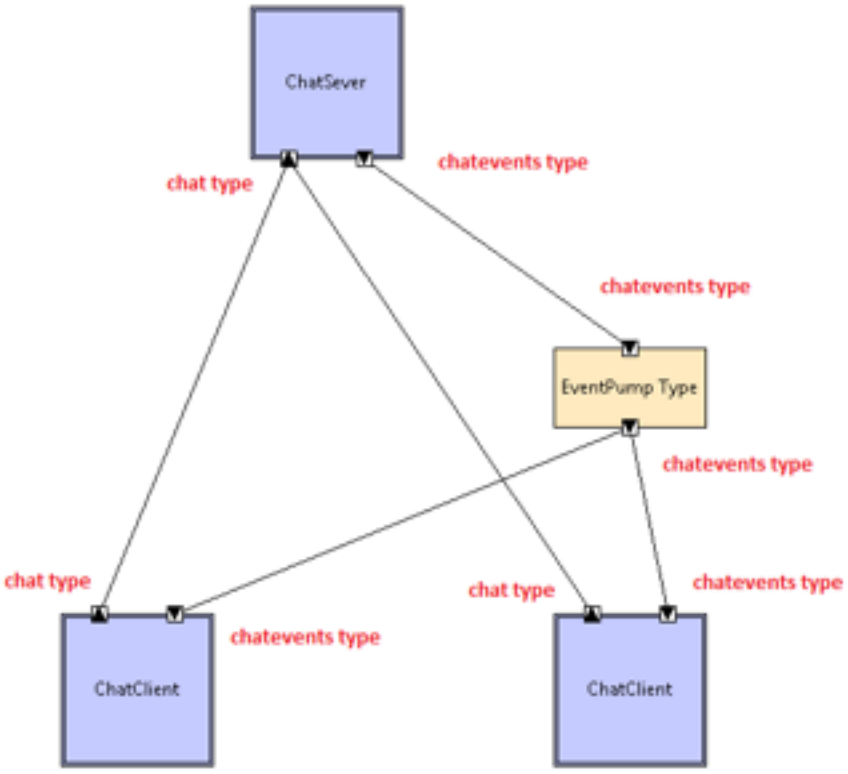


Notice that two interfaces are automatically created for the connector. You need to move one interface to see the other.

Next, create links (right click the editor and select New Link) to connect interfaces. Note that you should wait till the end becomes green before you release it, as shown in the following diagram.



Create all the connections and your final architecture should be as follows.



Remember the two interface types we created earlier? The last thing we need to do is to assign them to each interface. You can assign interface type by right clicking an interface and select “Assign Interface Type”. Refer to the above figure for the specific type of each interface.

Save all your changes in Archipelago.

5. Generating code and editing user-defined code.

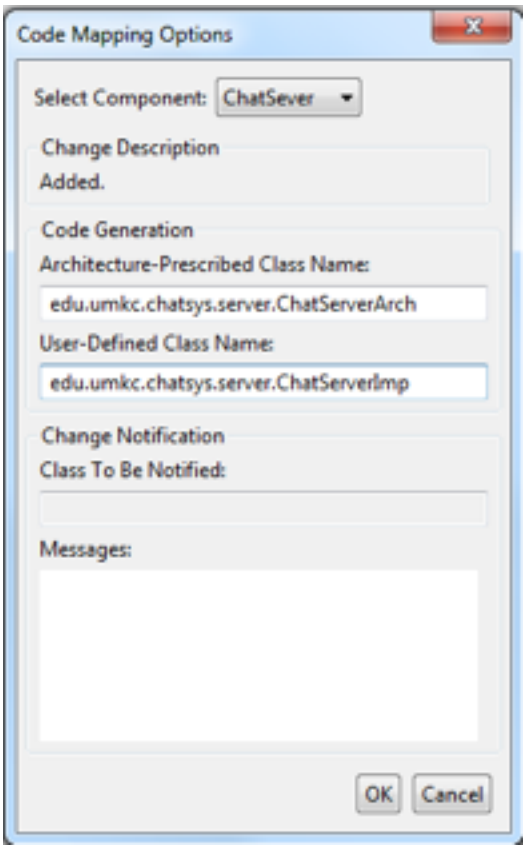
After you are done creating the architecture, right click in the Archipelago editor and select “Map Changes to Code with Dialog”.

In the dialog box that pops up (shown in the figure below), enter the class names of architecture-prescribed code and user-defined code for each component.

Fo the ChatServer component, enter “edu.umkc.chatsys.server.ChatServerArch” as its architecture-prescribed class name and “edu.umkc.chatsys.server.ChatServerImp” as its user-defined class name.

For the ChatClient component, its architecture-prescribed code should be “edu.umkc.chatsys.client.ChatClientArch” and user-defined code should be “edu.umkc.chatsys.client.ChatClientImp”.

Note that there are two components named ChatClient, and you must enter the same implementation information (i.e. class names) for both of them.



After you finish entering all the information, click OK and the code generation process will start.

Once code generation is done, you can right click a component and select “Open Architecture-prescribed Code” to view the generated code in Java Editor.

Next, we are going to edit user-defined code to complete application logics. we start with the ChatServer component. Right click it and select “Open User-defined Code”.

The only change you need to make here is in the sendMessage method. Replace the generated sendMessage method with the following code.

```
public void sendMessage (String sender,String message)   {
        //TODO Auto-generated method stub
        if (_arch.OUT_IChatListener != null){
              _arch.OUT_IChatListener.messageSent(sender, message);
        }
}
```

Similarly, open the user-defined code for the ChatClient component. Download the file (ChatClientImp.java) provided on the class website, and replace your existing code with the downloaded code.

Save both Java files.

6. Running the created Chatsys application in AIM Launcher.

Now you are ready to run the Chatsys application in AIM Launcher. Right click your architecture file, and select Open With > AIM Launcher.

Select “Chatsys” in the Outline view, and click the button labelled “Instantiate” in the AIM Launcher view.

END.