ArchStudio

Instructor: Yongjie Zheng
October 8, 2015

CS 490MT/5555
Software Methods and Tools
Outline

• What is ArchStudio
• Related Technologies
  • xADL
  • The Myx Architecture Style and Framework
  • 1.x-way mapping
• Toolset integrated in ArchStudio
  • AIM Launcher
  • Archipelago
  • ArchEdit
  • xMapper
  • Others: Archlight, Selector, Type Wrangler
What is ArchStudio

- Architecture-centric software development environment developed by Institute for Software Research (ISR) at University of California, Irvine (UCI).
- Open-source
- Eclipse plug-in
- Integrated tools for software architecture
  - Modeling
  - Visualizing
  - Analyzing
  - Implementing
More about ArchStudio

• Lead developer: Dr. Eric M. Dashofy (@Aerospace)
• Being used in a number of universities and several companies.
• The current (published) version is ArchStudio 5.
• In this class, however, we will be using an internal version of ArchStudio 4.
  • The code generator that we need in our lab/assignment is not included in ArchStudio 5.
Related Technologies

- xADL
- The Myx Architecture Style and Framework
- 1.x-way mapping.
xADL (pronounced as Z-A-DL)

- Architectural Description Language in XML developed by ISR at UCI
- This ADL is defined in a set of XML Schemas
- Modeling language behind ArchStudio
- Modular and highly extensible
- Core models:
  - Components (computation)
  - Connectors (communication)
  - Interfaces (the exposed entry and exit points for components and connectors)
  - Configurations (topology)
xADL Core Model

<archStructure id="archStructure90164" type="types:ArchStructure">
  <description>main</description>
  <component id="componentffa805157" type="types:Component">
    <description>Server</description>
    <interface id="interfaceffa80123" type="types:Interface">
      <description>getResource</description>
      <direction>in</direction>
    </interface>
  </component>
  <component id="componentffa12852" type="types:Component">
    <description>Client</description>
    <interface id="interfaceffa57518" type="types:Interface">
      <description>getResource</description>
      <direction>out</direction>
    </interface>
  </component>
  <connector id="connectorffa12435" type="types:Connector">
    <description>HTTP</description>
    <interface id="interfaceffa54685" type="types:Interface">
      <description>getResource</description>
      <direction>in</direction>
    </interface>
    <interface id="interfaceffa54686" type="types:Interface">
      <description>getResource</description>
      <direction>out</direction>
    </interface>
  </connector>
</archStructure>
<component id="componentffa805157" type="types:Component">
  <description>Server</description>
  <interface id="interfaceffa80123" type="types:Interface">
    <description>getResource</description>
    <direction>in</direction>
  </interface>
</component>

<component id="componentffa12852" type="types:Component">
  <description>Client</description>
  <interface id="interfaceffa57518" type="types:Interface">
    <description>getResource</description>
    <direction>out</direction>
  </interface>
</component>
xADL Core Model

<Client>

<Server>

<HTTP>

<interface id="interfaceffa54685" type="types:Interface">
<description>getResource</description>
<direction>in</direction>
</interface>

<interface id="interfaceffa54686" type="types:Interface">
<description>getResource</description>
<direction>out</direction>
</interface>

<connector id="connectorffa12435" type="types:Connector">
<description>HTTP</description>
</connector>

</Client>
xADL Core Model

```xml
<component id="componentffa805157" type="types:Component">
  <description>Server</description>
  <interface id="interfaceffa80123" type="types:Interface">
    <description>getResource</description>
    <direction>in</direction>
  </interface>
</component>

<component id="componentffa12852" type="types:Component">
  <description>Client</description>
  <interface id="interfaceffa57518" type="types:Interface">
    <description>getResource</description>
    <direction>out</direction>
  </interface>
</component>

<connector id="connectorffa12435" type="types:Connector">
  <description>HTTP</description>
  <interface id="interfaceffa54685" type="types:Interface">
    <description>getResource</description>
    <direction>in</direction>
  </interface>
  <interface id="interfaceffa54686" type="types:Interface">
    <description>getResource</description>
    <direction>out</direction>
  </interface>
</connector>
```
Some highlights of the xADL version that we will be using

- Component
  - Implementation - a fully qualified Java class name.
- Interface
  - Type: InterfaceType
- InterfaceType
  - Implementation - a fully qualified Java interface name.
- Connector
  - Interface
    - Type: InterfaceType
  - Type: ConnectorType
- ConnectorType
  - Implementation - a fully qualified Java class name.
Some highlights, cont.

• The implementation information is usually specified in the *Type* elements (e.g. `InterfaceType`, `ConnectorType`). The only exception is `Component`.

• A number of pre-defined or built-in connector types are available (e.g. `EventPump`), and can be reused in different applications.

• In other words, you can simply select a specific connector type when you create a new connector.
Myx.fw Framework

• Myx Architecture Style: support building flexible, high performance tool-integrating environments such as ArchStudio.

• Myx.fw is the supporting framework of the Myx style.

• Currently distributed as an integrated part of ArchStudio, but is also available as a separate package.

Implementing a myx.fw component

- Components have main classes that implement IMyxBrick.
- They may have as many auxiliary classes as you want.
- The main class may just be a wrapper for services provided internally.
Implementing a myx-fw component

• Components have three main jobs
  • Store data from the framework (IMyxBricklItems).
  • Implement lifecycle methods (init, begin, end, destroy).
  • Provide true objects for all provided interfaces.
Component Jobs

- Store IMyxBrickItems from the framework
  - The framework needs to store some data about the component along with the component.
  - Abstract base classes take care of this for you.
- Implement lifecycle methods
  - Called by the framework when the architecture is in particular states
    - init(): Brick is created
    - begin(): Brick is wired into the architecture and ready to start
    - end(): Brick is about to be unwired and shut down
    - destroy(): Brick is about to be dismissed
Component Jobs

• Provide true objects for each provided interface

• Each provided/required interface has a name.

• The framework will occasionally ask a component “give me the object that corresponds to this provided interface”.

• Likewise, a component may request, from the framework, the true object corresponding to one of its required interfaces.
Code Separation in ArchStudio

• The implementation of each component in ArchStudio is separated in two independent classes: *architecture-prescribed code* and *user-defined code*.

• Architecture-prescribed code is automatically generated, and cannot be manually edited.

• User-defined code is manually developed.

• A Java interface is also automatically generated. It contains the list of operations that architecture-prescribed code expects user-defined code to implement.
Architecture-prescribed code

- Architecture-prescribed code codifies externally visible characteristics of a component (the information prescribed in the architecture about the component).
- Can only be updated via code regeneration if architecture is changed.
User-defined code

- User-defined code contains implementation details of the operations and attributes generated in the corresponding architecture-prescribed code.

- User-defined code represents the internal implementation of a component.
Tools Integrated in ArchStudio

- Eclipse plug-in
- ArchStudio Perspective
  - AIM Launcher – launch the system in development within ArchStudio
  - ArchEdit
    - Visualization
    - Tree-view – schema driven
    - Indicate implementation mappings
- Archipelago
  - Visualization and modeling
  - Boxes and arrows
- xMapper - an architecture-implementation mapping tool, including a code generator
- ArchLight - analysis
- Selector – product Families
- Type Wrangler – consistency checks between types and instances.
Reference

• Some of today’s lecture materials are from the slides that Dr. Hazel Asuncion (@University of Washington, Bothell) made for her ArchStudio guest lecture.
Reminder

- Please bring and use your own laptop (with Eclipse and JDK installed) for Lab 5 next Tuesday.