Architecture Description Languages

Instructor: Yongjie Zheng

CS 5553: Software Architecture and Design

Modeling Notations

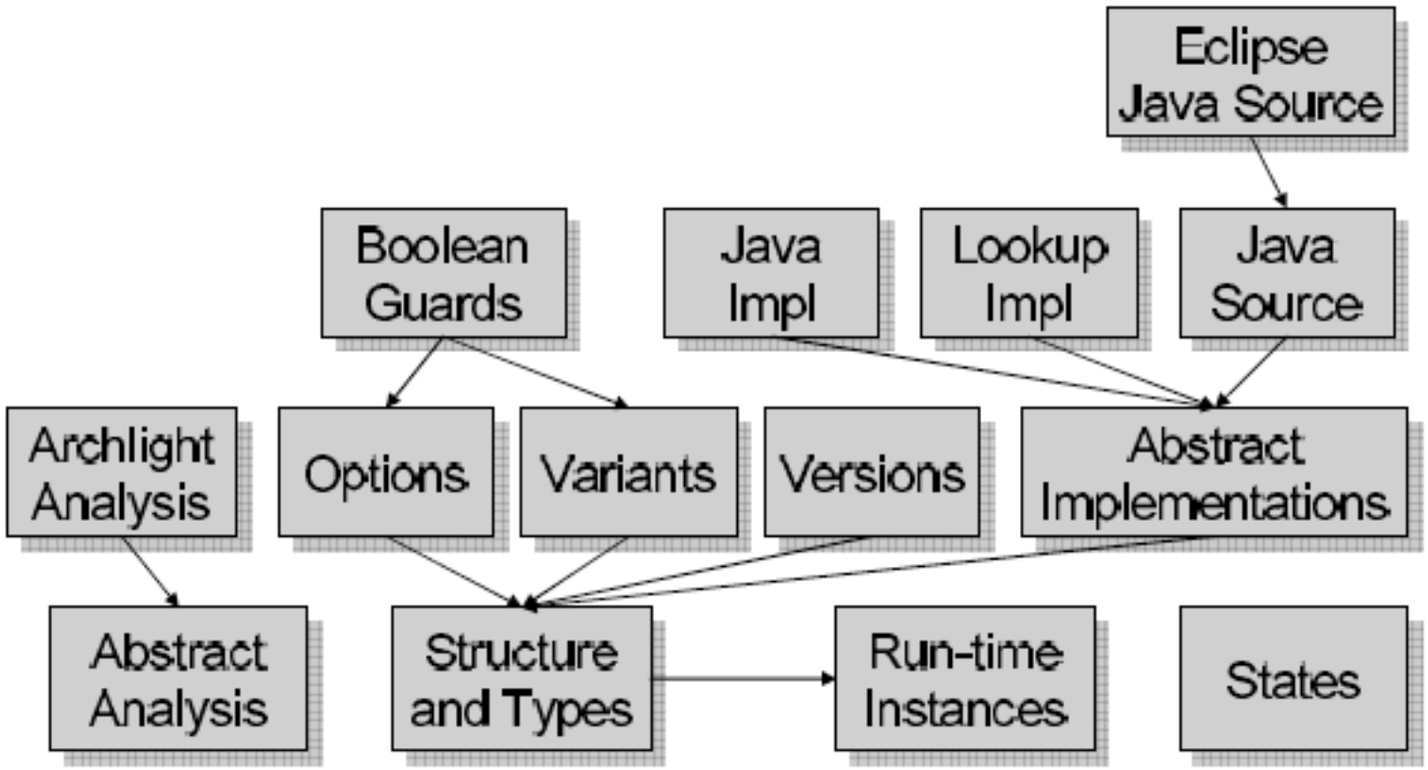
- Generic approaches
  - Natural language, PowerPoint-style modeling, the Unified Modeling Language (UML)
- Early architecture description languages (ADLs)
  - Darwin, Rapide, Wright, ...
- Domain-specific ADLs
  - AADL, Koala, ...
- Extensible ADLs
  - xADL, Acme, ADML, ...

Extensible ADLs

- There is a tension between
  - The expressiveness of general-purpose ADLs and
  - The optimization and customization of more specialized ADLs
- How do we get the best of both worlds?
  - Use multiple notations.
    - Difficult to keep the different models consistent.
    - Certain concerns may not be captured by any notation.
  - Add additional features we want to an existing ADL.
    - Existing ADLs provide little or no guidance for this.
- Extensible ADLs attempt to solve this problem.

xADL

- An Extensible XML-based ADL that was created and evolved at UC Irvine.
- Supports stakeholder-driven modeling.
  - Modeling architectures in a way that the architecture modeling notation can be defined and re-defined at the meta-language level (i.e. syntax design) by users.
  - Most modeling notations are defined at the meta-language level without internal partitions or segmentations, i.e. monolithic design.
- The xADL language is defined by a set of XML schemas.
  - Each schema adds a set of features to the language.
  - No features or schema definitions are privileged over any other.



xADL Schemas and Dependencies.

Instances (xArch): http://isr.uci.edu/projects/xarch/schema.html

Other extensions: http://isr.uci.edu/projects/archstudio-4/www/xarchuci/ext-overview.html

xADL Schemas

- Instance (defined by xArch)
  - Models run-time component, interface, and link instances.
  - Specifies that the root element of the xADL model must be "<*xArch*> ... </*xArch*>".
- Structure & Types
  - Models design-time components, connectors, interfaces, and links, component, connector, and interface types.
  - Defines the first-level elements <*archStructure*> ... </*archStructure*>, <*archTypes*> ... </*archTypes*>.
- Other extensions

Features of xADL

- **Extensibility**: leverages XML's (and XML Schema's) extensibility mechanisms to create a domain-specific ADL factory.
- **Modular Language Design**: xADL is not defined in one big block of meta-language. Instead, a set of meta-language modules are composed into the final xADL.
- **Tool Support**: supported by a variety of visualization, analysis, and utility tools in the ArchStudio environment.

```
<xsd:complexType name="Component">
  <xsd:sequence>
    <xsd:element name="description" type="archinstance:Description"/>
    <xsd:element name="interface"   type="Interface"
                 minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="type"        type="archinstance:XMLLink"
                 minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="id"  type="archinstance:Identifier"/>
</xsd:complexType>
```

XML Schema of a simple xADL component.

```
<component id=”comp0001”>
    <description>Test</description>
    <interface id=”intf0001”>
      <description></description>
      <direction>in</direction>
    </interface>
</component>
```

xADL Description of a component (with namespace and XML typing information removed).

```
<complexType name="Database" abstract="true"/>

<complexType name=”RelationalDatabase”>
  <complexContent>
    <extension base=”Database”>
      <sequence>
        <element name=”tableName” type=”string”/>
        <element name=”numReplicas” type=”int”/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name=”FileDatabase”>
  <complexContent>
    <extension base=”Database”>
      <sequence>
        <element name=”fileName” type=”string”/>
        <element name=”hostName” type=”string”
                 minOccurs=”0” maxOccurs=”1”/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name=”DatabaseComponent”>
  <complexContent>
    <extension base="Component">
      <xsd:sequence>
        <xsd:element name=”database” type=”Database”/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

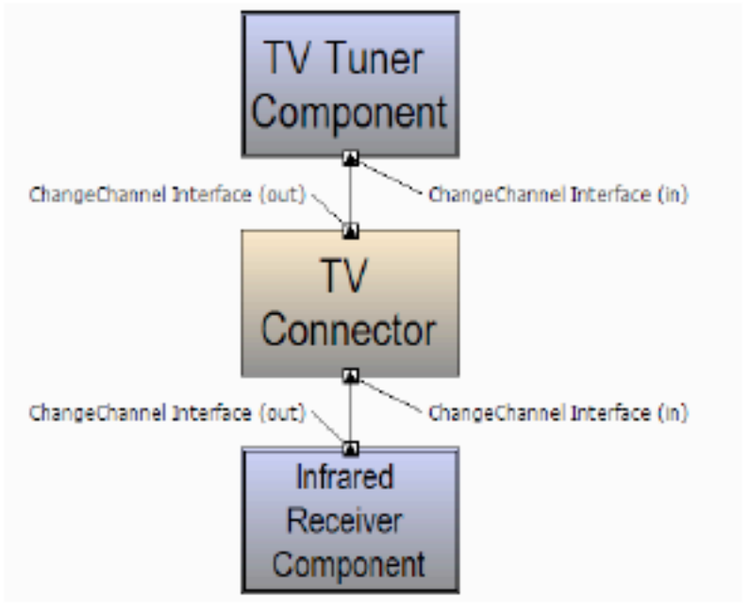
Schema.

An example of xADL extension.

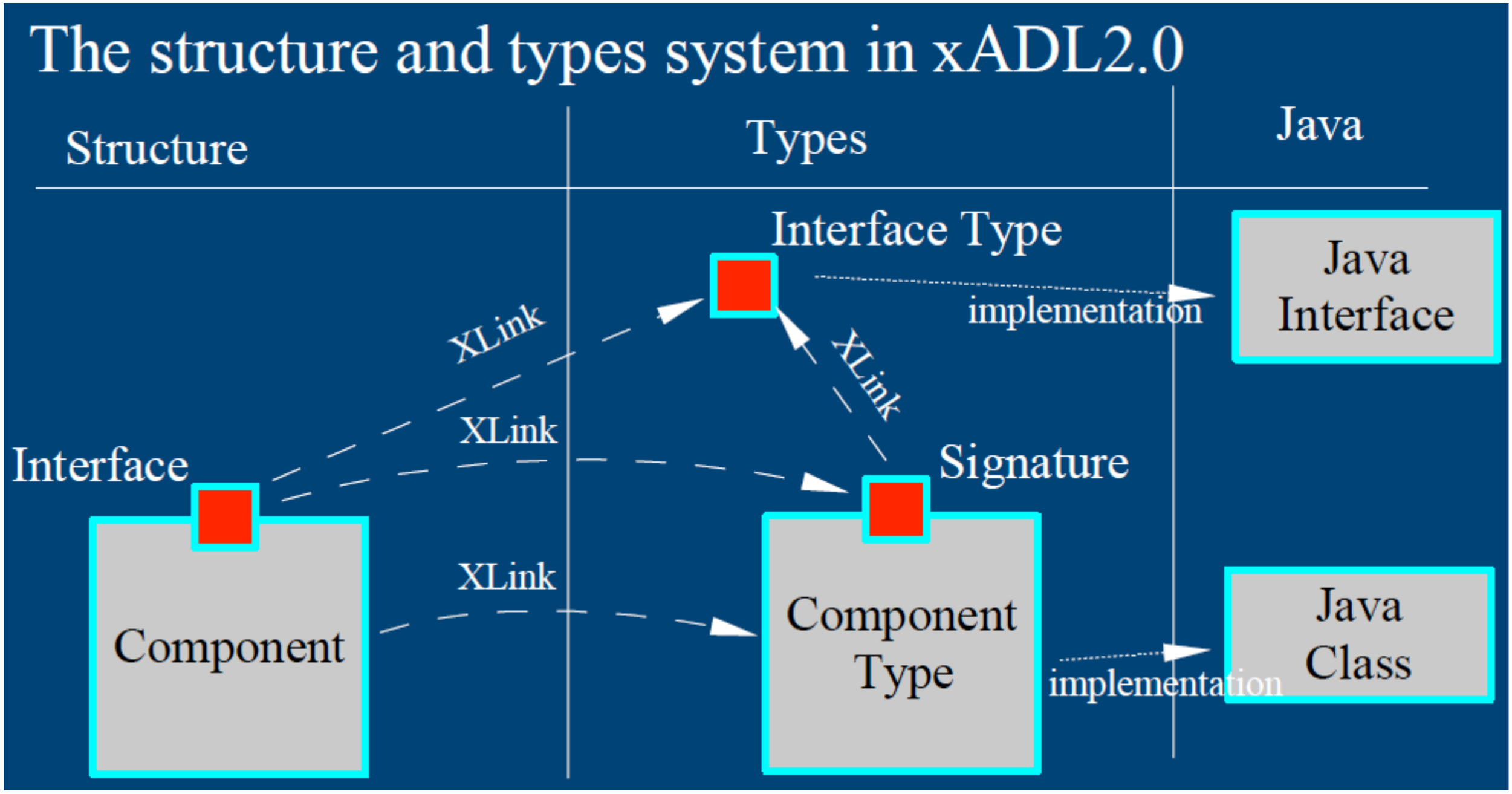
```
<component id="comp0002" type="DatabaseComponent">
  <description>Data Store<description>
  <interface id="intf0002">
    <description>Get Values Interface</descritpion>
    <direction>in</direction>
  </interface>
  <database type="RelationalDatabase">
    <tableName>data</tableName>
    <numReplicas>30</numReplicas>
  </database>
</component>
```

xADL Specification.

```
<xArch>
  <archStructure id="tvset">
    <description>TV Set</description>
    <component id="tuner">
      <description>
        TV Tuner Component
      </description>
      <interface id="tuner.channel">
        <description>
          ChangeChannel Interface
          (in)
        </description>
        <direction>in</direction>
      </interface>
    </component>
    <component id="ir">
      <description>
        Infrared Receiver Component
      </description>
      <interface id="ir.channel">
        <description>
          ChangeChannel Interface
          (out)
        </description>
        <direction>out</direction>
      </interface>
    </component>
    <connector id="tvconn">
      <description>
        TV Connector
      </description>
      <interface id="tvconn.in">
        <description>
          ChangeChannel Interface
          (in)
        </description>
        <direction>in</direction>
      </interface>
      <interface id="tvconn.out">
        <description>
          ChangeChannel Interface
          (out)
        </description>
        <direction>out</direction>
      </interface>
    </connector>
```




```
    <link id="link1">
      <description>
        Tuner to Connector
      </description>
      <point>
        <anchor
          href="#tuner.channel"/>
      </point>
      <point>
        <anchor
          href="#tvconn.out"/>
      </point>
    </link>
    <link id="link2">
      <description>
        Connector to IR
      </description>
      <point>
        <anchor href="#tvconn.in"/>
      </point>
      <point>
        <anchor
          href="#ir.channel"/>
      </point>
    </link>
  </archStructure>
</xArch>
```



Excerpted from Nobu Takeo's MS Final Defense, 2009.

Component
- Identifier
- Description
- Zero or more interfaces
- An optional link to its type

Component Type
- Identifier
- Description
- Zero or more signatures
- Implementations

Interface Type
- Identifier
- Description
- Implementations

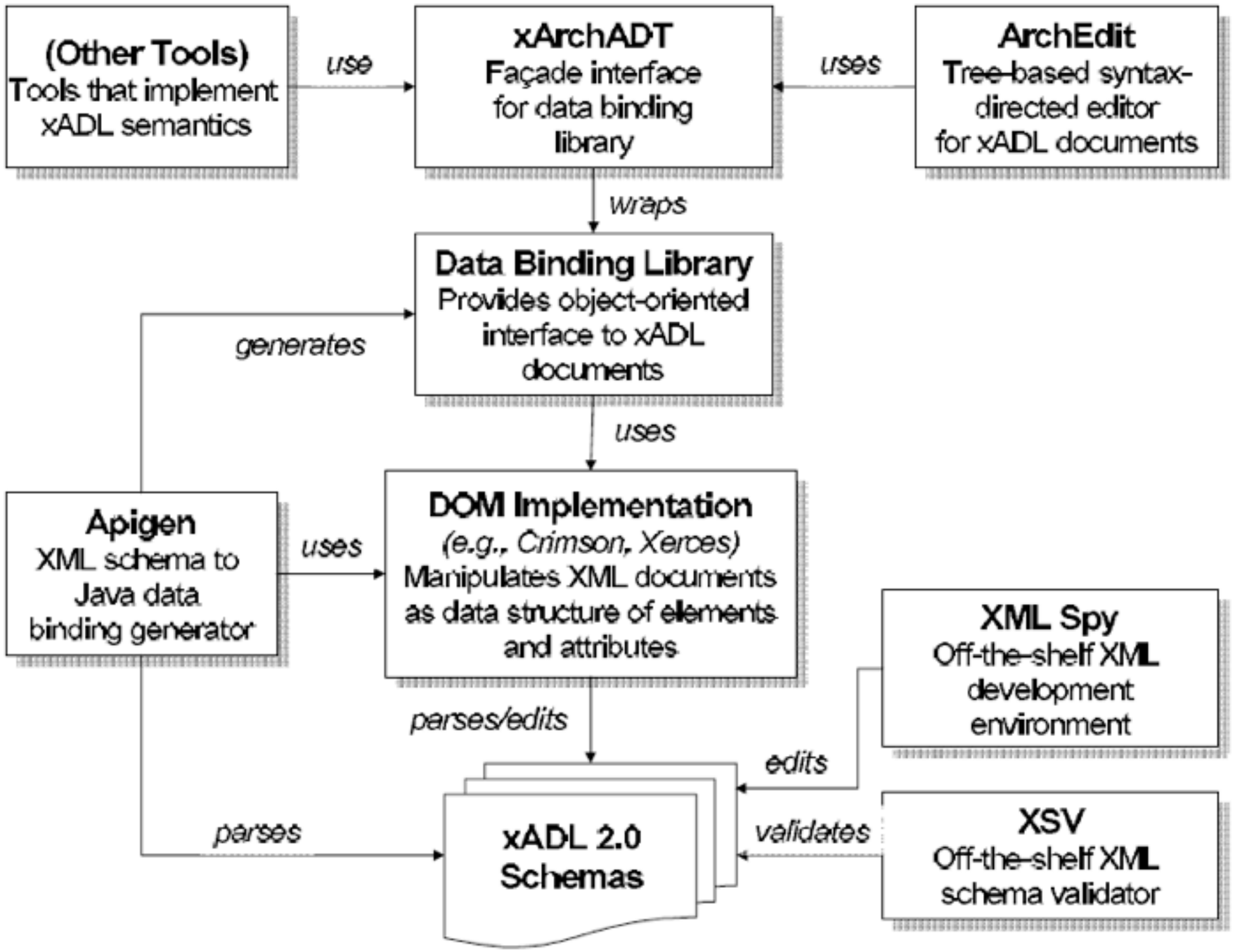
Interface
- Identifier
- Description
- Direction
- Signature
- Type

Signature
- Identifier
- Description
- Direction
- Type

Link
- Identifier
- Description
- Two points

```
<xArch>
  <archStructure>
    ...
  </archStructure>
  <archTypes>
    <componentType id=“tuner_type”>
      <description>TV Tuner Type</description>
      ...
      <implementation>
         <mainClass>
           <javaClassName>edu.uci.isr.tv.TelevisionTuner</javaClassName>
           <url>http://www.example.com/classes/tuner.jar</url>
           <initializationParameter>
              <name>tv_model</name>
              <value>Astro 5000</value>
           </initializationParameter>
         </mainClass>
         <auxClass>
           <javaClassName>edu.uci.isr.tv.TelevisionUtils</javaClassName>
           <url>http://www.example.com/classes/tuner.jar</url>
         </auxClass>
       </implementation>
    </componentType>
  </archTypes>
</xArch>
```

An example of xADL extensions for implementation support.



xADL 2.0 tools and their relationships.

xADL Tools

- Off-the-shelf XML Tools: manipulates xADL documents in terms of low-level XML concepts, like elements and attributes.
- *Data Binding Library:* provides an object-oriented interface that includes functions that are closer to the concepts of software architecture, e.g. addInterface(), removeComponent.
- *Apigen:* automatically regenerates the data binding library when the schema changes.
- *xArchADT:* wraps a simple, one-level API around the object-oriented interface of the data binding library. For example, *xArchADT.add(componentRef, "interface", interafaceRef)*.
- ArchStudio integrated tools.

Evaluation Rubric for xADL

Scope and Purpose	Modeling architecture structure, product lines, and implementations, with support for extensibility.
Basic Elements	Components, connectors, interfaces, links, plus any basic elements defined in **extensions**..
Static & Dynamic Aspects	Static structure is modeled natively, dynamic properties can be captured through **extensions**.
Non-Functional Aspects	**Extensions** can be written to capture non-functional aspects.
Accuracy	**Tools** are provided to check the correctness of xADL documents; additional constraints can be written into these tools to handle **extensions**.
Precision	**Extensions** can be used to annotate existing element types with whatever detail is required or create entirely new first class constructs.
Viewpoints	Structural viewpoints (both run-time and design time) are supported as well as product-line views; **extensions** can be used to provide additional viewpoints.

Resources

- xADL 2.0: http://isr.uci.edu/projects/archstudio-4/www/xarchuci/
- xArch: http://www.isr.uci.edu/projects/xarch/
- ArchStudio: http://www.isr.uci.edu/projects/archstudio/
- Eric Dashofy's Dissertation: http://www.antconcepts.com/~edashofy/files/dashofy-dissertation-2007.pdf