Experiment No.: 2

Visualization using xADL 2.0

Learning Objective: Student should understand and able to do Visualization using xADL 2.0

Tool:- Apigen or Data binding Library.

Theory:

What is xADL 2.0?

xADL 2.0 is a software architecture description language (ADL) developed by the University of California, Irvine for modeling the architecture of software systems. Unlike many other ADLs, xADL 2.0 is defined as a set of XML schemas. This gives xADL 2.0 unprecedented extensibility and flexibility, as well as basic support from the many available commercial XML tools.

The current set of xADL 2.0 schemas includes modeling support for:

- run-time and design-time elements of a system;
- support for architectural types;
- advanced configuration management concepts such as versions, options, and variants;
- product family architectures; and
- architecture "diff"ing (initial support).

xADL 2.0 is also an application of xArch, a core XML schema defined jointly by UCI and Carnegie Mellon University.

xADL 2.0 includes constructs that permit modeling of:

- Architecture structure and types,
- Product families (architectural versions, options, and variants), and
- Implementation mappings (mappings from architecture types to their implementations).

xADL 2.0's modules are defined as XML Schemas, making xADL 2.0 an XML-based language. All xADL 2.0 documents i.e. architecture descriptions are XML documents that are valid with respect to the xADL 2.0 schemas.

xADL 2.0 can be extended by end-users to optimize the language for particular domains. Tools are available that provide users with support for both using existing modules (schemas) and creating and manipulating their own extensions to xADL 2.0.

xADL 2.0 is an XML-based language. XML, the extensible Markup Language, was originally created to annotate, or "mark up" text documents with semantic information. Elements of text are marked up using tags, or special strings, that delimit a section of text. Tags begin with an open angle-bracket (<) and end with a closing angle-bracket (>). In XML documents, tags generally come in pairs, signifying the start and end of a text element. Start tags contain a tag name immediately after the opening angle-bracket, and end-tags contain the same name, prefaced by a forward slash (/) immediately after the opening angle-bracket. Elements may be nested as necessary, but may not overlap. An example of some marked up text in XML might be:

```
<name><first>Herb</first> <last>Mahler</last></name>
```

Constructs Defined in the Instances Schema :

A common theme throughout xADL 2.0, defined first in the instance schema, is that of IDs and Descriptions. Many elements in xADL 2.0 have an ID and/or a Description. Identifiers are assumed to be unique to a particular document. They do not necessarily have to be human-readable, although it helps if they are. Descriptions are intended to be human-readable identifiers of the described constructs.

Furthermore, the instance schema defines an element type called an XMLLink that is used over and over again in other xADL 2.0 schemas. XMLLinks are links to other XML elements. xADL 2.0 borrows the linking strategy from the XLink standard. However, because of poor support for the XLink standard in terms of real tools, xADL 2.0 document authors are advised to follow the following simplified convention for specifying XMLLinks.

In xADL 2.0, anything with an ID can be the target (i.e. the thing being pointed to) for an XMLLink. Every XMLLink has two parts (implemented as XML attributes), type and href; these are defined by the XLink standard. For xADL 2.0 XMLLinks, the type field should always be the string simple, indicating a simple XLink. The href field should be filled out with a URL such as:

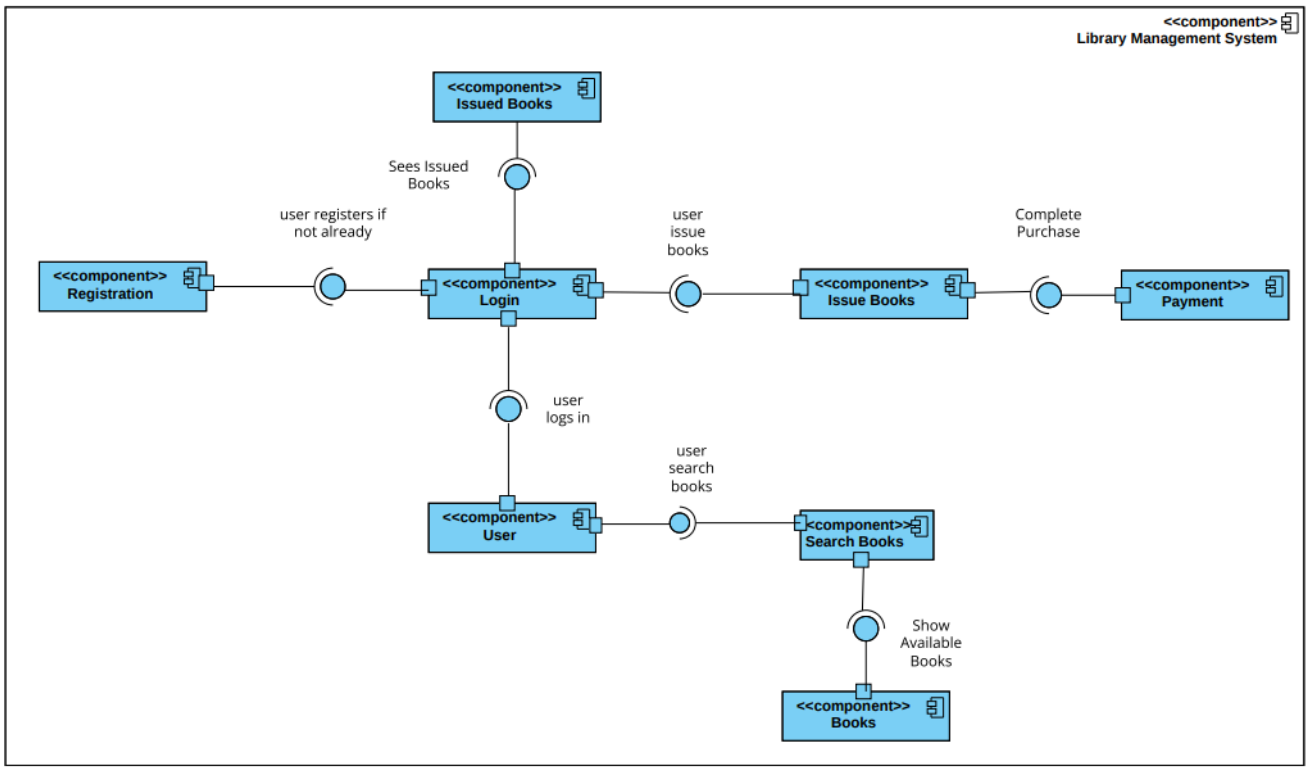
http://*server*/*directory*/*document*.xml#*id*

This is a fairly standard fully-specified URL, linking to a document, but using the *anchor* part of the URL (i.e. the part after the pound sign ('#')) to indicate the identifier of the specific target element. Of course, if you are linking to an element in the same document, it is often preferable to link using a relative URL, such as:

#*id*

Which would be the element with ID *id* in the current document. So, two examples of valid hrefs might be:

http://www.isr.uci.edu/foo/bar/archstudio.xml#ArchEdit
#ArchEdit *(from within the file archstudio.xml)*



Result and Discussion:

Learning Outcomes: Students should have be able to understandLO1:

Define xADL 2.0.

LO2: Identify xADL Command.

LO3: Apply xADL Command for desired Output.

Course Outcomes: Upon completion of the course students will be able to do visualization using xADL 2.0.

Conclusion:

Viva Questions:

1. Define xADL 2.0.
2. Explain xADL 2.0.
3. Explain syntax of xADL 2.0.
4. Explain syntax of Constructs Defined in the Instances

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	