

Software Architecture

Ambra Molesini,
`ambra.molesini@unibo.it`

ALMA MATER STUDIORUM—Università di Bologna

Cesena, 3rd October 2007



- 1 Software Architectures
- 2 Architecture Description Languages
- 3 Architectural Styles



Classic Definitions

Kruchten: The Rational Unified Process 1999

An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization—these elements and their interfaces, their collaborations, and their composition



Classic Definitions

Garlan and Shaw 1993

...beyond the algorithms and data structures of the computation; designing and specifying the overall system structure emerges as a new kind of problem. Structural issues include gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives



Modern Definition

Software Architecture in Practice (2nd edition), 2003

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them



So...

- The architecture defines elements: it embodies information about how the elements relate to each other
- This means that architecture *specifically omits* certain information about elements that does not pertain to their interaction
- Thus, an architecture is foremost an *abstraction* of a system that suppresses details of elements that do not affect how they use, are used by, relate to, or interact with other elements



So...

- Every software system has an architecture because every system can be shown to be composed of elements and relations among them
- The behavior of each element is part of the architecture as that behavior can be observed or discerned from the point of view of another element



Architecture Description Languages

- Architecture description languages (ADLs) are used to describe a Software Architecture
- Several different ADLs have been developed by different organizations:
 - Wright (developed by Carnegie Mellon),
 - Acme (developed by Carnegie Mellon),
 - xADL (developed by UCI),
 - Darwin (developed by Imperial College London)
- Common elements of an ADL are *component*, *connector* and *configuration*



ACME

- Common interchange format for architecture design tools
- Provides a structural framework for characterizing architectures, together with annotation facilities for additional ADL-specific information
- This scheme permits subsets of ADL tools to share architectural information that is jointly understood, while tolerating the presence of information that falls outside their common vocabulary

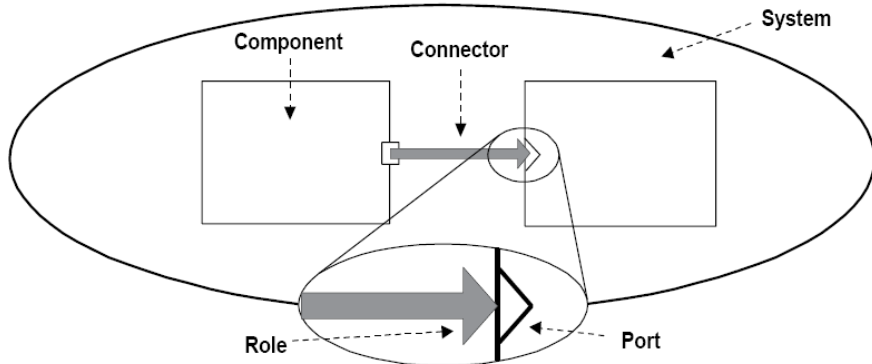


ACME: key elements

- *Components* represent the primary computational elements and data stores of a system
- *Connectors* represent interactions among components.
 - connectors mediate the communication and coordination activities among components
 - they provide the **glue** for architectural designs
- *Systems* represent configurations of components and connectors
- *Port* of a component identifies a point of interaction between the component and its environment.
- *Role* of a connector defines a participant of the interaction represented by the connector.



ACME: Example 1

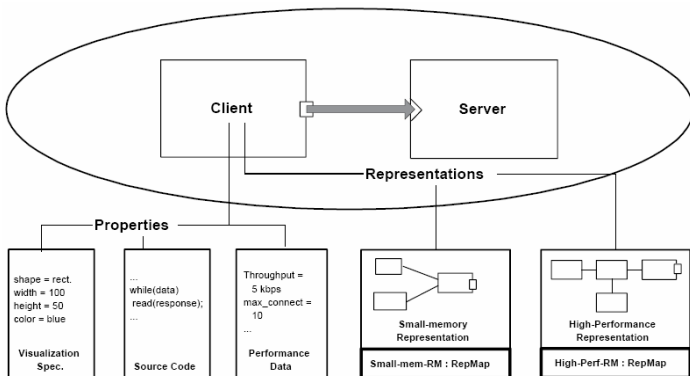


ACME: key elements

- *Attachment* represents the topology of the system
- *Representation* is a more detailed, lower-level descriptions of component or connector.
- A *Rep-map* (*representation map*) indicates the correspondence between the internal system representation and the external interface of the component or connector
- *Properties* are annotations of architectural structure



ACME: Example 2



ACME: Example 3

```
System simple_cs = {  
  Component client = {  
    Port send-request;  
    Properties { Aesop-style : style-id = client-server;  
                 UniCon-style : style-id = cs;  
                 source-code : external = "CODE-LIB/client.c" }}  
  
  Component server = {  
    Port receive-request;  
    Properties { idempotence : boolean = true;  
                 max-concurrent-clients : integer = 1;  
                 source-code : external = "CODE-LIB/server.c" }}  
  
  Connector rpc = {  
    Roles {caller, callee}  
    Properties { synchronous : boolean = true;  
                 max-roles : integer = 2;  
                 protocol : Wright = "..."}  
  
  Attachments {  
    client.send-request to rpc.caller ;  
    server.receive-request to rpc.callee }  
}
```



xADL

- Based on XML
- xADL 2.0 is based on xArch, a core representation for basic architectural elements that uses the XML schema extension method for extending this core.
- xADL makes a clear distinction between architectural *prescription* (the design-time template that is used for instantiating the architecture) and architectural *description* (which describes the run-time state of the system).



xADL: dependencies

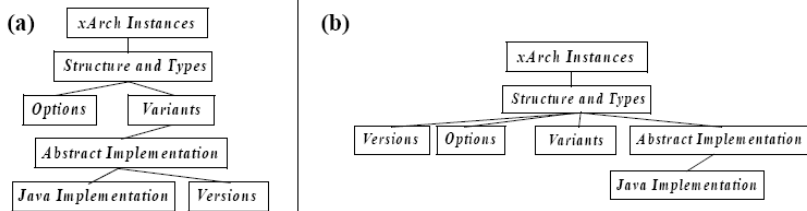


Fig. 1. Actual and conceptual dependencies of the xADL 2.0 XML schemas. Child nodes are dependent on their parent node.



xADL: Schemas

Purpose	Schema	Features Provided
Architecture Modeling - Description and Prescription	xArch Instance	Component, connector, interface, and link instances; arbitrary groups; hierarchical construction.
	Structure & Types	Design-time architectural prescription; architectural structure (components, connectors, interfaces, and links); programming-language style types-and-instances model; hierarchical construction via types.
Instantiatable Architectures	Implementation	Abstract placeholder for implementation information for components and connectors.
	Java Implementation	Java-specific implementation information for components and connectors.
Architecture Configuration Management / Product Family Architectures	Options	Optional components, connectors, and links.
	Variants	Variant component and connector types.
	Versions	Version graphs for components, connectors, and interfaces.

Table 1. xADL 2.0 schemas and features provided.



xADL

- In xADL 2.0, two schemas accomplish the separation of run-time and design-time models
- Constructs modeling run-time aspects of a system are defined in the *INSTANCES* schema, also known as *xArch*
- Constructs modeling design-time aspects of a system are defined in another schema, the *STRUCTURE & TYPES* schema.



xArch: key elements

- The xArch instances schema provides definitions for the following elements typically found in an ADL:
 - Component, connector, interface, and link instances
 - Subarchitectures, for specifying hierarchically composed component and connector instances;
 - Groups, allowing the combination of basic elements into logical aggregations such that they can be identified as a single entity by architectural tools
- The definitions of these elements are semantically neutral: no particular behavior is attached. Constraints on the arrangement and workings of the various architectural elements are specified in xArch extensions.



STRUCTURE & TYPES: key elements

- The STRUCTURE & TYPES schema provides definitions for the following elements:
 - Component, connector, interface, and link
 - Subarchitectures (composite components and connectors with internal architectures)
 - General groups
 - Component and connector types
 - Interface type

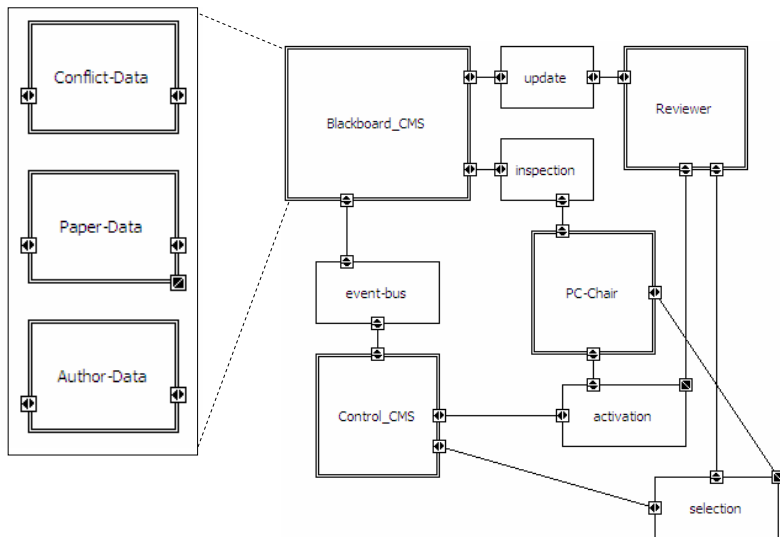


xADL: Example 1

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



xADL: Example 2



Architectural Styles or Architectural Patterns

Architectural Style

defines a family of systems in terms of a pattern of structural organization. More specifically, an architectural style determines the *vocabulary of components and connectors* that can be used in instances of that style, together with a set of constraints on how they can be combined. These can include topological constraints on architectural descriptions

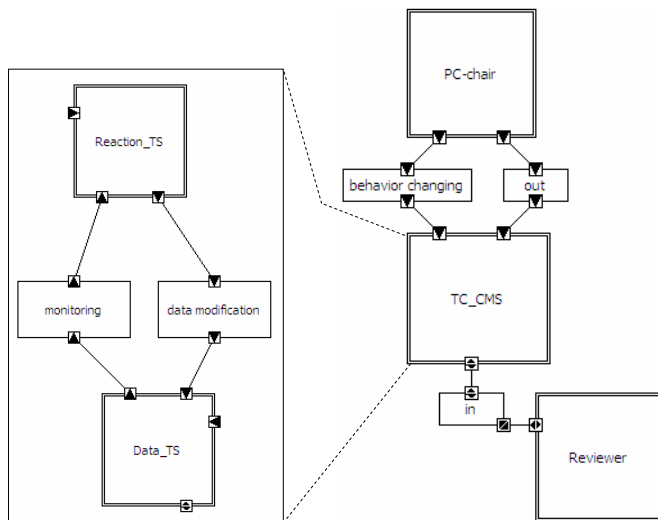


Style Examples:

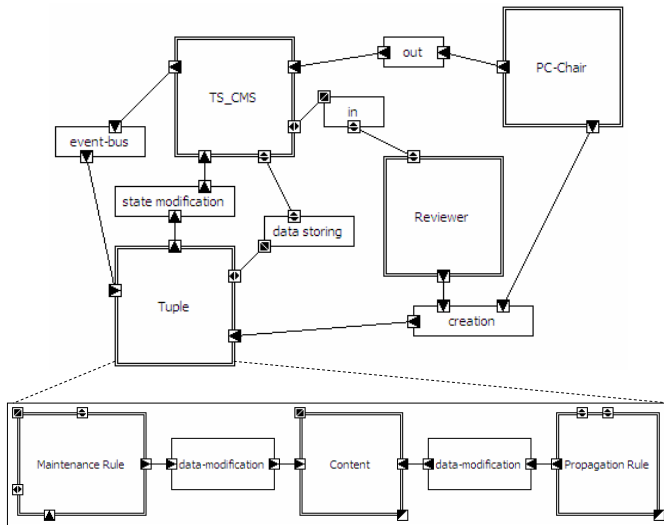
- Pipe and Filter
- Client-Server
- Blackboard
- Layer
- . . .



Reactive Coordination



Stigmergic Coordination



Software Architecture

Ambra Molesini,

`ambra.molesini@unibo.it`

ALMA MATER STUDIORUM—Università di Bologna

Cesena, 3rd October 2007

