On Software Architecture

References

[1] F. Buschmann et al., “A System of Patterns — Pattern-Oriented Software Architecture”, Wiley, 1996, ISBN 0-471-95869-7.
[2] L. Bass et al., “Software Architecture in Practice”, Addison Wesley, 1998, ISBN 0-201-19930-0.

Component (Subsystem)

Definition:

> A *component* is an encapsulated part of a software system. A component has an interface. Components serve as the building blocks for the structure of a system. At programming language level, components may be represented as modules, classes, objects, or a set of related functions. [1]

Architectural Elements in UML:

- package
- component diagram

Some Terms

Software Architecture

Definition:

> A *software architecture* is a description of the subsystems and components of a software system and the relationship between them. Subsystems and components are typically specified in different views to show the relevant functional and non-functional properties of a software system. The software architecture of a system is an artifact. It is the result of the software design activity. [1]

UML elements:

- package
- component diagram
- deployment diagram: nodes, connections

Non-UML Elements

- processes, threads
- data flows, control flows
- network connections, pipes, data pools

One Categorization of Components:

- processing elements
- data elements
- connecting elements

Another more OO-Biased Categorization

- controller component
- coordinator component
- interface component
- service provider component
- information holder component
- structuring component

Relationship/Connector

Definition:

> A relationship denotes a connection between components. A relationship may be static or dynamic.Static relationships show directly in source code. The deal with the placement of components within the architecture. Dynamic relationship deal with temporal connections and dynamic interaction between components. The may not be easily visible from the static structure of the source code. [1]

Kinds of Relationships:

- USE relationship (static or dynamic)
- Aggregation, composition (IS_COMPONENT_OF, COMPRISES)
- Inheritance
- Implementation
- Object instantiation
- Template instantiation

Programmatic Constructs

- (remote) procedure call
- (remote) method invocation
- synchronous/asynchronous communication

- Deployment view: deployment diagram, i.e., the mapping of the software to hardware and processing nodes

View

Definition:

> A view represents a partial aspect of a software architecture that shows specific properties of a software system. [1]

Views

- state views (=> state diagrams)
- communication view (=> sequence diagrams)

But also

- Conceptual architecture: components, connectors...
- Module architecture: subsystems, modules, exports, imports...
  Java: Package architecture = managing the namespace for class names
- Code architecture: files, directories, libraries, includes...
- Execution architecture: tasks, threads, processes...

UML Views:

- Logical view: object model, conceptual class diagram, sequence diagrams
- Concurrency view: activity diagrams, sequence diagrams, state diagrams
- Component view: components diagram

Properties of Software Architecture

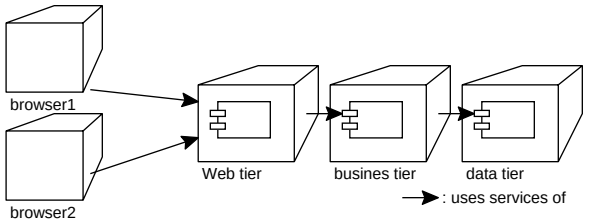
Functional Properties

- Abstraction
  => Layers architectural pattern
- Encapsulation
  => good class design, many patterns
- Information Hiding (Parnas)
  => Composite pattern, use of interfaces in stead of implementations, factories
- Separation of Concern
  => Good choice of responsibility assignment to classes
  => Strategy pattern, Template Method pattern
- Coupling
  => Strength/inflexibility of the association between one module (class/object/component) and the other
- Cohesion
  => Degree of relatedness of the offered functions or services of a module (class/object/component)
- Separation of Interface and Implementation
  => Implementations can be replaced, Bridge pattern, factories

Non-Functional Properties

- Changeability
- Interoperability
- Efficiency
- Reliability
  => fault tolerance, robustness
- Testability
- Reusability

Multi Tier



Architecture — What?

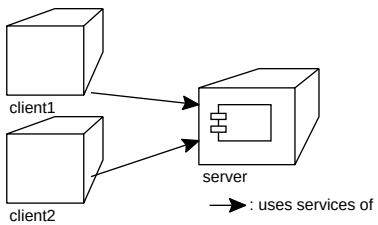
Architectural Style

Definition:

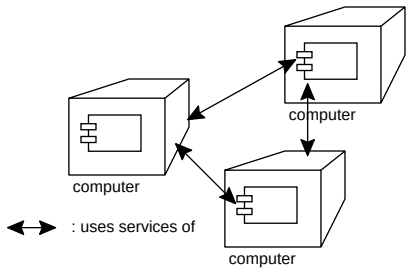
An architectural style is a description of the component types and pattern of their run-time control and/or data transfer. [2]

Examples:

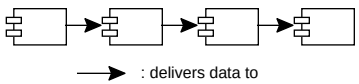
Client/Server



Peer to Peer



Dataflow



Reference Model

Definition:

A reference model is a division of functionality together with data flow between the pieces. [2]

Examples:

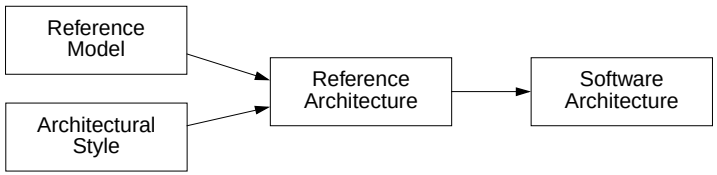
- Communication protocol stacks (TCP/IP, OSI)
- Interactive applications: presentation layer, application logic (tool layer, business object layer), persistence layer

Reference Architecture

A reference architecture is a reference model mapped onto software components (that will cooperatively implement the functionality defined in the reference model) and the data flows between the components. [2]

Software Architecture

The relationship between architectural style, reference model, and reference architecture, is best illustrated with the following figure [2]:



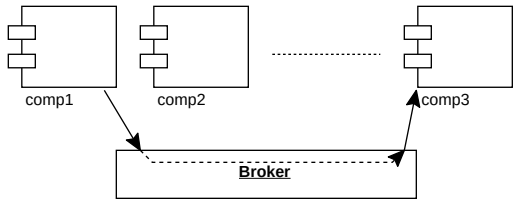
Some Architectural Patterns

Enforcing Structure

- Layers pattern [1]
- Facade [GoF]

Distributing Components

- Broker [1] (for example, CORBA)



- Publish/Subscribe (for example, JMS)