Software Architecture

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

From *Software Architecture in Practice*, Bass, Clements, Kazman, referenced in Garlan
Uses of an Architectural Description

- **Understanding**
  » Abstraction means that we can grasp the major elements in a view and the rationale behind them

- **Reuse**
  » Reusable chunks must be visible to be recognized, extracted, generalized and reapplied to new areas

- **Construction**
  » Some views provide a partial blueprint for development - components and dependencies

More Uses of an Architectural Description

- **Evolution**
  » Expose the “load-bearing walls” of the design and distinguish between components and connectors

- **Analysis**
  » Consistency, performance, conformance

- **Management**
  » Milestone: successful analysis of valid architecture

- **Communication**
  » Stakeholders can prioritize explicit tradeoffs

How to describe an architecture?

- “Boxes and lines”
  » graphical, adaptable, intuitive
  » traditional architecture description

- Some issues
  » meaning of the graphical symbols varies
  » inconsistent or incomplete information
  » difficult to formally analyze for consistency, completeness, correctness
  » constraints are hard to show, enforce

Architectural Description Languages

- Formal notations for representing and analyzing architectural descriptions

- Provide a conceptual framework and concrete syntax for characterizing software architectures
  » also provide tools for parsing, displaying, compiling, analyzing, or simulating the architectural description

- Details of the ADL vary widely depending on the intended application domain
  » Like metrics - useful but judgement required for use
Multiple views

- A key understanding is that *multiple views* of the architecture are valid
  - different stakeholders need to see different things
  - different aspects of the system are best viewed from different points of view
- Code-oriented views
  - modular structure of the system, layers
- Execution-oriented views
  - dynamic configurations, performance, reliability

Entities in an execution-oriented view

- System and Software Components
  - hardware, programs, data blocks
- Connectors
  - mediate interactions among components
- Configurations
  - combinations of components and connectors
- Constraints
  - resource limitations, operating environment

Enterprise Java Bean Examples

- This is the specification of the Enterprise JavaBeans™ architecture.
- The Enterprise JavaBeans architecture is a component architecture for the development and deployment of component-based distributed business applications.
- Applications written using the Enterprise JavaBeans architecture are scalable, transactional, and multi-user secure.
- These applications may be written once, and then deployed on any server platform that supports the Enterprise JavaBeans specification.

Chap 3: Roles and Scenarios

- Discusses the responsibilities of
  - Enterprise Bean Provider (Aardvark, Wombat)
  - Application Assembler (Wombat)
  - Deployer (IT Staff)
  - EJB Container and Server Providers (Acme)
  - System Administrator (IT Staff)
- with respect to the Enterprise JavaBeans architecture.
Module view of deployed application

(c) Wombat's application is deployed in ACME's EJB Container at the ABC enterprise.

6.2.2 What a container provides

The following diagram illustrates the view that a container provides to clients of session beans that provide local and/or remote client views. Note that a client may be a local client of some session beans and a remote client of others.

Client View of session beans deployed in a Container

Inheritance Relationships

State Transition Diagram

A session object does not exist until it is created. When a client creates a session object, the client has a reference to the newly created session object's component interface.
Data Flow Diagrams (DFD)

- DFDs document a process by documenting the flow of data throughout the process.
  - square: external data source or sink
  - arrow: data flow
  - circle: process input data to output data
  - parallel lines: data store

Why do boxes and lines persist?

Boxes and Lines are generally understandable and adaptable