General Software Architecture Models

- Architecture usually a more abstract level of design, but overlaps with design patterns
- Initially we look at software architectures in terms of large scale components without considering the fact that the components may end up distributed on a network
- Distributing an application on a network superimposes extra constraints on the components and adds different cost tradeoffs

Software Architecture Models

- Abstract level of design; between requirements and low-level implementation
- Architectural models support:
  - system-level abstractions
  - composition of large-scale components
  - re-use of design idioms
  - component-based re-engineering, maintenance
  - pre-implementation system evaluation
  - …

Architecture

- the underlying structure of things
- not only *what* but *why*
- good design/good architecture
  - quality: functional and nonfunctional properties
  - applying a set of principles and techniques
  - product lifetime
    - maintainability
    - changeability
  - …

Architectural Model

- An abstract/essential view of a system is as a combination of:
  - functionality (computation)
  - data (state)
  - communication and control

Different architectures

- may locate the functionality and data differently,
- may communicate data in different ways,
- may control the application of functionality to data in different ways.
- …….

Architecture: basic principles

Overview:

- Based partially on:

Mary Shaw, David Garlan: Software Architecture,

(and maybe a little from *Buschmann et al:Pattern-oriented Software Architecture)*

Serves to look at basic principles before looking at the range of architectural models/styles in the various books on architectures

Architectural Style

- An architectural style describes a pattern of structural and/or behavioural organisation
- an architectural style consists of (Shaw, Garlan):
  - a set of component types
  - a set of connector types
  - constraints or properties of construction or behaviour
  - and possibly a semantic model which allows system properties to be derived from properties of the parts

Architectural Style

- Components: the computation and state
  - e.g. objects, databases, filters
- Connectors: interfaces between components; interactions, communications
  - e.g. procedure call, pipes, events
- Properties or constraints: information for construction or analysis
  - e.g. topology: pipe and filter; protocols; pre/post conditions

Architectural Model

Among the main factors determining why different architectures choose different ways to implement and locate functionality, data, communication and control are:

- the characteristics of the problem

  e.g. some characteristics included in Davis taxonomy of computer applications

- non-functional requirements

Taxonomy of Applications(Davis)

1. Difficulty of problem
2. Relationship in time between data and processing
3. Number of simultaneous tasks
4. Relative difficulty of basic aspects of problem
   - data
   - control
   - algorithmic
5. Deterministic versus non-deterministic

Example: Control flow vs. Data flow

- Control flow
  - how locus of control moves through application
  - data may accompany control but is not dominant
  - reasoning about order of computation

- Data flow
  - how data moves through computations
  - as data moves control is activated
  - reasoning about data availability, transformation, latency
  - data flow often between processes

Architecture Characterization

Various different ways an architecture can be characterized

These different characterizations are not independent: they can overlap, and several may apply to a particular implementation

A certain problem with particular constraints will prompt an architecture with particular characteristics

Examples of Architecture Characterization

Structure:

- Hierarchical Layers
- Independent services
- Client and server

Location of data:

- Data flow systems
- Data centered systems
- Encapsulated data (object-oriented) systems

Location of computation:

- Thin client/fat client – server
- Pipe-and-filter

Data Flow System Example

Batch sequential system:

- Processing steps, where each step runs to completion
- Data transmitted as whole between steps
- Applications
  - Classical data processing

Data Flow System Example

Pipes and Filters

Filter

- Incrementally transform some amount of data at inputs to data at outputs
- Use little or no local context in processing
- Preserve no state between instantiations

Pipe

- Move data from filter output to filter input
- Pipes form data transmission graph

Computation

- Run pipes and filters (non-deterministically) until no more computations possible

Data Centered Examples

Repository (Classical database model):

- Clients write/read from centralized data store

Blackboard architecture model

- Blackboard notifies subscribers when data of interest changes
- Computations react to changes in blackboard

Architecture Characterization

Communication type:

Synchronous communication

– Call and Return

Asynchronous communication

– Communicating processes

– Publish-subscribe event systems

– Broadcast event systems

Architecture Characterization

Virtual machines:

- Interpreters
- Rule-based systems

Architecture Characterization

A system may display various characteristics.

A particular system may be have a hierarchical layer structure, may locate its data centrally, may use synchronous communication, and be implemented as a virtual machine interpreter

Note, we still are considering an architecture without emphasizing that the system will be deployed on servers and a network

Architecture Principles

Have seen how an architecture can be characterized in various different ways

In designing an architecture, various aspects

- Particular characteristics of problem
- Non-functional requirements
- Larman's GRASP principles
- ….. more in following lectures