Architecture Styles

EE 382-V

Siddharth Balwani and Khalid Ghori

Overview

- Brief Introduction
- Classification of Architectural Styles
- Style Examples
- Architectural Styles from Component Specifications
- Implementation
- Architecture Style Determination
- Conclusion

Brief Introduction

- What is :
  - Software Architecture
    - Components
    - Connectors
    - Constraints
  - Architectural Style
    - Why?
      - Improve Design Cycle time
      - Quality of Software System
      - Evolution of the System
    - How ?

Classification of Architectural Styles

- Classification categories:
  - Constituent Parts: i.e. components and connectors
  - Control issues
  - Data issues
  - Control/Data Interaction
  - Type of Reasoning

Components and Connectors

- Component: A unit of software that performs a function at run time: e.g. programs, objects, processes and filters
- Connectors: Mechanism that mediates communication, cooperation or coordination among components. eg: shared representations, remote procedure calls, transaction streams
- **Not Enough** to Define a whole Architectural Style

Control Issues

- Topology:
  - Geometric form of control flow of system:
    - Pipeline has a linear topology
    - Server Systems have a Star topology (hub and spoke)
    - Sequential processes have arbitrary topology
- Synchronicity:
  - Individual components dependence on another's control state. eg: SIMD algorithms on parallel machines.
  - Types: Synchronous, Asynchronous, Opportunistic

Control Issues (continued)

- Binding Time:
  - When is identity of collaborating component in transfer-of-control-operation established?
  - Three possibilities: program Write time, Compile Time, Invocation time (when OS initializes process), Run Time

Data Issues

- Topology:
  - Geometric Shape of Systems data flow graph. Same possibilities as Control flow.
- Continuity: 2 dimensions:
  - **Continuous, Sporadic**
  - **High Volume (data intensive), Low Volume (compute)**
- Mode: How data is made available throughout the system
  - **Shared**
  - **Broadcast, Multicast**
  - **Passed around**

Data Issues (continued)

- Binding Time:

  Analogous to issue in Control Flow .

  i.e. When is identity of collaborating component in transfer-of-control-operation established?

Control/Data Interaction Issues

- Shape:
  - Isomorphism in Control/Data topologies
- Directionality:
  - Control flow in:
    - same direction (pipe and filter)
    - or opposite (client-server)

Type of Reasoning

- Type of analysis used:
  - Eg:
    - Non deterministic state machine theory (for asynchronous systems)
    - Function composition (for system executes as fixed sequence of steps)
  - Choice of Architecture may be influenced by kind of analysis required.

Architectural Styles

- 1. Dataflow Styles (Styles dominated by motion of data through system)
  - Dataflow network
    - Component:
    - Connectors:
    - Control Topology:
    - Control Synchronicity:
    - Binding time:
    - Data Topology:
    - Data Continuity:
    - Data Mode:
    - Data binding time:
    - Control/Data Interaction:
    - Flow Direction:
    - Type of Reasoning:

Architectural Styles

- 1. Dataflow Styles (Styles dominated by motion of data through system)
  - Dataflow network
    - Component: Transducer
    - Connectors: Data stream
    - Control Topology: Arbitrary
    - Control Synchronicity: Asynch
    - Binding time: Run time
    - Data Topology: Arbitrary
    - Data Continuity: Continuous, low or high vol
    - Data Mode: Passed
    - Data binding time: Run time
    - Control/Data Interaction: isomorphic shape
    - Flow Direction: Same
    - Type of Reasoning: Functional composition

Architectural Styles

- 2. Call and Return Style (Dominated by order of computation, Single thread of control)
  - Main program/subroutine
    - Component:
    - Connectors:
    - Control Topology:
    - Control Synchronicity:
    - Binding time:
    - Data Topology:
    - Data Continuity:
    - Data Mode:
    - Data binding time:
    - Control/Data Interaction:
    - Flow Direction:
    - Type of Reasoning:

Architectural Styles

- 2. Call and Return Style (Dominated by order of computation, Single thread of control)
  - Main program/subroutine
    - Component: procedures, data
    - Connectors: procedure calls
    - Control Topology: Hierarchical
    - Control Synchronicity: Sequential
    - Binding time: Write/Compile time
    - Data Topology: Arbitrary
    - Data Continuity: Sporadic, low volume
    - Data Mode: Passed and shared
    - Data binding time: Run time
    - Control/Data Interaction: not isomorphic
    - Flow Direction: N/A
    - Type of Reasoning: Hierarchy

Architectural Styles

- **3. Interacting Process Style (Dominated by Communication patterns among independent processes)**
  - Client-Server
    - Component:
    - Connectors:
    - Control Topology:
    - Control Synchronicity:
    - Binding time:
    - Data Topology:
    - Data Continuity:
    - Data Mode:
    - Data binding time:
    - Control/Data Interaction:
    - Flow Direction:
    - Type of Reasoning:

Architectural Styles

- **3. Interacting Process Style (Dominated by Communication patterns among independent processes)**
  - Client-Server
    - Component: Processes
    - Connectors: Request/Reply message
    - Control Topology: Star
    - Control Synchronicity: Synchronous
    - Binding time: Write/Compile/Run time
    - Data Topology: Star
    - Data Continuity: Sporadic, low volume
    - Data Mode: Passed
    - Data binding time: Write/Compile/Run time
    - Control/Data Interaction: Isomorphic
    - Flow Direction: Opposite
    - Type of Reasoning: Non Determinism

Architectural Styles

- 4. Data-centered Repository Style
  - Transactional Database
    - Component:
    - Connectors:
    - Control Topology:
    - Control Synchronicity:
    - Binding time:
    - Data Topology:
    - Data Continuity:
    - Data Mode:
    - Data binding time:
    - Control/Data Interaction:
    - Flow Direction:
    - Type of Reasoning:

Architectural Styles

- 4. Data-centered Repository Style
  - Transactional Database
    - Component: memory, computation
    - Connectors: Queries
    - Control Topology: Star
    - Control Synchronicity: Asynchronous, Opportunistic
    - Binding time: Write time
    - Data Topology: Star
    - Data Continuity: Sporadic, Low volume
    - Data Mode: shared, passed
    - Data binding time: Write time
    - Control/Data Interaction: Isomorphic shape
    - Flow Direction: Opposite
    - Type of Reasoning: ACID (Atomicity, Consistency, Isolation, Durability)

Architectural Styles

- 5. Data sharing Styles
  - Hypertext
    - Component:
    - Connectors:
    - Control Topology:
    - Control Synchronicity:
    - Binding time:
    - Data Topology:
    - Data Continuity:
    - Data Mode:
    - Data binding time:
    - Control/Data Interaction:
    - Flow Direction:
    - Type of Reasoning:

Architectural Styles

- 5. Data sharing Styles
  - Hypertext
    - Component: Documents
    - Connectors: Hyperlinks
    - Control Topology: n/a
    - Control Synchronicity: n/a
    - Binding time: n/a
    - Data Topology: Arbitrary
    - Data Continuity: Continuous
    - Data Mode: Shared
    - Data binding time: Write/Compile/Run time
    - Control/Data Interaction: n/a
    - Flow Direction: n/a
    - Type of Reasoning: Representation

Architectural Styles

- **6. Hierarchical Style (Dominated by reduced coupling, with resulting partition of a system into subsystems with limited interaction)**
  - Virtual Machine
    - Component:
    - Connectors:
    - Control Topology:
    - Control Synchronicity:
    - Binding time:
    - Data Topology:
    - Data Continuity:
    - Data Mode:
    - Data binding time:
    - Control/Data Interaction:
    - Flow Direction:
    - Type of Reasoning:

Architectural Styles

- **6. Hierarchical Style (Dominated by reduced coupling, with resulting partition of a system into subsystems with limited interaction)**
  - Virtual Machine
    - Component: Memory, State machine
    - Connectors: Direct Data Access
    - Control Topology: Hierarchy
    - Control Synchronicity: Synch
    - Binding time: Write/Compile time
    - Data Topology: Hierarchy
    - Data Continuity: Continuous
    - Data Mode: Shared
    - Data binding time: Write/Compile time
    - Control/Data Interaction: No isomorphic shape
    - Flow Direction: n/a
    - Type of Reasoning: Levels of Service

Design Guidance

- Choose appropriate style to fit the requirements (mapping)
- Some simple rules of thumb
  - Problem: Sequential stages decomposition of problem Solution: batch sequential or pipeline architecture
  - Problem: Understanding long lived data, its management & representation. Solution: Data Repository
  - Problem: Computation designed, but no machine to execute it. Solution: Interpreter
  - Problem: Embedded system controlling continued action. Solution: Closed loop Architecture
  - Problem: High Degree of flexibility & with loose coupling. Solution: Interacting Processes

Predicting Architectural Styles from Component Specification

- Idea: Why not use component specifications to determine architectural style
  - Different architectural styles support different quality attributes
  - Quality attributes of a system are determined by system requirements
  - Ability to predict architectural style of a system will helps us determine whether desired quality attributes will be satisfied

Predicting Architectural Styles from Component Specification

- Query a component repository with detailed architectural specifications
- Determine whether components returned by the repository conform to any specific architectural style
- Identify a set of these that conform to a desired architectural style and hence meet desired quality attributes

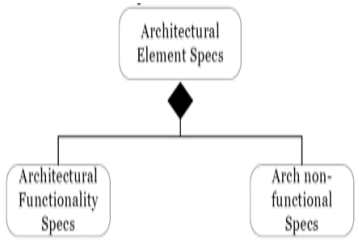
Implementation

- System Integrator identifies list of services that is to be implemented using pre-built components
- Components have been specified using asset specification model. Brief overview of architectural and asset specification models follows.

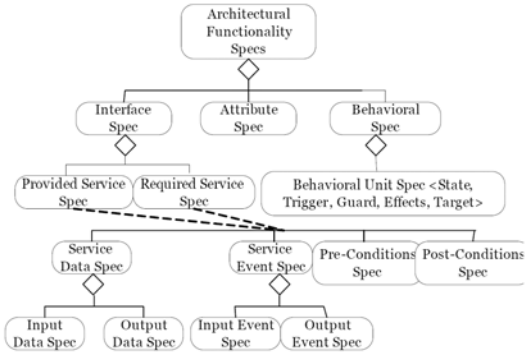
Specification model

- **Architectural element specification :**

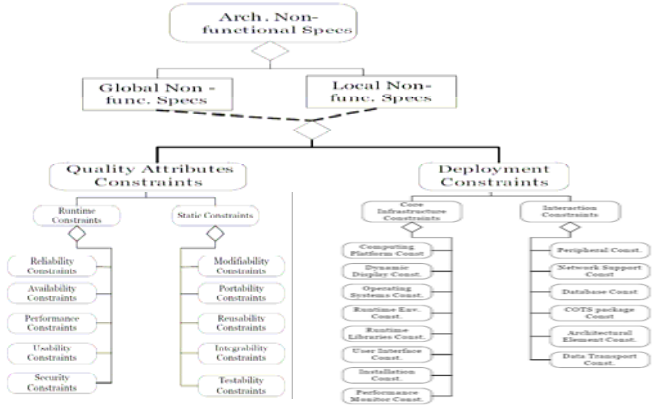
Enables Functional Partitioning and object orientation.



Arch Functionality Specification



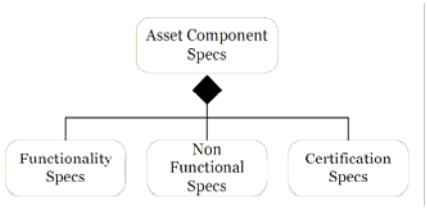
Arch Non-functionality Specification



Specification models

- **Asset component specification**

Asset Specification same as architectural specification except that it has a certification specification describing dependability.



Determining Architectural Style

- Components, Connectors: Returned by database. Does not by itself uniquely identify style.
- Control Factors:
  - Topology:
    1. Select service from scenario list
    2. If last service go to 8
    3. Pick component from repository that is registered to service and has highest value service complaint metric
    4. Add component to control flow (CF) list
    5. For all input events for this components identify components that generates corresponding output events
    6. For all its output events identify the components that consumes them
    7. Goto step 1

Control Factors, Topology

- **Analysis**

8. If all component occur only once in the CF then topology linear
9. If component follow hub-and spoke pattern then control topology is Star
10. If component follow tree like pattern then hierarchical
11. If first element is different from the last in CF then Acyclic
12. Else, Arbitrary

Control Factors, Synchronicity

- Use control flow list (CFL) developed during topology to determine synchronicity
  1. In CFL, if output events of one component are same as input events of next component , this its synchronicity is sequential
  2. In CFL, if list of output events of all preceding components exactly match the input events of next component then synchronous
  3. In CFL, if input events of all components corresponds to output events of same component then synchronicity is opportunistic
  4. In CFL, if synchronicity not determined in steps 1-3 then it is asynchronous

Control Factors, Binding time

- Can not be used for classification

Data Factors

- Topology :
  - Almost same process as Control Factors Topology.
  - In control factors, we identify all components that generate input events and all those that consume output events. This is **NOT TRUE** in data topology. Consumption of all generated data needn't be analyzed.

Data Factors

- Continuity
  - If component requires input data to execute service and generates output data after executing the service, system of components is continuous
  - Else it is sporadic
- Binding time, Mode
  - Do not use for style distinction

Control/Data interaction

- Shape:
  - If control and data topologies are seen to be the same then shape of control data interaction is isomorphic, else non isomorphic
- Directionality
  - Not considered for distinction

Architectural Style Determination

1. System integrator specify scenario or which a software config needs to be built from the services specified in the Architecture Functionality Specs
2. For each service identify the component that is the best fit.
3. Make a note of the type of most common component in Base Component List.
4. Make a note of the type of most common connectors in Base Component List.

Architecture Style Determination

5. Determine Control Topology, develop CFL
6. Determine Control Synchronicity
7. Determine Data Topology, develop DFL
8. Determine Data Continuity
9. Determine Shape of control/data interaction
10. Reference Table to determine which architecture is to be used.

Architecture Style Table

Style	Constituent Parts		Control Issues		Data Issues		Control/Data Interaction
	Components	*Connectors*	*Topology*	*Synchronicity*	*Topology*	*Continuity*	*Isomorphic Shapes*
Data Flow Architectural Styles							
Batch Sequential	Stand-alone programs	Batch data	Linear	Sequential	Linear	Sporadic	Yes
Data-Flow Network	Transducers	Data Stream	Arbitrary	Asynchronous	Arbitrary	Continuous	Yes
Pipes and Filter	Transducers	Data Stream	Linear	Asynchronous	Linear	Continuous	Yes
Call and Return							
Main Program/ Subroutines	Procedure	Procedure Calls	Hierarchical	Sequential	Arbitrary	Sporadic	No
Abstract Data Types	Managers	Static Calls	Arbitrary	Sequential	Arbitrary	Sporadic	Yes
Objects	Managers	Dynamic Calls	Arbitrary	Sequential	Arbitrary	Sporadic	Yes
Call based Client Server	Programs	Calls or RPC	Star	Synchronous	Star	Sporadic	Yes
Layered	-	-	Hierarchical	Any	Hierarchical	Sporadic	Often
Independent Components							
Event Systems	Processes	Signals	Arbitrary	Asynchronous	Arbitrary	Sporadic	Yes
Communicating Processes	Processes	Message Protocols	Arbitrary	Any but Sequential	Arbitrary	Sporadic	Possibly
Data Centered							
Repository	Memory, Computations	Queries	Star	Asynchronous	Star	Sporadic	Possibly
Blackboard	Memory, Components	Direct Access	Star	Asynchronous	Star	Sporadic	No

Table 1: Architectural Style Classification

Conclusion

- Described Division of Architectural Styles. How to identify a style, what it consists of .
- Reasoned about architectural styles using component specifications obtained from a list of components, using system requirements.

Questions?