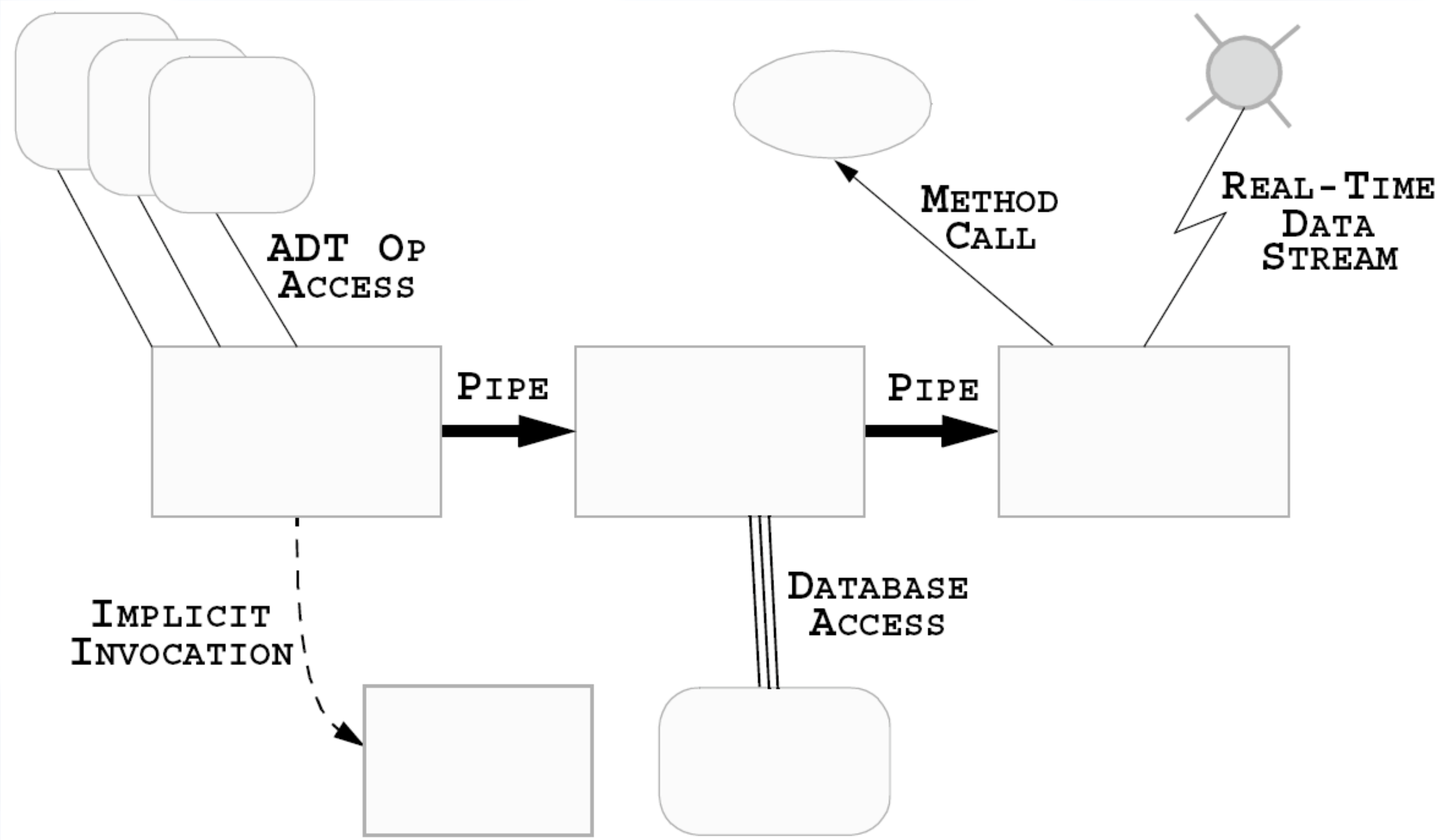
Software Connectors

Software Architecture
Lecture 7

What is a Software Connector?

- Architectural element that models
  - Interactions among components
  - Rules that govern those interactions
- Simple interactions
  - Procedure calls
  - Shared variable access
- Complex & semantically rich interactions
  - Client-server protocols
  - Database access protocols
  - Asynchronous event multicast
- Each connector provides
  - Interaction duct(s)
  - Transfer of control and/or data

Where are Connectors in Software Systems?



Implemented vs. Conceptual Connectors

- Connectors in software system implementations
  - Frequently no dedicated code
  - Frequently no identity
  - Typically do not correspond to compilation units
  - Distributed implementation
    - Across multiple modules
    - Across interaction mechanisms

Implemented vs. Conceptual Connectors (cont'd)

- Connectors in software architectures
  - First-class entities
  - Have identity
  - Describe all system interaction
  - Entitled to their own specifications & abstractions

Reasons for Treating Connectors Independently

- Connector ≠ Component
  - Components provide application-specific functionality
  - Connectors provide application-independent interaction mechanisms
- Interaction abstraction and/or parameterization
- Specification of complex interactions
  - Binary vs. N-ary
  - Asymmetric vs. Symmetric
  - Interaction protocols

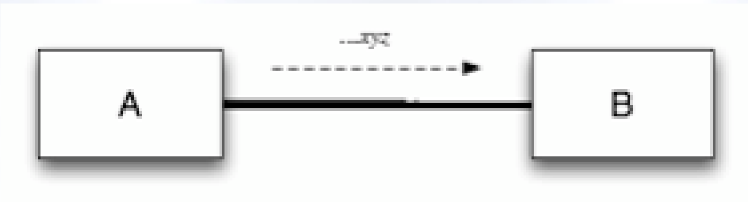
Treating Connectors Independently (cont’d)

- Localization of interaction definition
- Extra-component system (interaction) information
- Component independence
- Component interaction flexibility

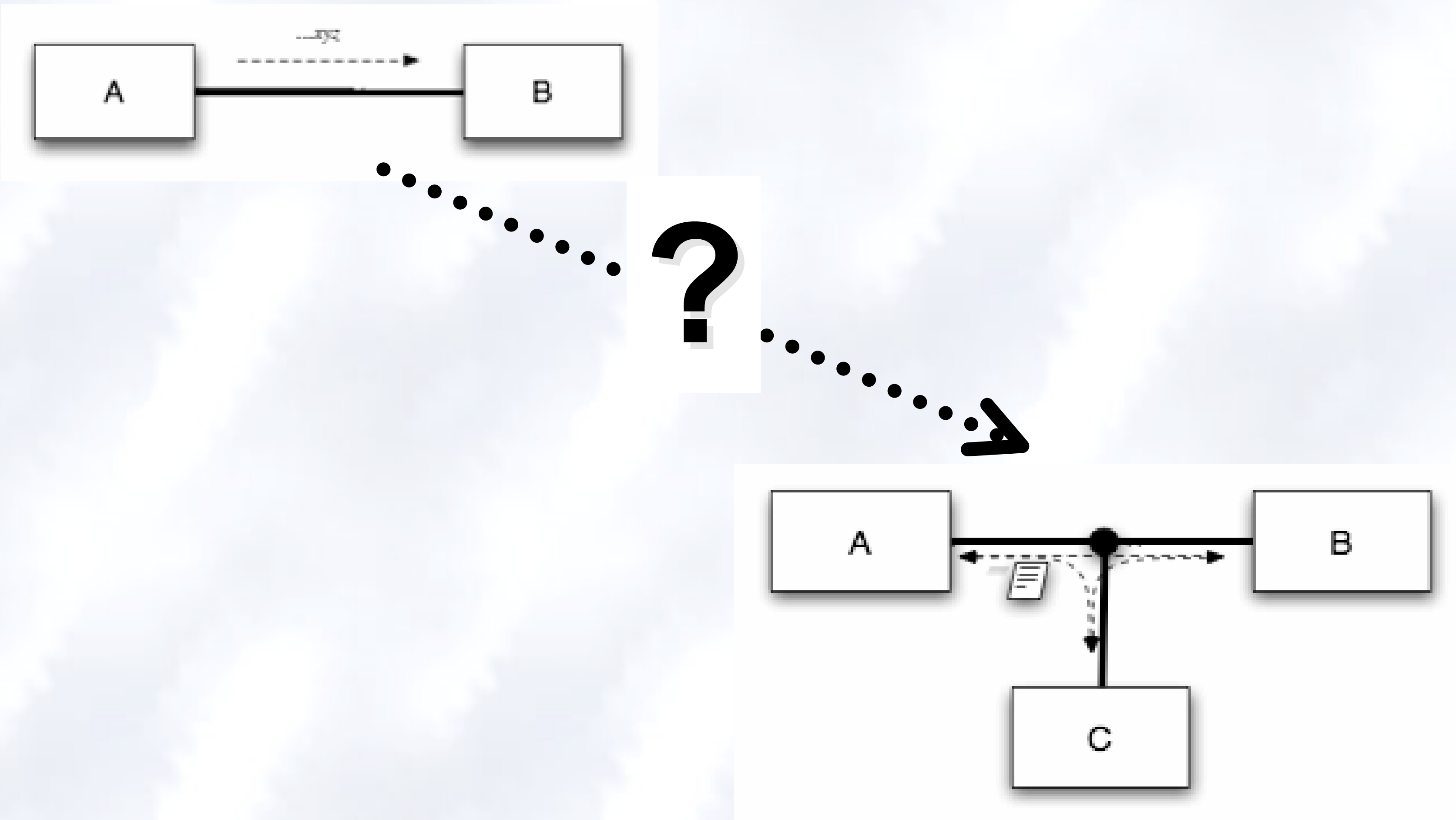
Benefits of First-Class Connectors

- Separate computation from interaction
- Minimize component interdependencies
- Support software evolution
  - At component-, connector-, & system-level
- Potential for supporting dynamism
- Facilitate heterogeneity
- Become points of distribution
- Aid system analysis & testing

An Example of Explicit Connectors



An Example of Explicit Connectors (cont'd)



Software Connector Roles

- Locus of interaction among set of components
- Protocol specification (sometimes implicit) that defines its properties
  - Types of interfaces it is able to mediate
  - Assurances about interaction properties
  - Rules about interaction ordering
  - Interaction commitments (e.g., performance)
- Roles
  - Communication
  - Coordination
  - Conversion
  - Facilitation

Connectors as Communicators

- Main role associated with connectors
- Supports
  - Different communication mechanisms
    - e.g. procedure call, RPC, shared data access, message passing
  - Constraints on communication structure/direction
    - e.g. pipes
  - Constraints on quality of service
    - e.g. persistence
- Separates communication from computation
- May influence non-functional system characteristics
  - e.g. performance, scalability, security

Connectors as Coordinators

- Determine computation control
- Control delivery of data
- Separates control from computation
- Orthogonal to communication, conversion, and facilitation
  - Elements of control are in communication, conversion and facilitation

Connectors as Converters

- Enable interaction of independently developed, mismatched components
- Mismatches based on interaction
  - Type
  - Number
  - Frequency
  - Order
- Examples of converters
  - Adaptors
  - Wrappers

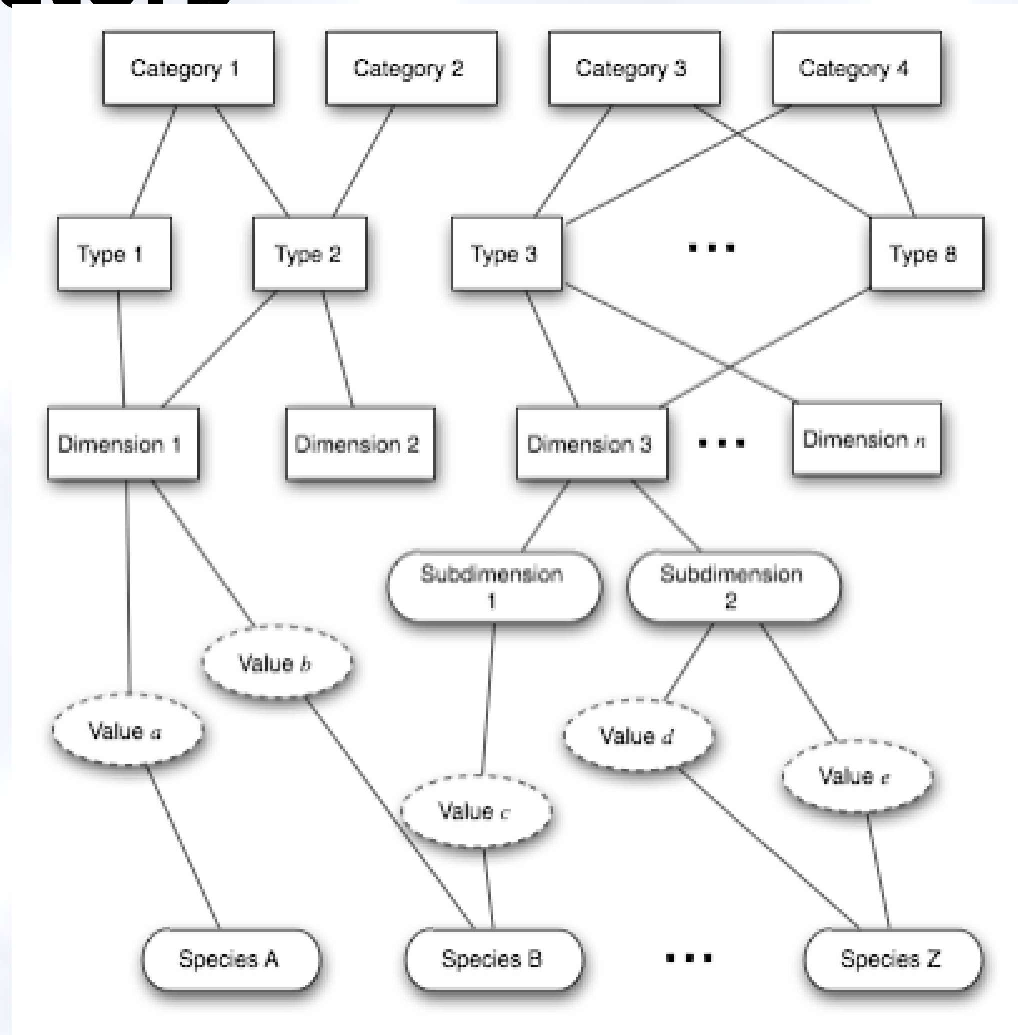
Connectors as Facilitators

- Enable interaction of components intended to interoperate
  - Mediate and streamline interaction
- Govern access to shared information
- Ensure proper performance profiles
  - e.g., load balancing
- Provide synchronization mechanisms
  - Critical sections
  - Monitors

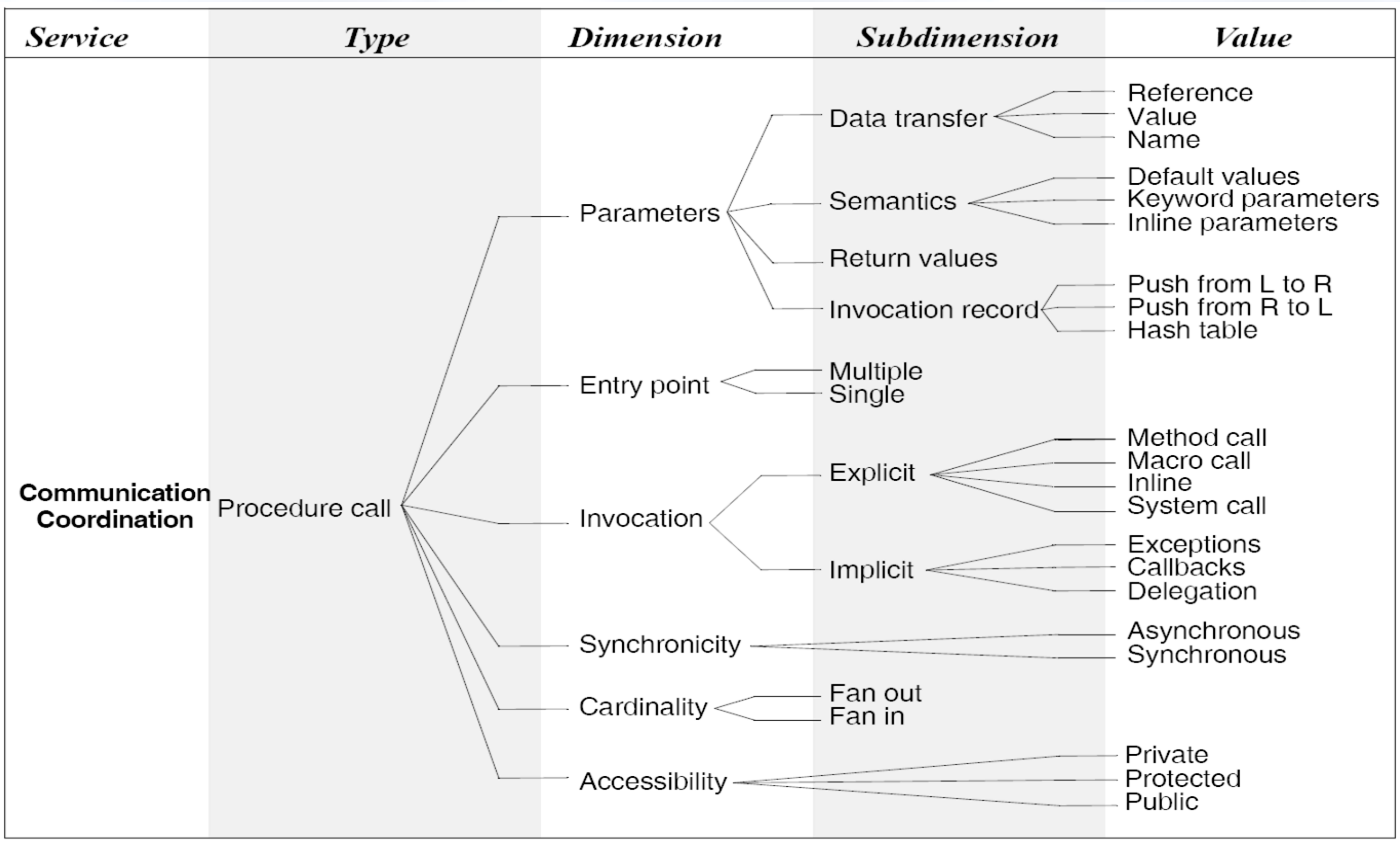
Connector Types

- Procedure call
- Data access
- Event
- Stream
- Linkage
- Distributor
- Arbitrator
- Adaptor

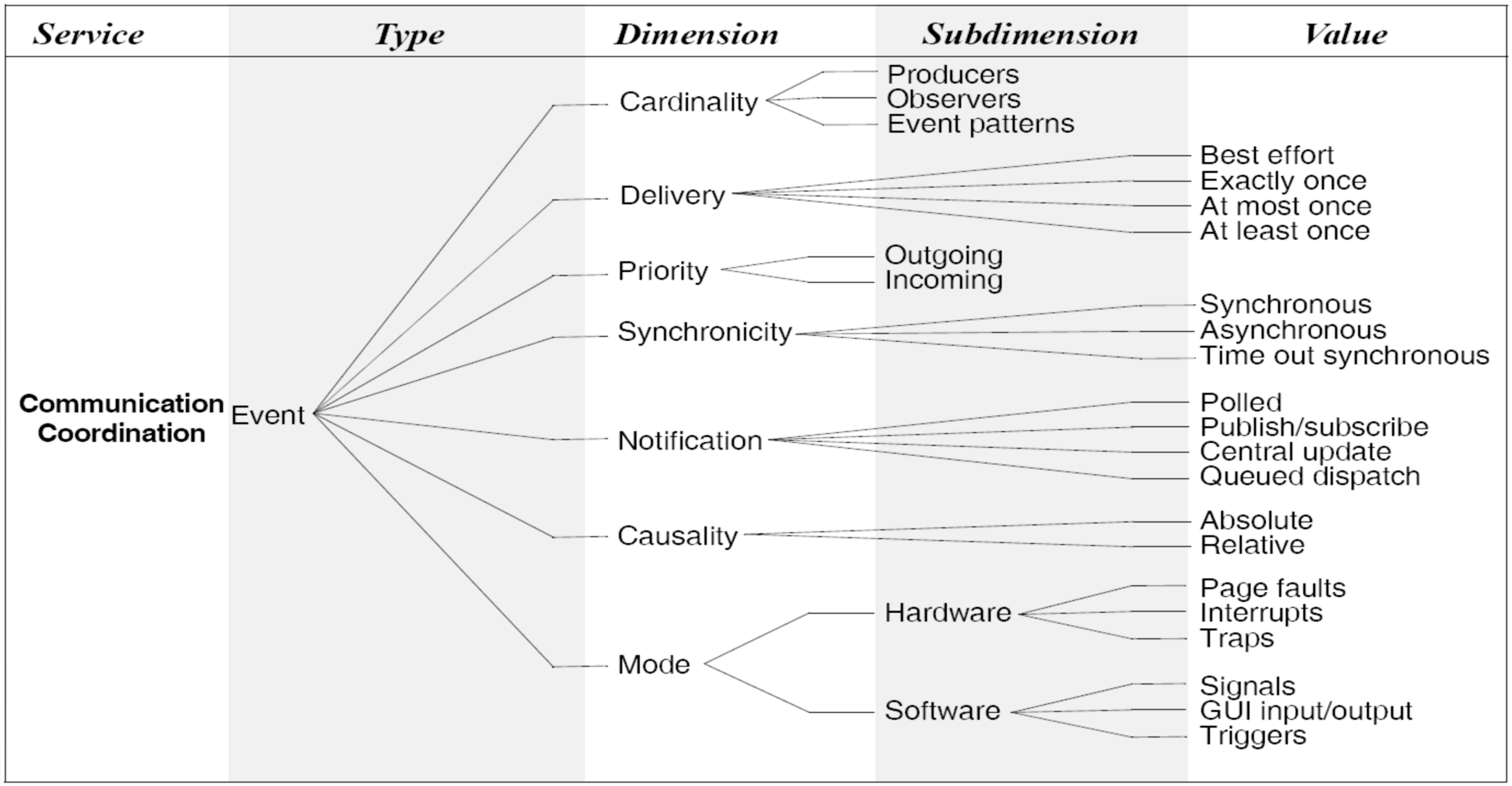
A Framework for Classifying Connectors



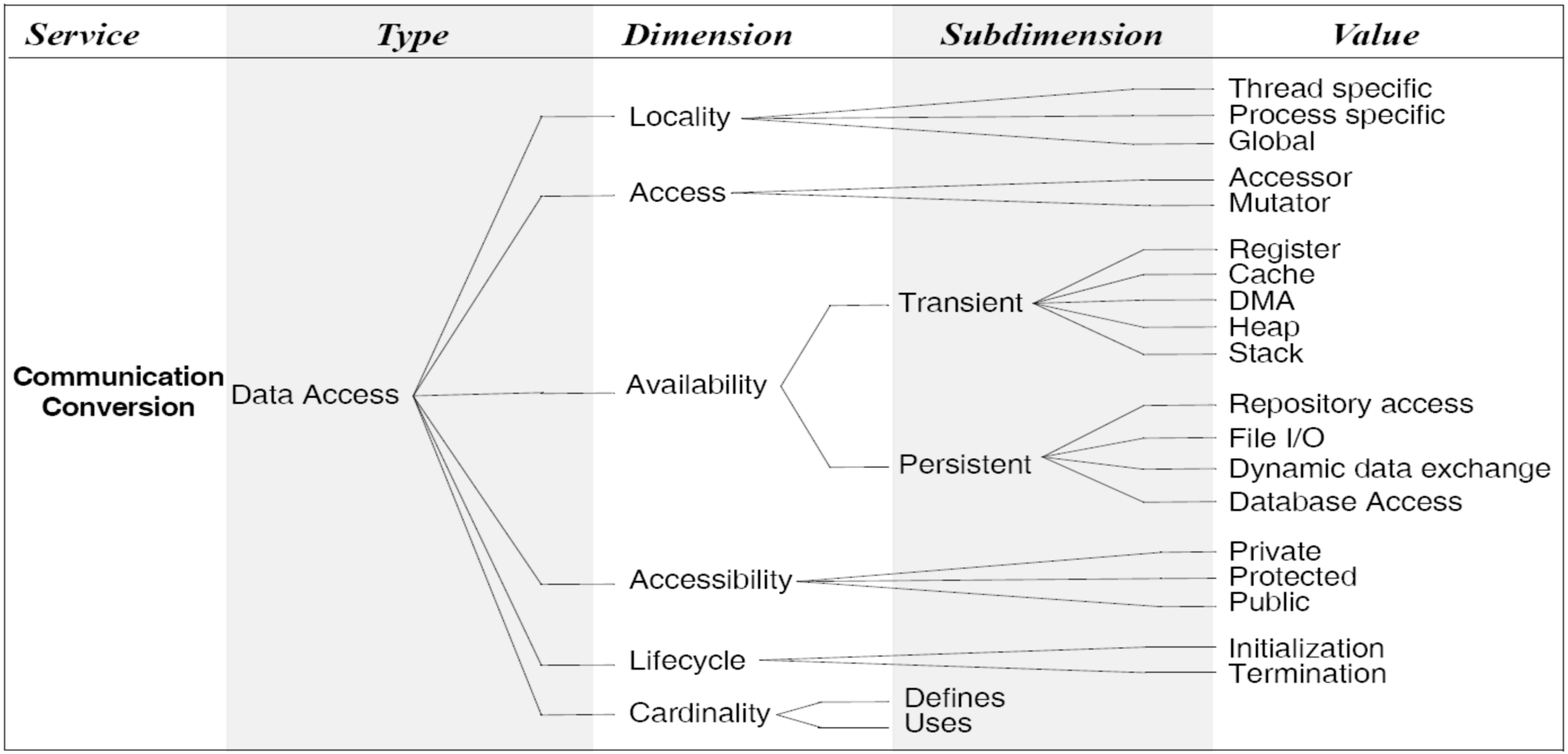
Procedure Call Connectors



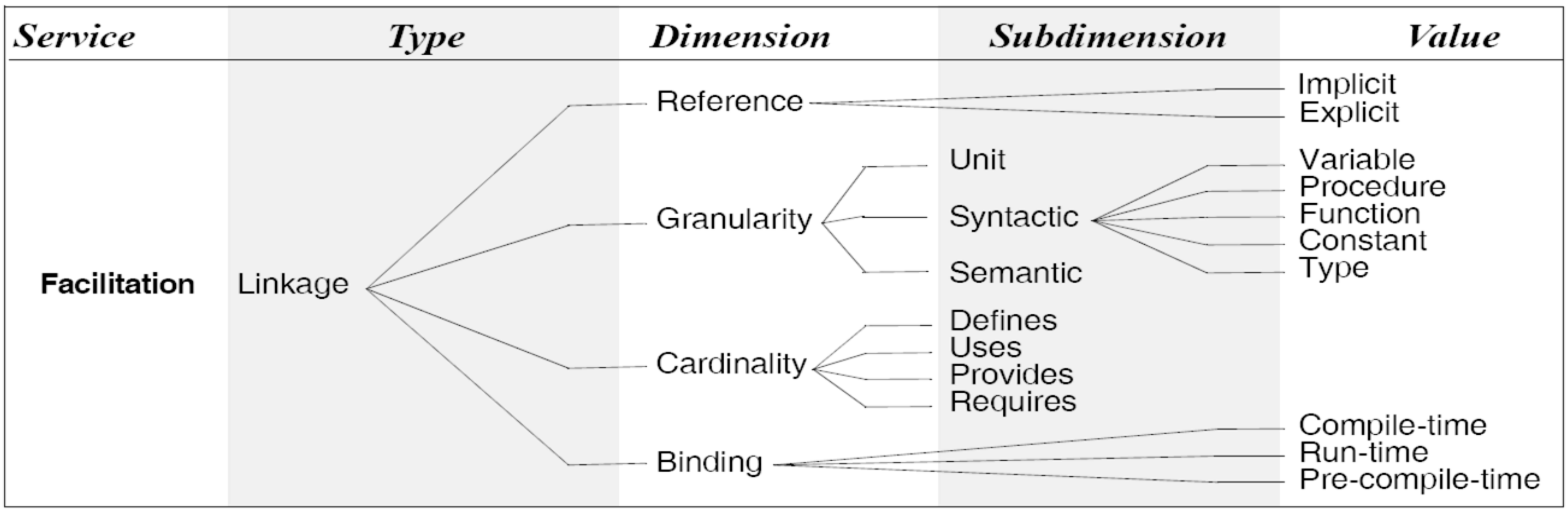
Event Connectors



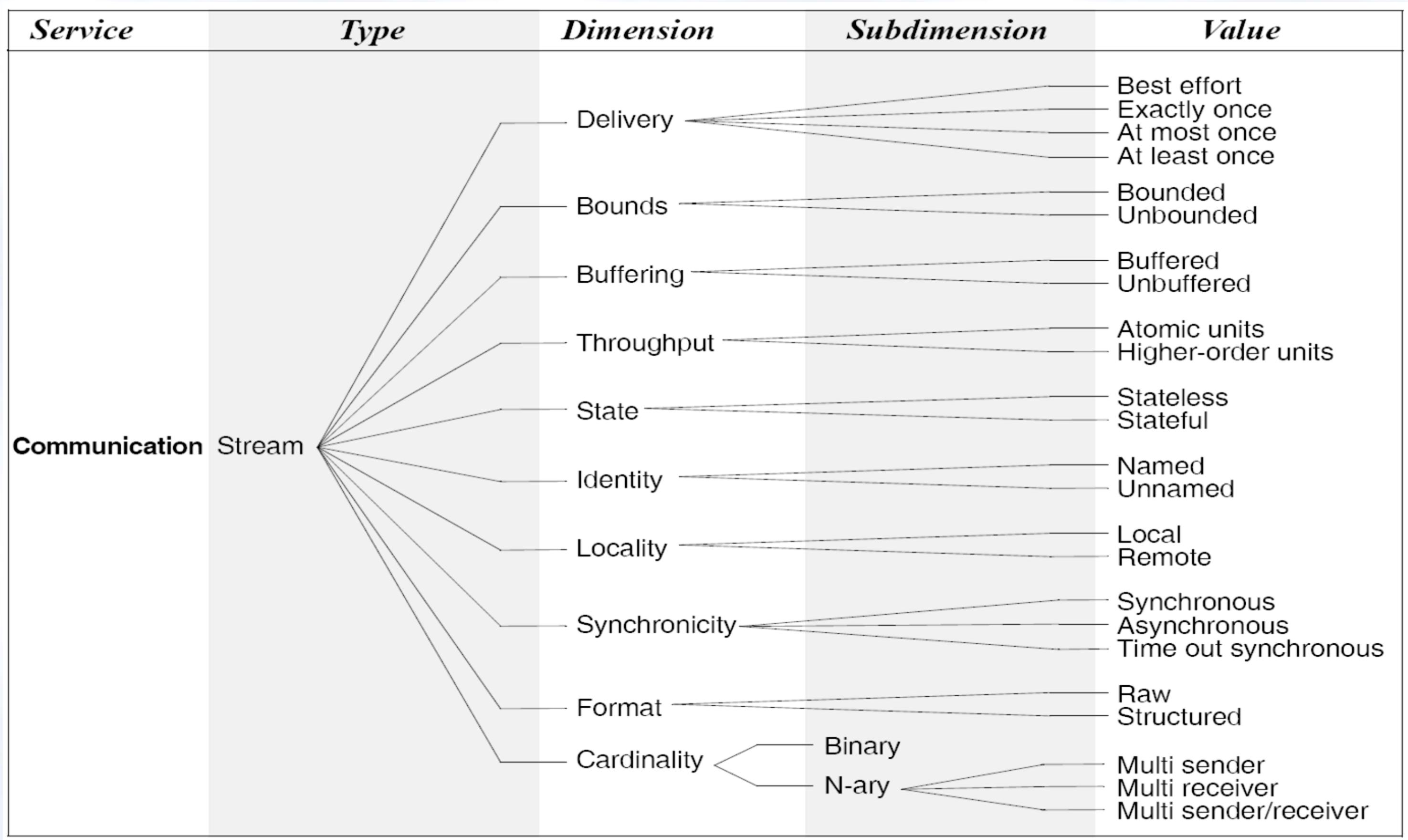
Data Access Connectors



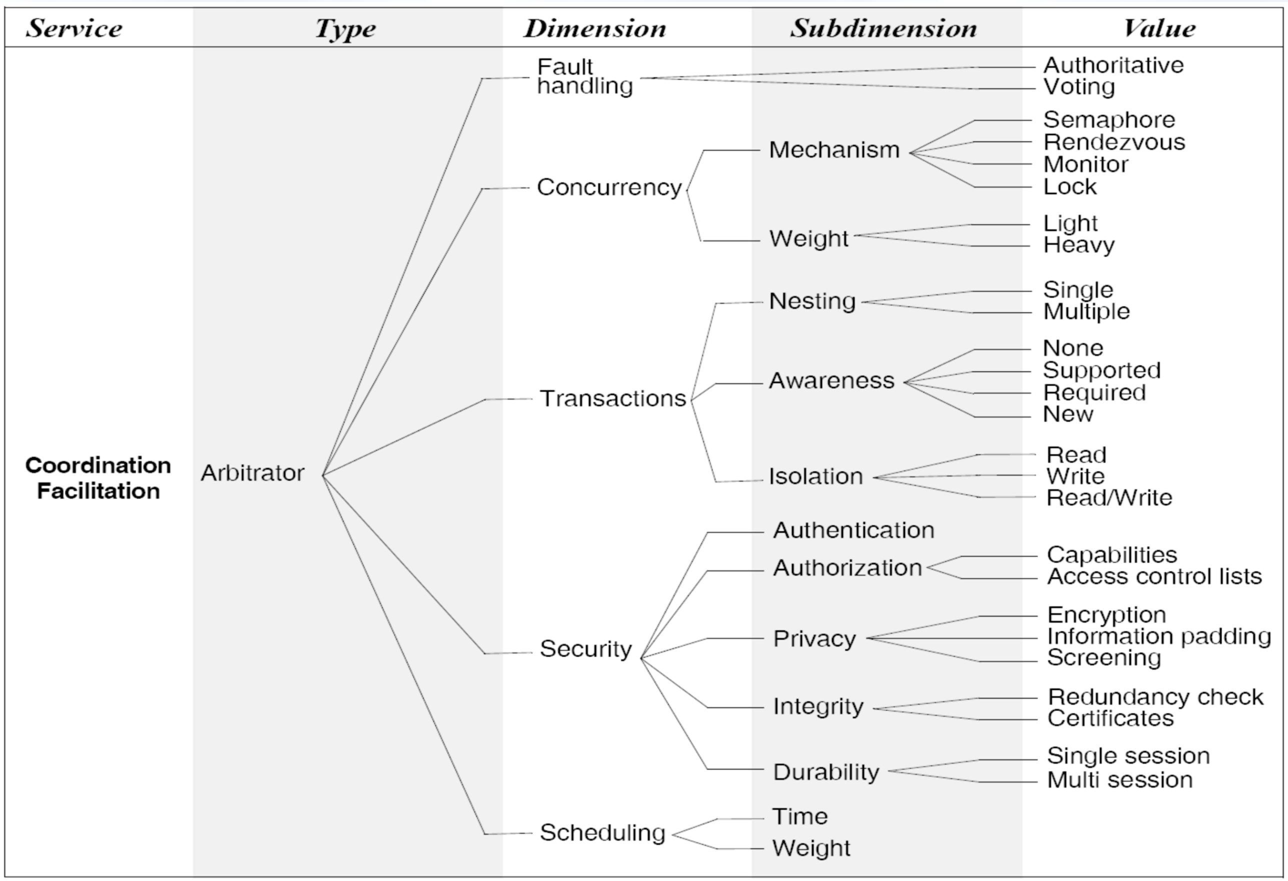
Linkage Connectors



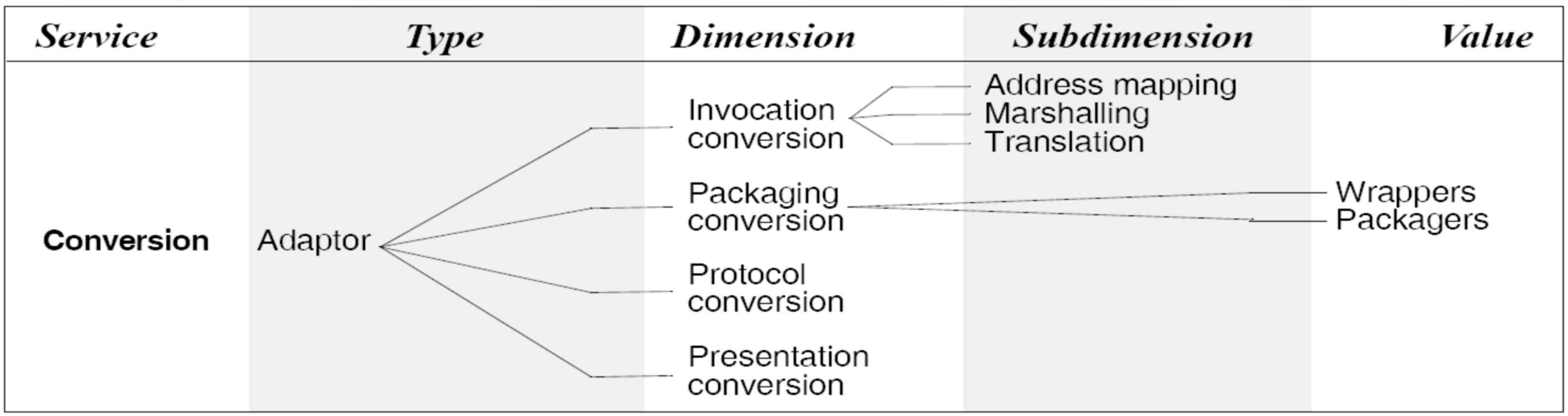
Stream Connectors



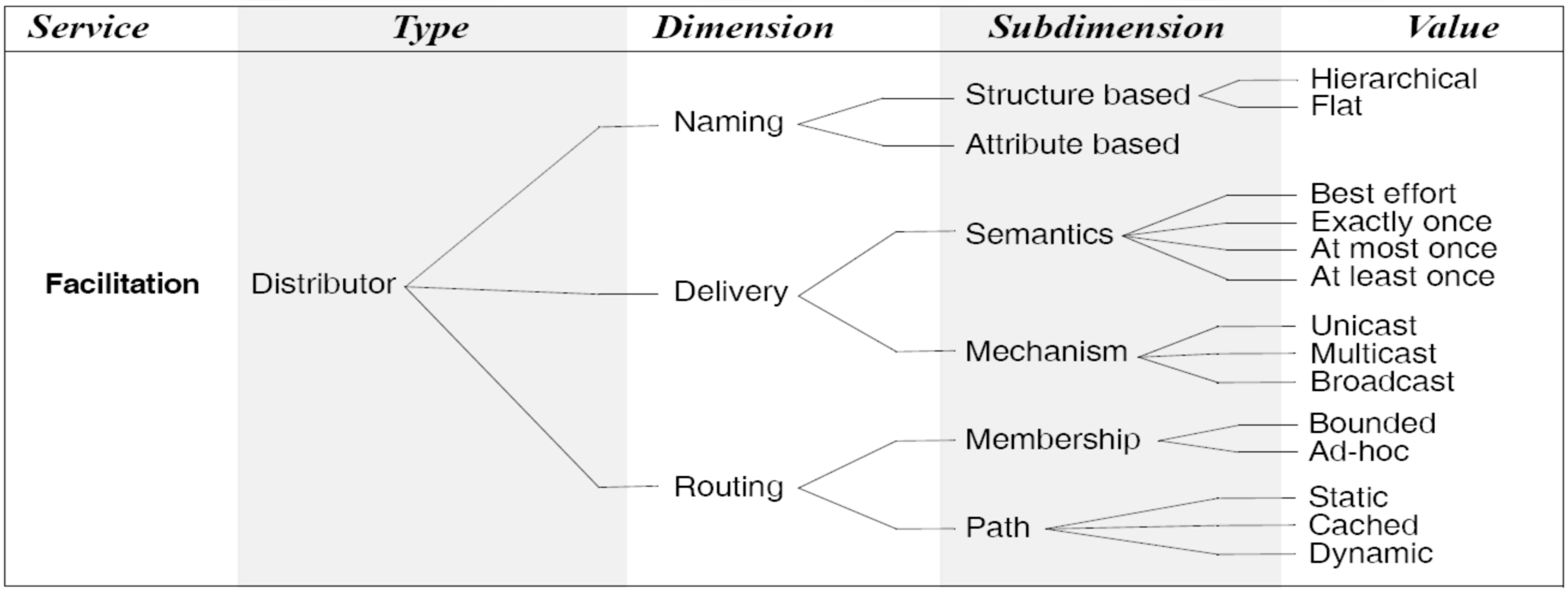
Arbitrator Connectors



Adaptor Connectors



Distributor Connectors



Discussion

- Connectors allow modeling of arbitrarily complex interactions
- Connector flexibility aids system evolution
  - Component addition, removal, replacement, reconnection, migration
- Support for connector interchange is desired
  - Aids system evolution
  - May not affect system functionality

Discussion

- Libraries of OTS connector implementations allow developers to focus on application-specific issues
- Difficulties
  - Rigid connectors
  - Connector "dispersion" in implementations
- Key issue
  - Performance vs. flexibility