

Distributed System Software Architecture Perspective on System Interoperability *

Janis Putman

Presented at:

Ground System Architectures Workshop '98

The Aerospace Corporation

February 25-27, 1998

Organization: D500

*** The views and opinions expressed in this paper are those of the author and do not reflect MITRE's current work position.**

MITRE

Mission

•Information Support to the Warfighter and Decision Makers

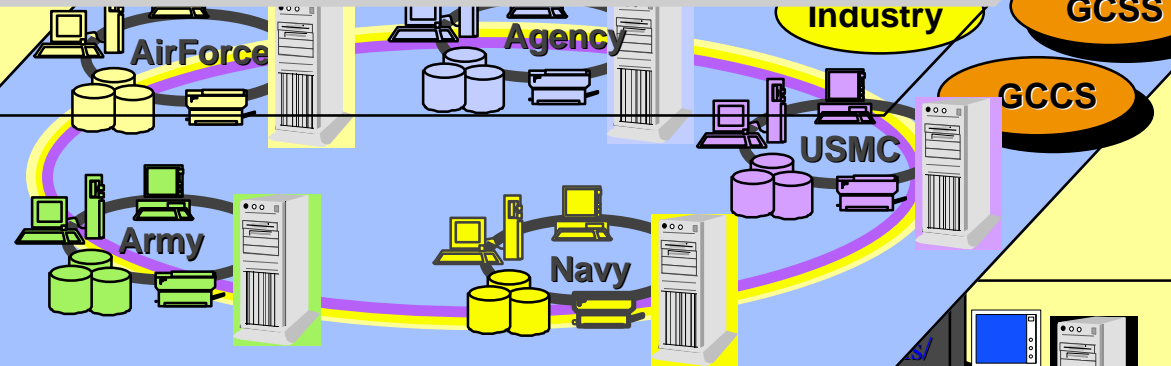


Focused Logistics: Precise Application of Logistics

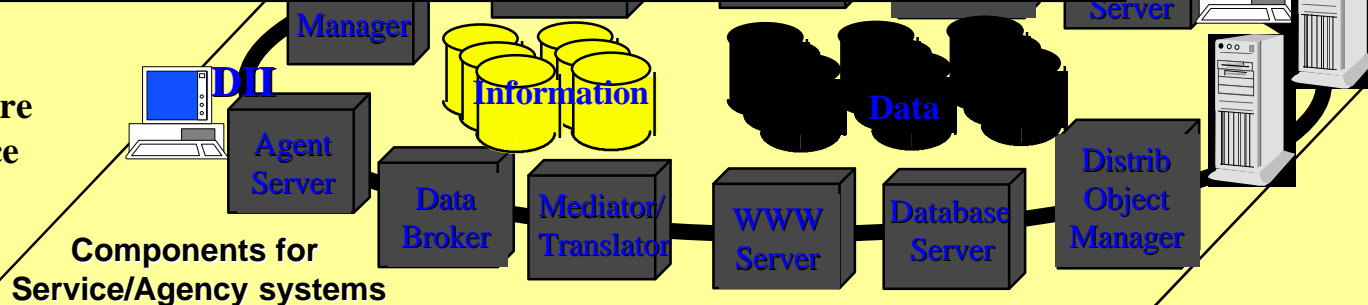
Require
Report
Repair
Replace

•Federation of Component Systems Based on Process Environment

Domains



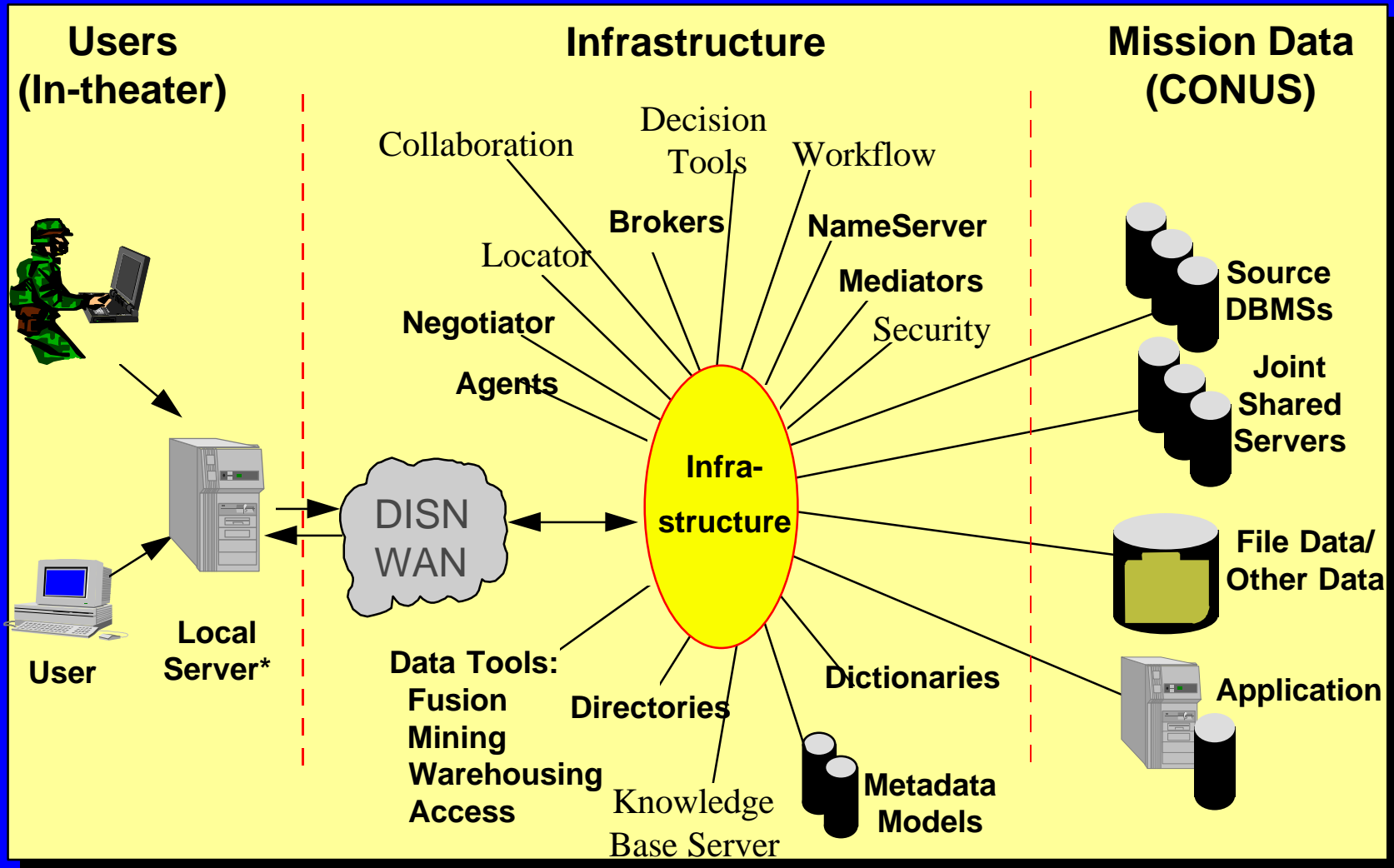
•Prescribed Infrastructure and Interface Solutions



Composable Architectures

Strategic Framework (*)

* Adapted from GCSS Systems Architecture, In Progress



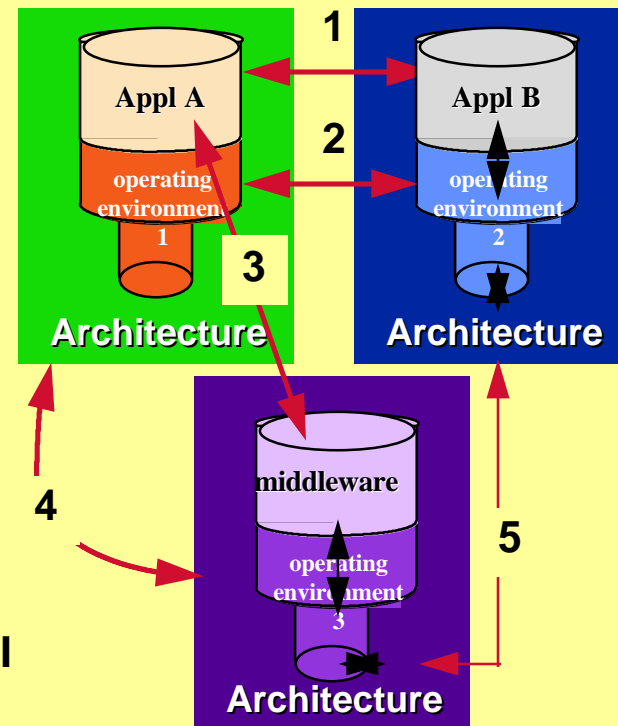
Interoperability Focus

- Interactions among system components need to be addressed :

- 1--application components and connectors,
- 2--application support service components and connectors,
- 3--middleware support service components and connectors,
- 4--application and service architectures, and
- 5--interface mechanisms and behavior/state

- DOD enablers:

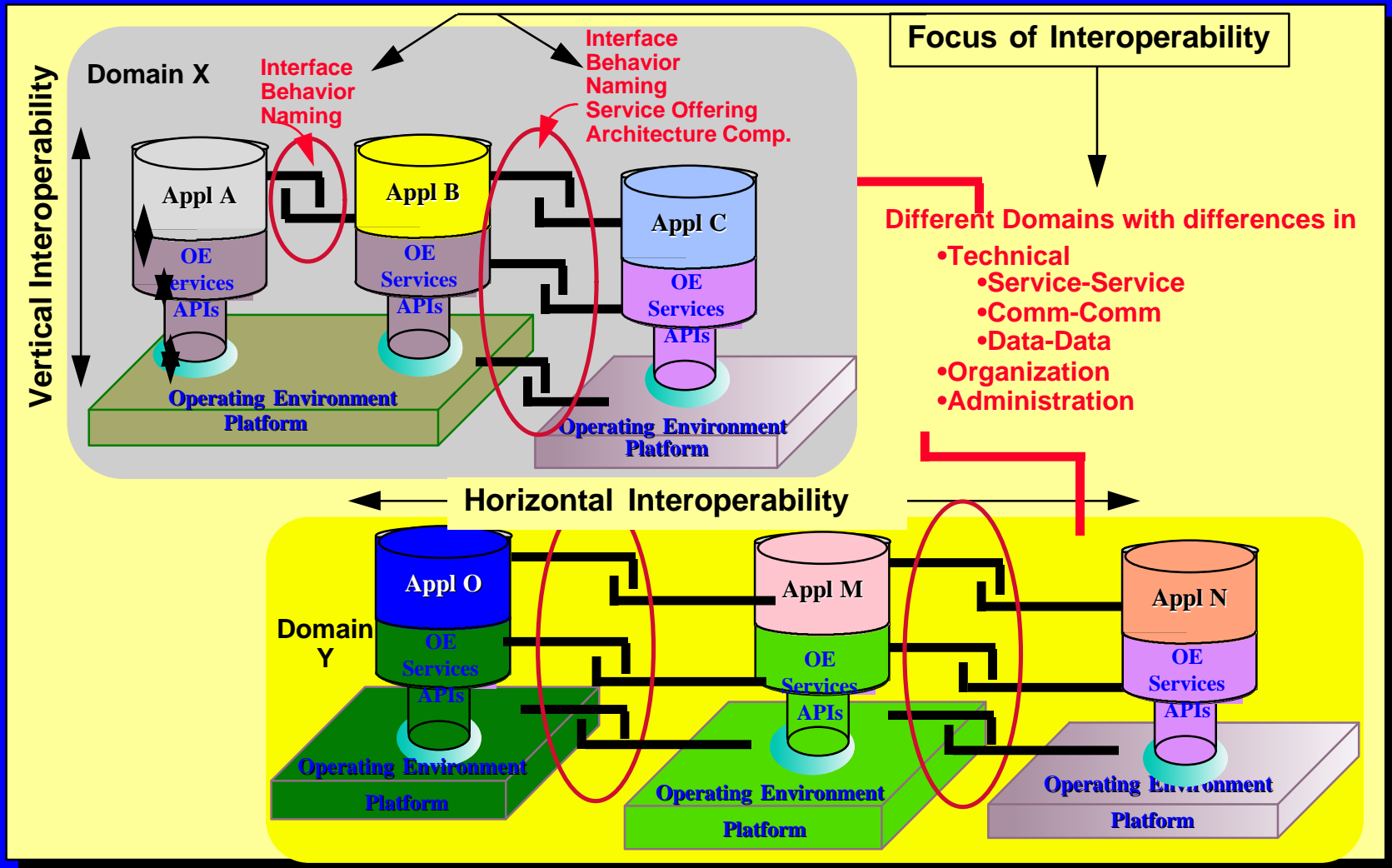
- common operating environments (DII COE)
- standards on the interfaces (JTA)
- common interfaces (APIs to COE)
- common architecture (client/server)



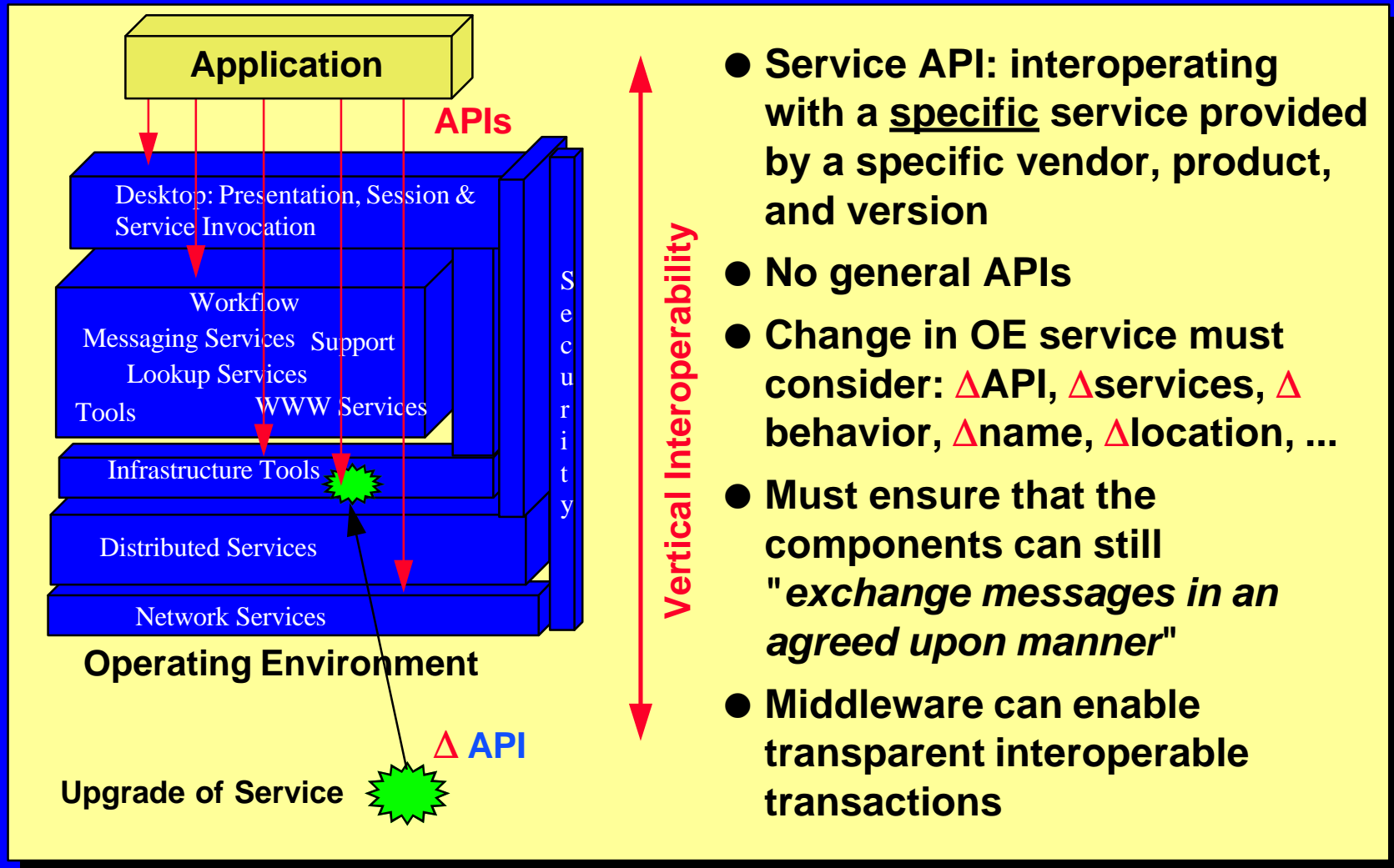
Position

- **Distributed systems composition to realize interoperability and transparency is complex and in need of focused software engineering, and new perspectives of **system interoperability**.**
- **2 concepts to address interoperability across a system of applications and services:**
 - **1. **Interoperable architectures** must be considered: interoperability can only be achieved with composable heterogeneous architectures**
 - **2. Technologies to support interoperability within a domain, and an enhanced set of technologies to support **interoperability across domains** need to be addressed. These technologies may differ.**

Perspectives of System Interactions for Interoperability



Interoperability Perspective: Application To Its Operating Environment (OE)



Interoperability Perspectives Between Client and Server

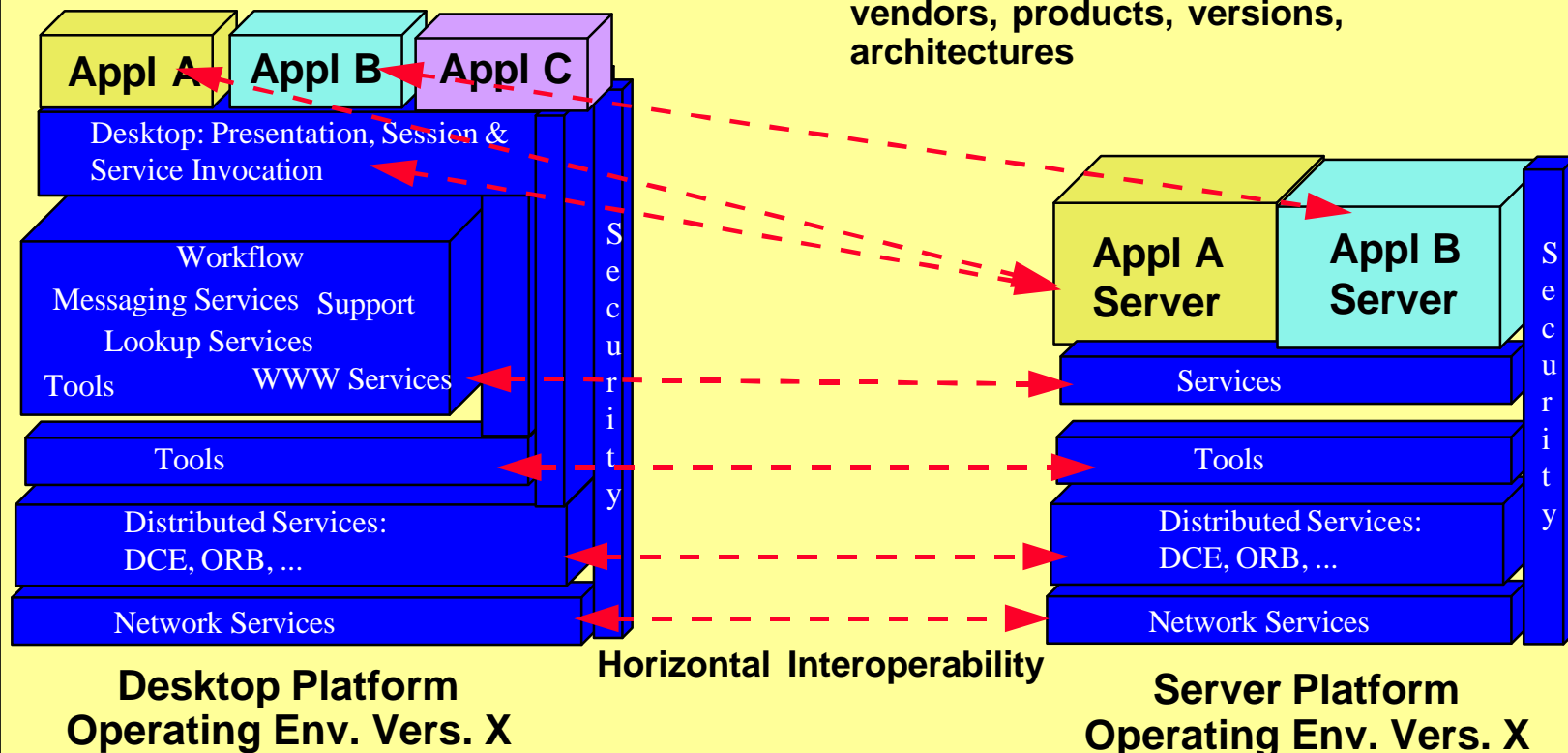
Example Application Use of "Common" Services

Enables

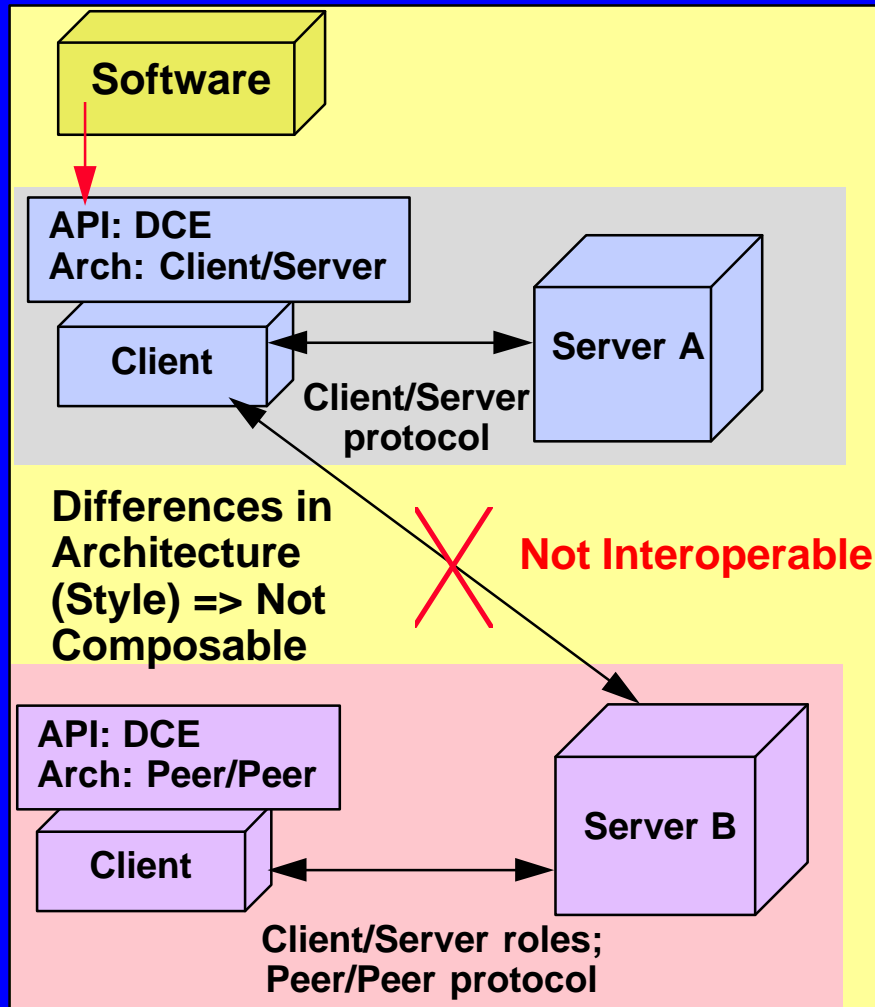
- platform sharing,
- interoperable platform services,
- reusable platform services

Requires

- Composable architectures
- Agreed upon interfaces, states, semantics, behavior, naming, ... across heterogeneous platforms vendors, products, versions, architectures

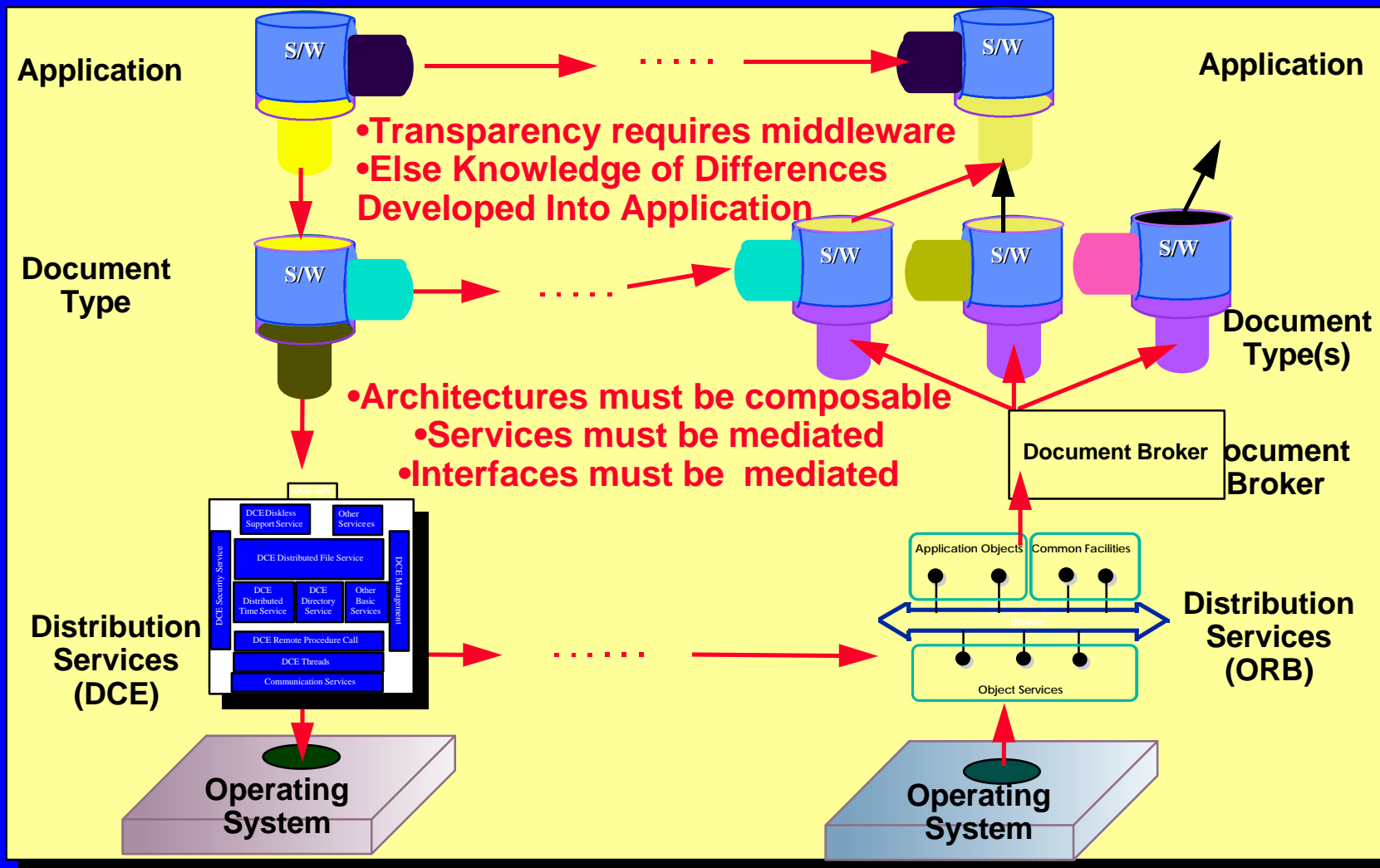


Perspectives of Architecture Composability

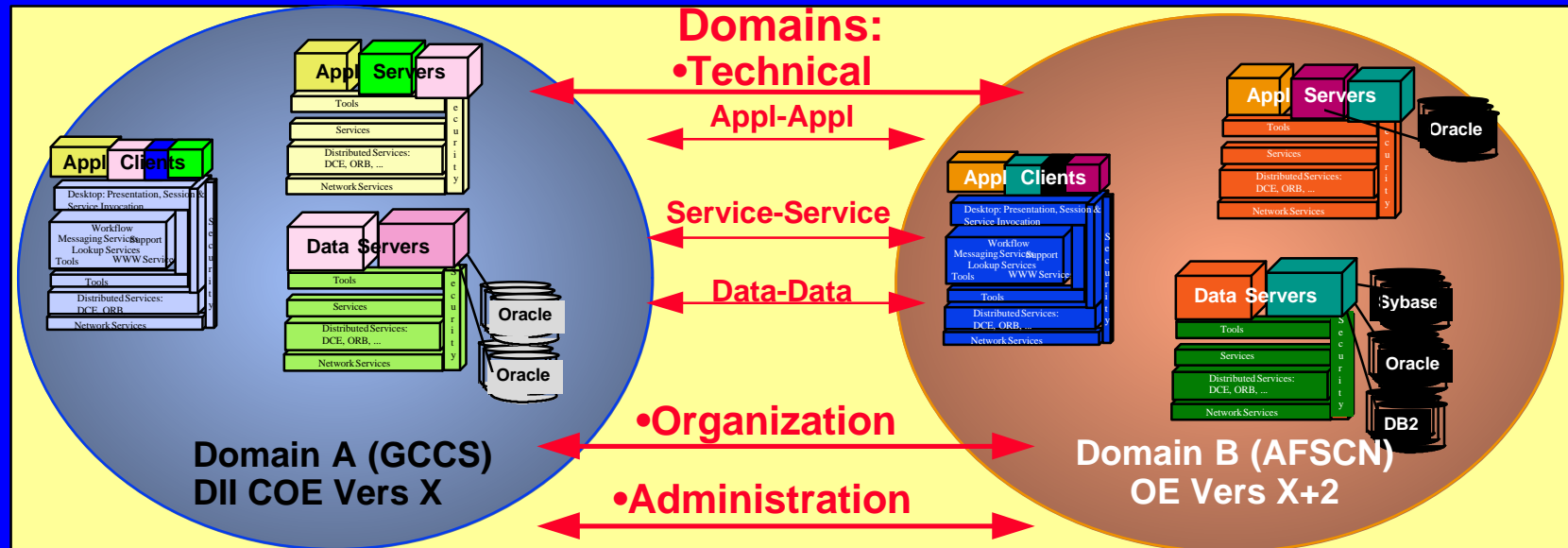


- Difference between “acting in the role of” and “protocol”
- Can meet the API standards
- May not accomplish system interoperability
- Standards must consider both API and Architecture Styles to accomplish interoperability
- Must consider
 - architecture style
 - protocol vs. role
 - service “split” across interface

Interoperability Perspectives: Architecture of Components and Interactions



Interoperability Perspectives Across Domains: Enhanced Technologies Needed



•Different Domains with Differences in

- Technology
- Organization
- Administration
- Different Components & Vers
- Different Applications
- Different Data
- Need to Interact

Addressing Interoperability
Across Domains May Require an
Enhanced Set of Technologies
More Than Those for
Interoperability Within a Domain

Interoperability Perspectives Across Domains: Some Emerging Technologies For Use

- **Federation:** Enables transparent domain-specific software architectures of components and connectors, specifies a “common” means for transparent interoperability
 - what will be shared with other autonomous domains
 - negotiation: binding with distributed components;
 - exporting service offers from separate subsystems;
 - importing service offers from separate subsystems;
 - allowing or forbidding offers from federated components;
 - communicating and negotiating policies;
 - supporting message passing, information sharing, negotiation, and transaction sharing.

Interoperability Perspectives Across Domains: Some Emerging Technologies For Use (Concluded)

● **Negotiator :**

- request for service,
- agreement to perform the service,
- delivery of the result, and
- agreement that the results conform to the request.

● **Mediators:**

- perform translations between schema, data formats, etc.
- reconcile, integrate, and interpret information from multiple, diverse sources

● **Trader (ISO/ITU):**

- links to the interface identifier of the service provider,
- well defined protocol (e.g., import, search, select, add, remove, modify, export, and withdraw)
- provides trading within and across federated domains

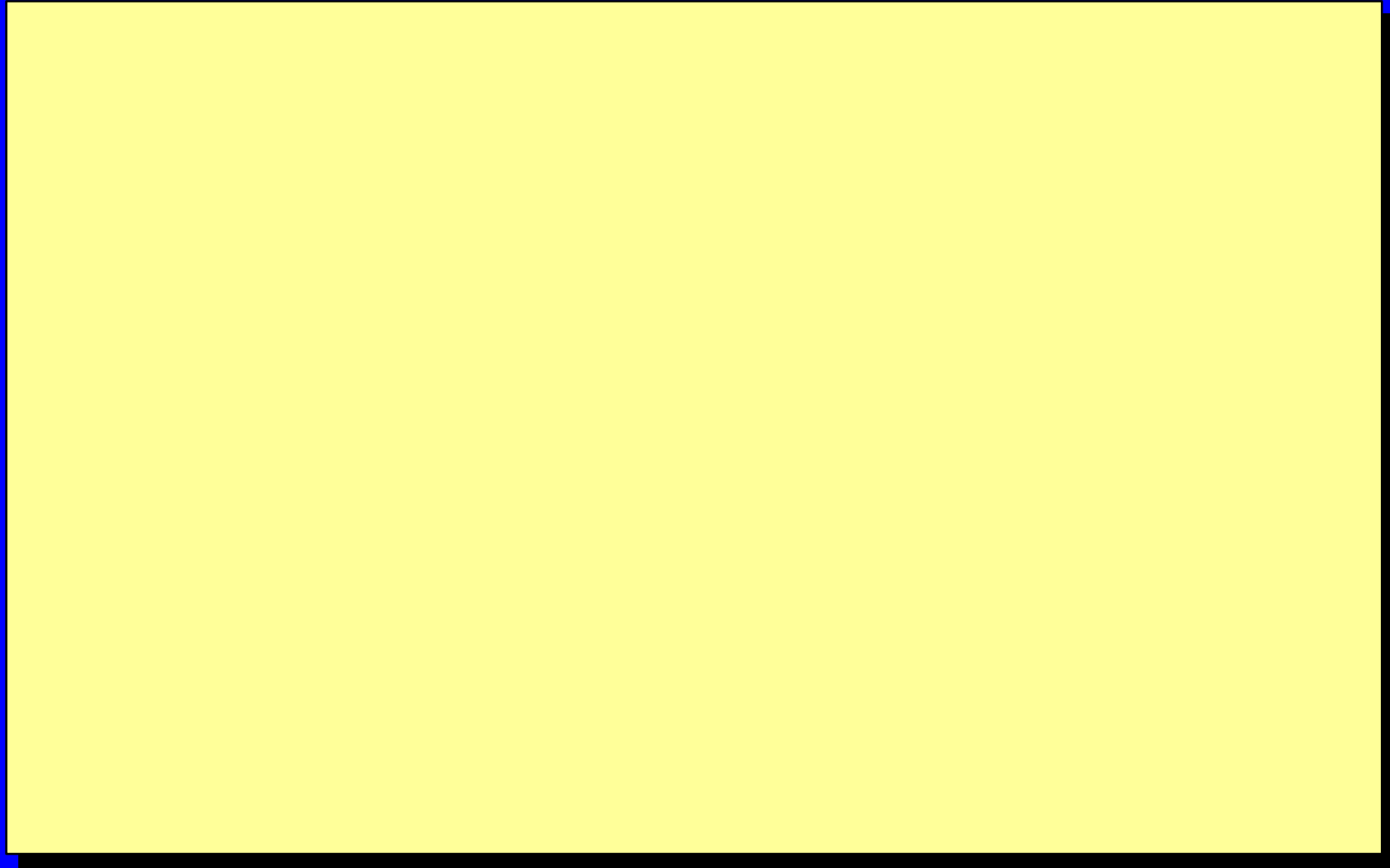
● **Distributed object manager (DOM):**

- provides ability to identify and access available services in a dynamically configurable distributed environment.
- requires trader and other components to support the establishment between two or more interfaces

Recommendations

- **Systems engineers need to focus on interoperability within their domain**
- **Integrate distribution transparency mechanisms with interoperability mechanisms**
- **Interoperability requirements must be highlighted in each software requirements specification (SRS)**
- **A consistent approach towards interoperability across the enterprise strategies should be stated, identifying where each contributes to the solution**
- **Begin to focus on cross-domain interoperability, and consider the emerging technologies as possible enablers**

BACKUP



What Is Interoperability? Transparency?

- ***Interoperability*** is typically defined as “The ability of two or more systems or components to exchange and use information” [IEEE STD 610.12]
 - Two system components are able to exchange information in a mutually agreed upon manner
 - Includes information sharing as well as synchronized operation of the two applications on a common task
 - Interoperability is about agreement
 - Requires good system and software engineering to attain a “higher level” of interoperability
-
- ***Distribution transparency*** is the ability to hide the details of some aspect of distribution from system components
 - Distribution transparencies include: access, location, migration, failure, relocation, persistence, replication, transaction
 - Some are inter-dependent; all require metadata & infrastructure services

DOD Approach

- **DII COE**

- a common set of products that interface with each other through other common set of products, or by use of standards based interfaces across the (common) operating environment set of COTS or GOTS products, and a common “architecture” (client/server)

- **JTA**

- mandates adherence to a set of standards or standards profiles

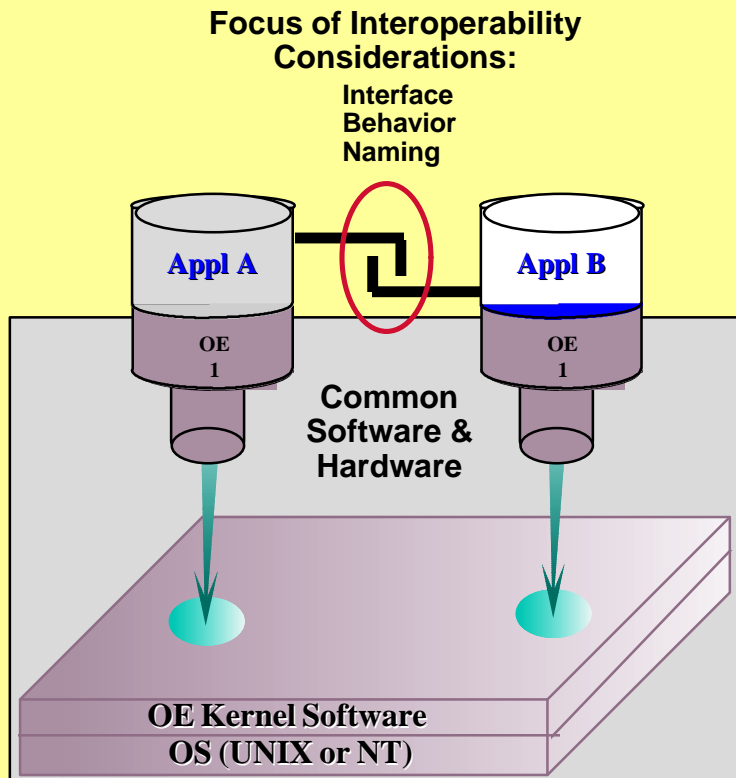
- **TAFIM**

- common, multipurpose, standards-based technical infrastructure

Engineering Perspectives for Interoperability

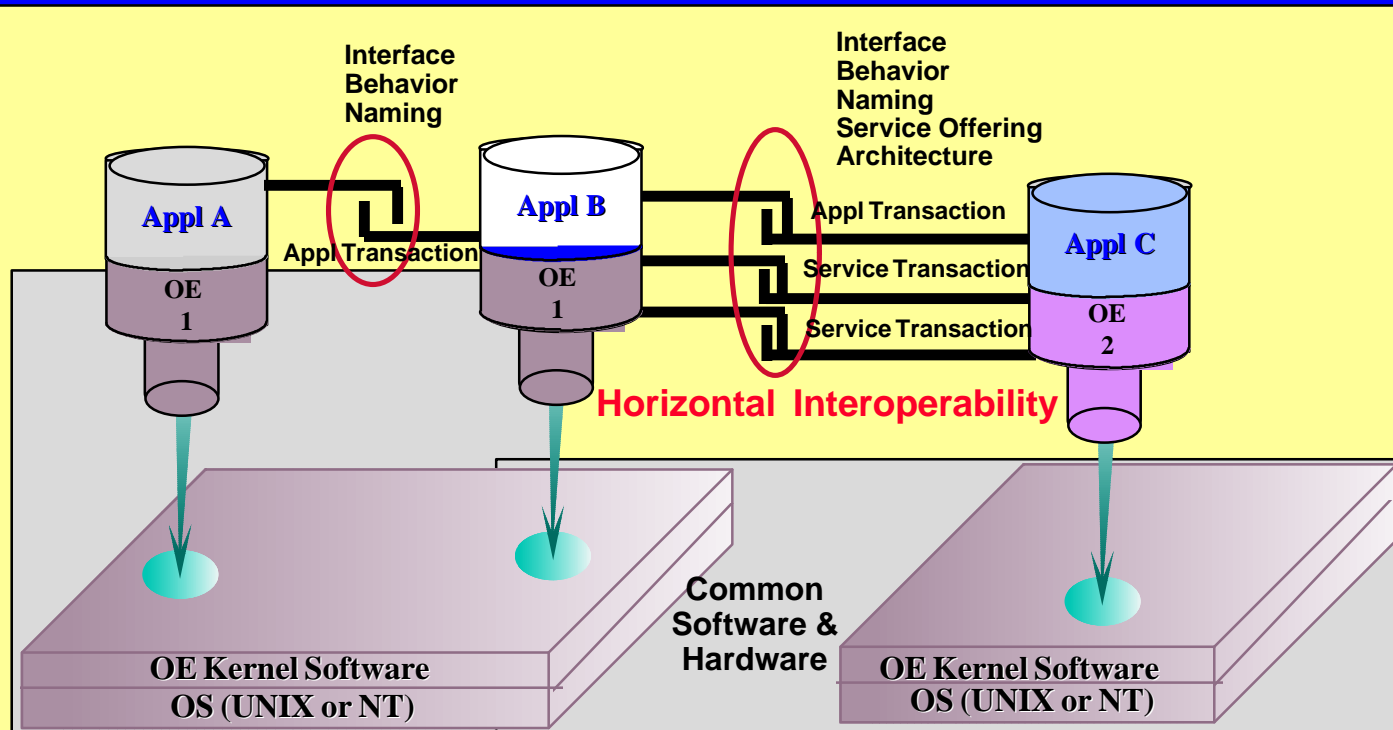
- Interactions between one application and another,
- Interactions between an application and its component based operating environment (OE),
- Interactions between a service within a OE and the service or services of a target (different) OE,
- Agreement on the interfaces, in terms of name, syntax, and semantics (such as side effects),
- Architecture of the components and their interactions involved in interoperability, and
- Distribution transparency, or if the interoperable transactions should be transparent.

Interoperability Perspective: Application To Application With Same OE Components



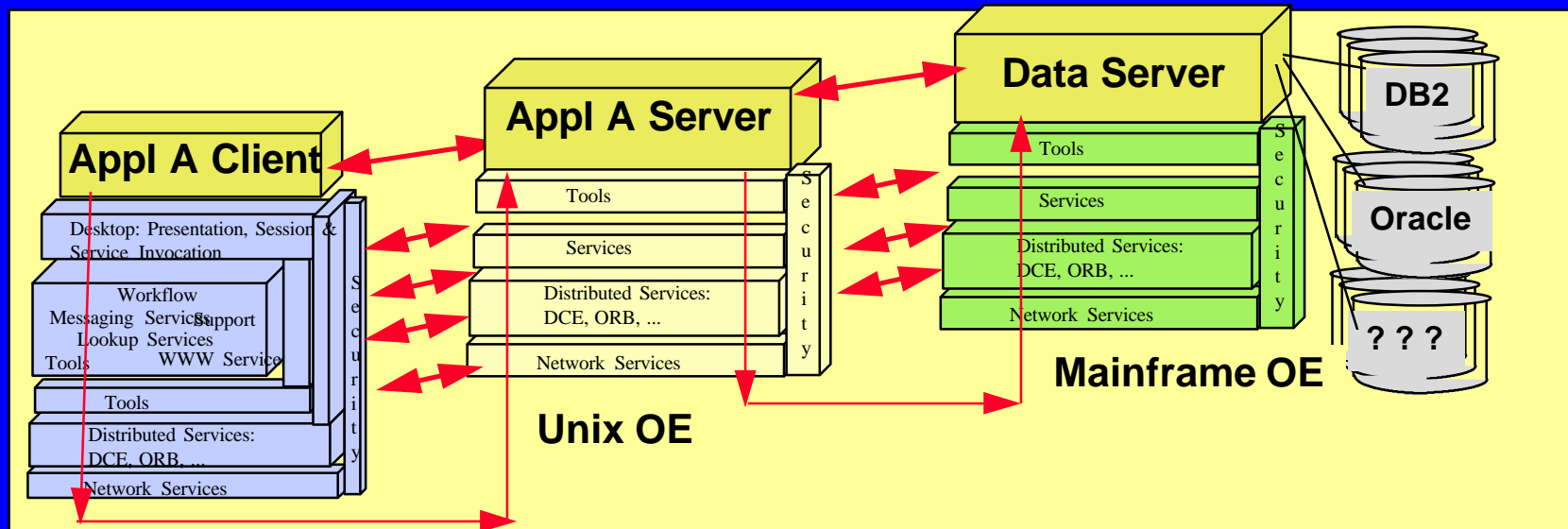
- Same operating environment => No interoperability concerns
- Appl transaction agreed =>
 - Interface between the Appls
 - Behavior of the operations of the interface
 - Format of the data
 - Format of the message being exchanged
 - Naming and location of Appl
- Requires Appl Knowledge:
 - Use of same OE
 - Target application's message interface
 - Target application 's location and availability

Interoperability Perspective: Application With Different OE Components



- Appl-App transaction considerations
- Plus for each of the different OE components and connectors:
interface, behavior, naming,
service offering, and architecture

Interoperability Perspectives For Single Distributed Application



PC/NT OE

- Components Differ Across NT, Unix, Mainframe OEs
- Different Products for Same Component (e.g., Oracle, Sybase)
- Application Engineered To Different OEs
- Engineer Must Consider:

Transparency ?

- Vertical Interoperability of Components on Each Platform
- Horizontal Interoperability Across Platform Components
- Connectors Everywhere
- Presumes Common Architecture: 3-Tier, Client/Server
- Presumes Compatible Architectures

Interoperability Perspective: Agreement On The Interfaces

- Agreed upon standards on the interface
 - same version of standard
 - same mandatory services
 - same optional services, or agreement of how to handle unknown services
 - same syntax of the transfer protocol
 - same semantics of the transfer protocol
 - same means in the protocol to identify the standard in use
- Products compliant with a standard on the interface must also comply with the use of the standard services, or an agreement of how to handle differences.
- Examples: SQL, DCE

Interoperability Perspective: Architecture*

- **Software architecture -- concerned with design at the system level**
 - system structure (or topology)
 - discriminations among different kinds of structures
 - abstractions or generalizations about structures
 - identification, specification, and analysis of properties related to these structures
- **Components of interest are modules and the interconnections among modules**
- **Architectural styles guide the selection of kinds of components and of the strategies for composing them**
 - kinds of components and interconnections can differ substantially between architectural styles
 - properties of interest include system structure, gross performance, component consistency, and other aggregate properties such as security and reliability

* According to Prof. Mary Shaw, CMU

Interoperability Perspective: Across Domains

- **Across multiple domains, interoperability is more complex**
 - **Need to address organizational differences (functional, rules, constraints, mandates, ...)**
 - **Need to address administration differences (security, management, ...)**
 - **Need to address technology differences (version of DII COE, migrating systems, legacy, state-of-the-art, ...)**
 - **Not reasonable to expect the same OE will be utilized across domains, but within a given domain**
- **Differences in computation requiring different capable products: real-time, large scale transactions, massive data transactions, near real-time interactive responses, etc.**
- **Communication differences**
- **Data representations and database schema differences**
- **Functional, performance, evolutionary, and administration differences**

Distribution Transparencies

Each transparency provides a level of independence for the application

- **Access transparency:** A distribution transparency which masks differences in data representation and invocation mechanisms to enable interoperability between components
- **Location transparency:** A distribution transparency which masks the use of information about location in space when identifying interfaces
- **Migration transparency:** A distribution transparency which masks, from an element, the ability of a system to change the location of that element; migration is often used to achieve load balancing and reduce latency
- **Failure transparency:** A distribution transparency which masks, from an element, its failure and possible recovery, to ensure fault tolerance
- **Relocation transparency:** A distribution transparency which masks relocation of an interface from other interfaces bound to it
- **Replication transparency:** A distribution transparency which masks the use of a group of mutually behaviorally compatible components to support an interface; replication is often used to enhance performance and availability
- **Persistence transparency:** A distribution transparency which masks, from an element, variations in the ability of a system to provide processing, storage and communication functions to that element
- **Transaction transparency:** A distribution transparency which masks coordination of activities among a configuration of components, to achieve consistency

Glossary

- **Architecture:** where the design issues involve overall association of system capability with components
- **Architectural interoperability:** deals with inter-operation between two systems with possibly different styles
Composable: two elements (in this case, architectures) can be combined or coalesced to work together, most likely through the use of middleware
- **Distribution Transparency:** ability to hide the details of some aspect of distribution from system components
- **Horizontal interoperability:** deals with inter-operation between peer components
- **Interoperability:** ability of two or more systems or components to exchange and use information
- **Mediate:** reconcile or arbitrate differences between two elements, typically through translation services
- **Sameness:** means the same standards on the interfaces, software, hardware, and architecture
- **System:** Something of interest both as a whole or as comprised of parts. A component of a system may itself be a system, in which case it may be called a subsystem. [RM-ODP]
- **System:** People, machines, and methods organized to accomplish a set of specific functions. [FIPS PUB 11-3]
- **Vertical interoperability:** deals with inter-operation between a service requester and a service provider;
- **Operating Environment:** a profile of components that are manifested as services/products