

Distribution Transparencies For Integrated Systems*

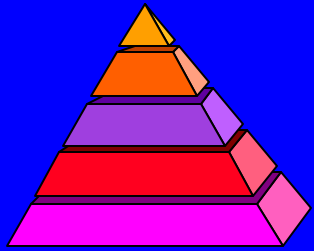
Janis Putman, The MITRE Corporation

Ground System Architectures Workshop 2000
The Aerospace Corporation
February 2000

Organization: D500

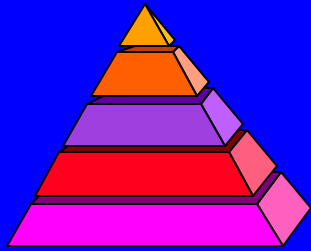
* The views and opinions expressed in this paper are those of the author and do not necessarily reflect MITRE's current work position.

MITRE

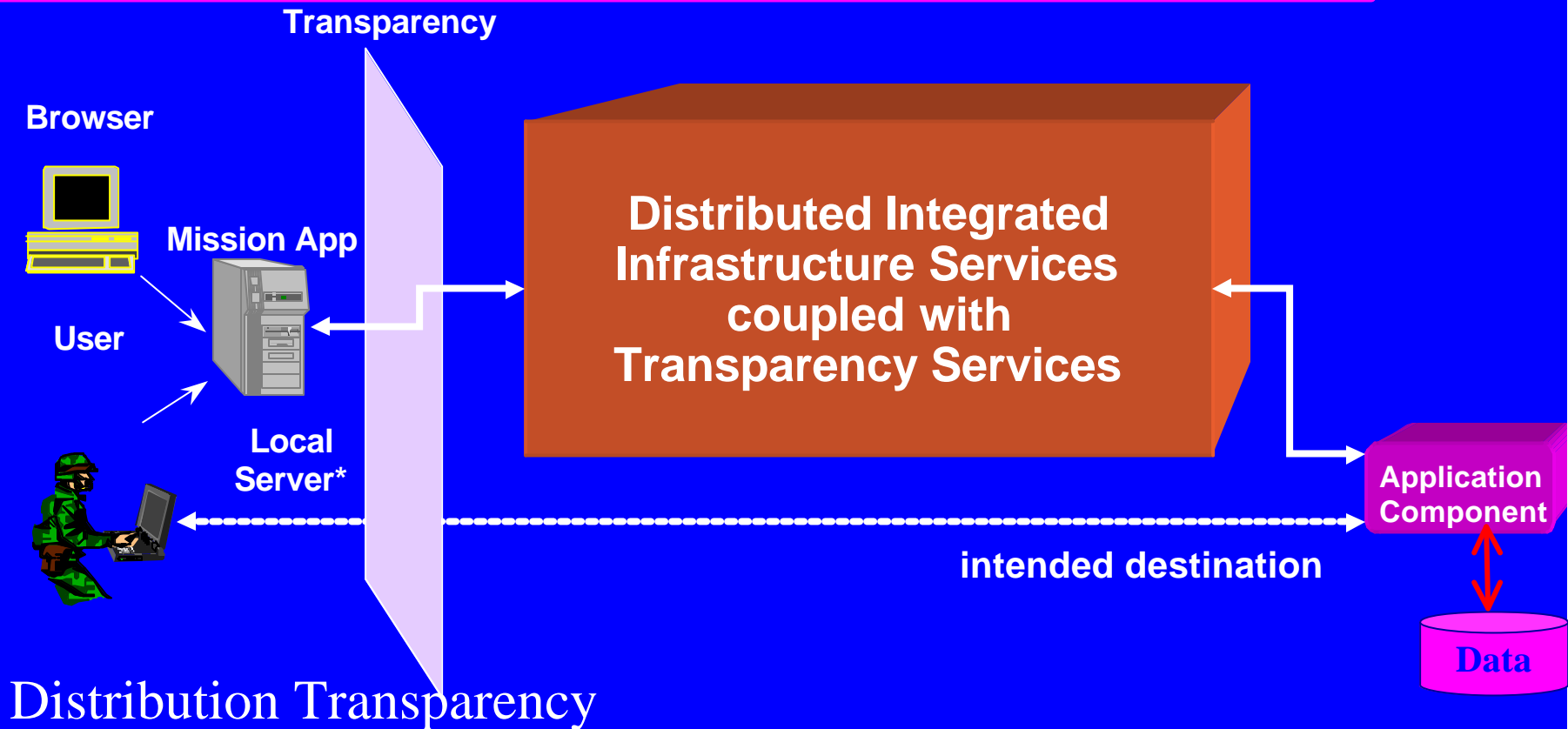


Outline

- ◆ **Overview**
- ◆ **Transparency Mechanisms**
 - Failure Transparency
 - Fault Tolerant CORBA 3.0
 - RM-ODP Transparency Mechanisms
 - Access and Location Transparency
 - Security and Location Transparency
- ◆ **Semantic Behavior**
- ◆ **Summary**



Distribution Transparency Perspective*

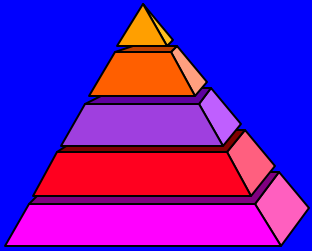


The property of hiding from a user the potential behavior of some parts of a distributed system

NOTE - Users include end-users, application developers, service implementers, etc.

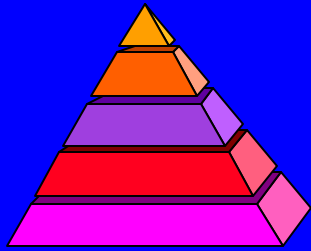
*adapted from [JP]

MITRE



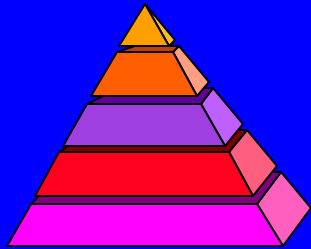
What Transparency Provides

- ◆ Frees an end user or an application developer from having to provide some service, and the details of that service
- ◆ "Hides" many of the cumbersome and required details of where resources and services are and how to utilize them in a large distributed system
- ◆ Allows all kinds of changes (static, dynamic, what have you) to be made, without affecting the application
- ◆ These changes can be made to the runtime environment, the topology of the architecture, even the hardware
- ◆ Provides strategic separation of concerns in the architecting process: application and (transparent) service



Properties of Distribution Transparency

- ◆ **Distribution transparency can be broken down into a number of individual transparency issues, e.g., access, security, failure**
 - **Selectable: the architect selects those that are appropriate**
 - **Some are inter-dependent; all constrained by transparent-specific schema**
 - **All require specific extra infrastructure services**
- ◆ **Transparency enhances achieving system properties:**
 - **reliability (continuity of service),**
 - **availability (how often the service is ready for use),**
 - **fault-tolerance (recoverability from failure),**
 - **enhanced performance (load-balancing),**
 - **decreased latency (replication), and**
 - **others**

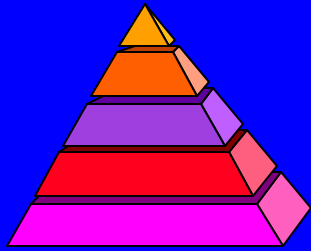


Distribution Transparencies

Each transparency provides a level of independence for the application

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

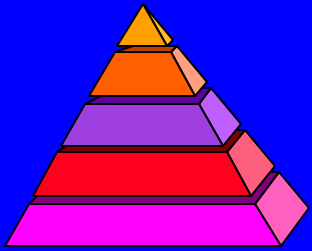
*Emerging transparency in RM-ODP



Example: Enabling Reliability Through Failure Transparency

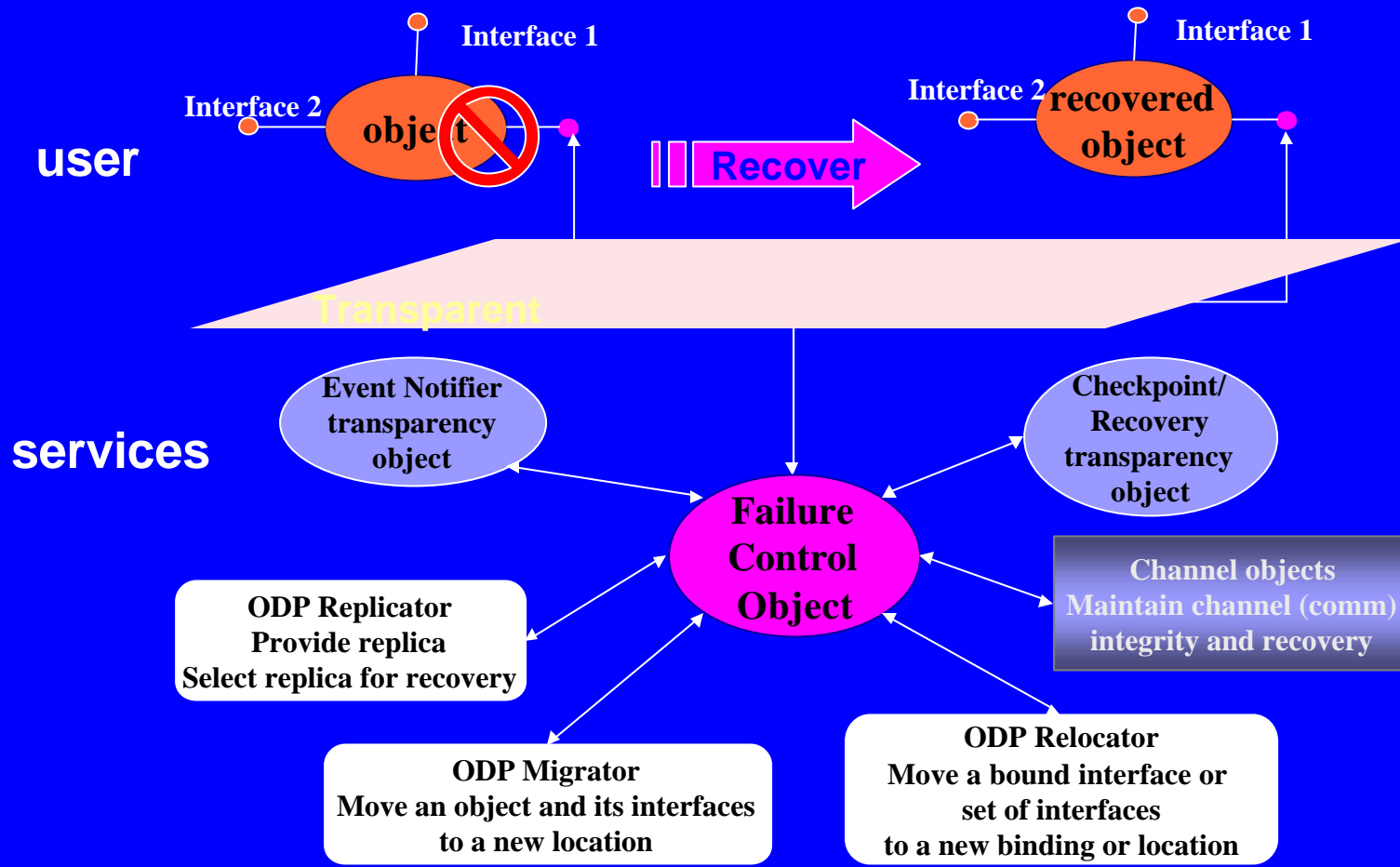
Failure Transparency: localizing to the infrastructure explicit capabilities to enable management of the system under failure:

- ◆ **detecting failure**
- ◆ **replicating components of the system for more assured availability,**
- ◆ **ensuring the integrity of a binding across a channel, and**
- ◆ **controlled management of movement of a software component, recovery of an object, relocating an interface, restoring a bound interface**



Transparent Reliability Mechanisms*

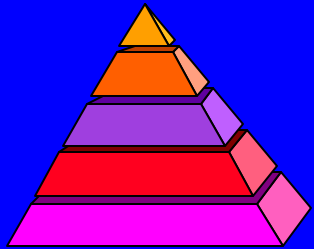
Based on RM-ODP Concepts



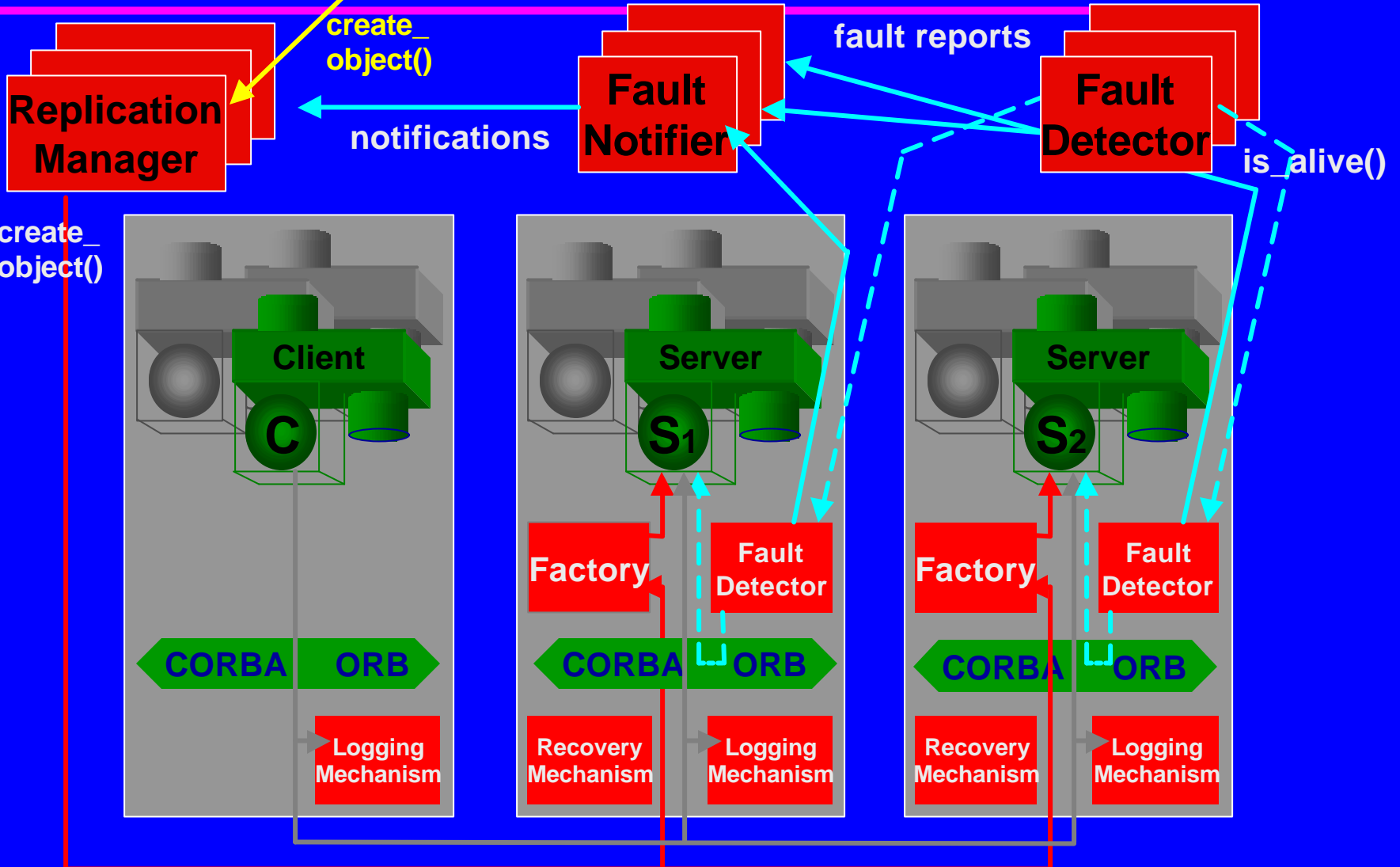
*[JP]

Fault Tolerant CORBA: Architectural Overview

Achieving Dependability

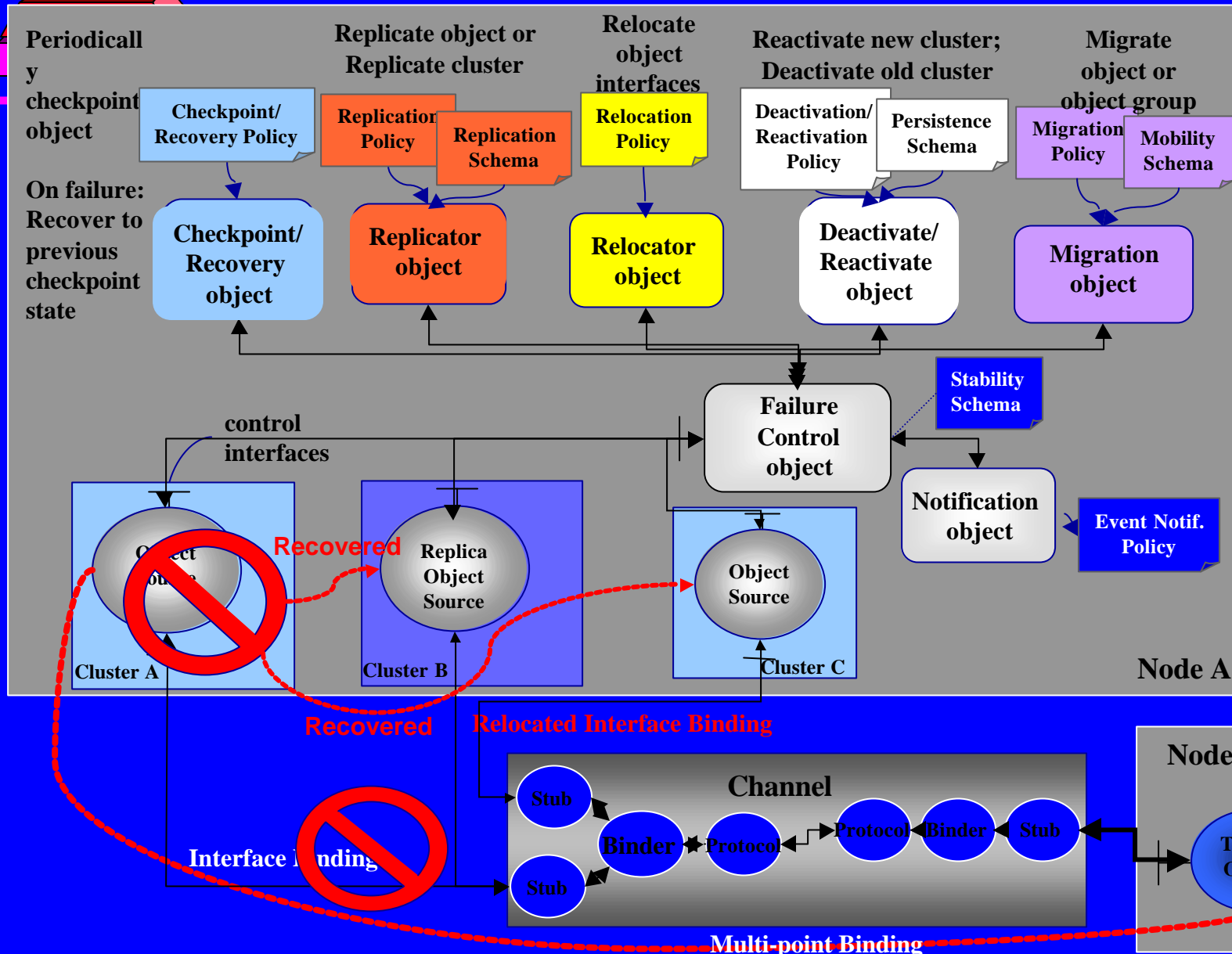


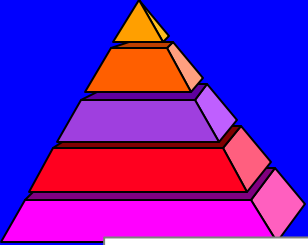
Application IDL



RM-ODP Fault Tolerant Framework*

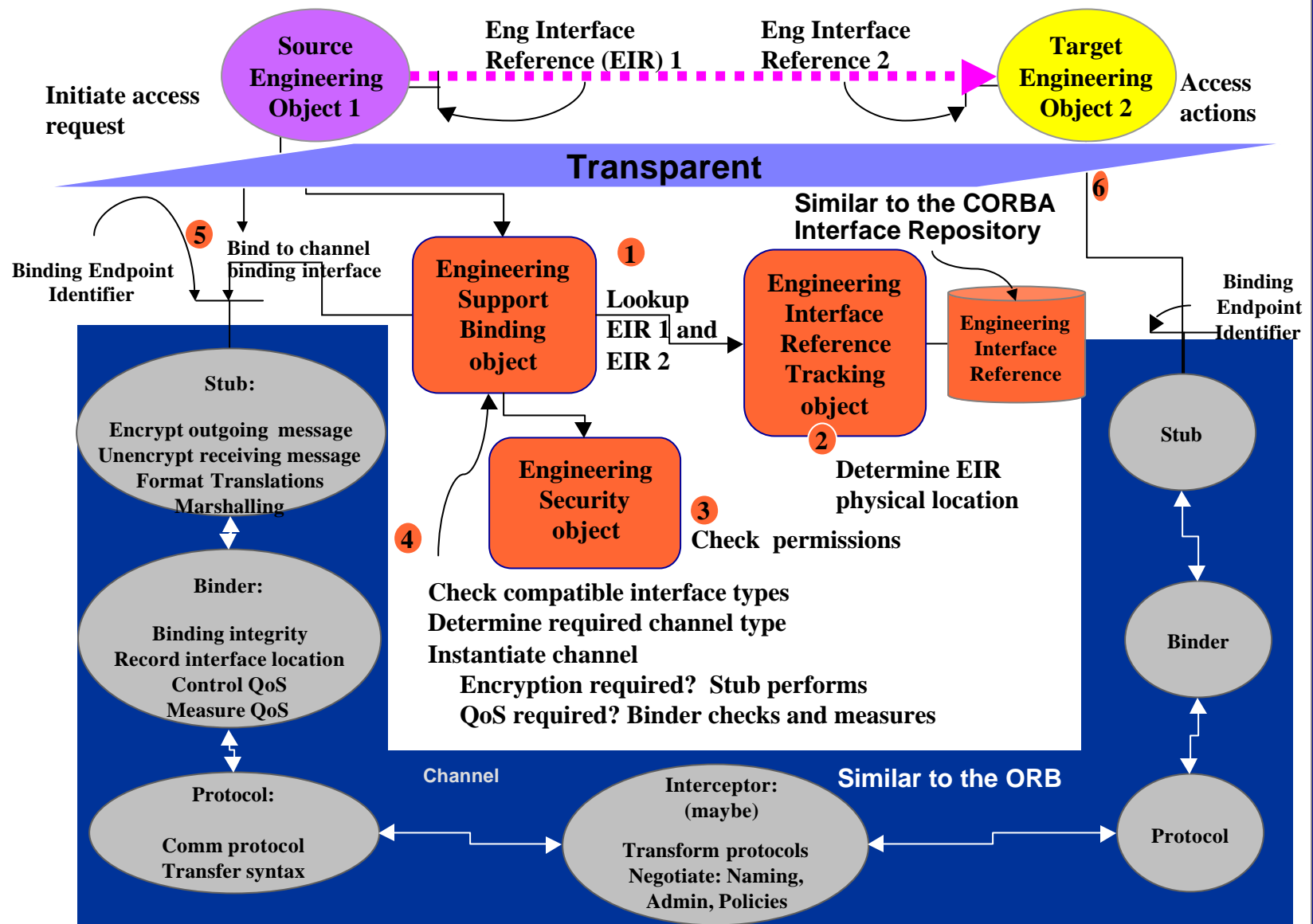
Achieving Reliability: Dependability, Availability, Integrity of Bindings

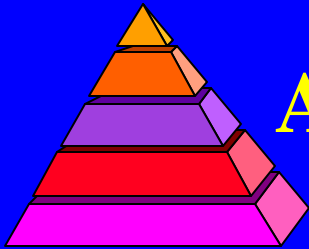




Access and Location Transparency Framework*

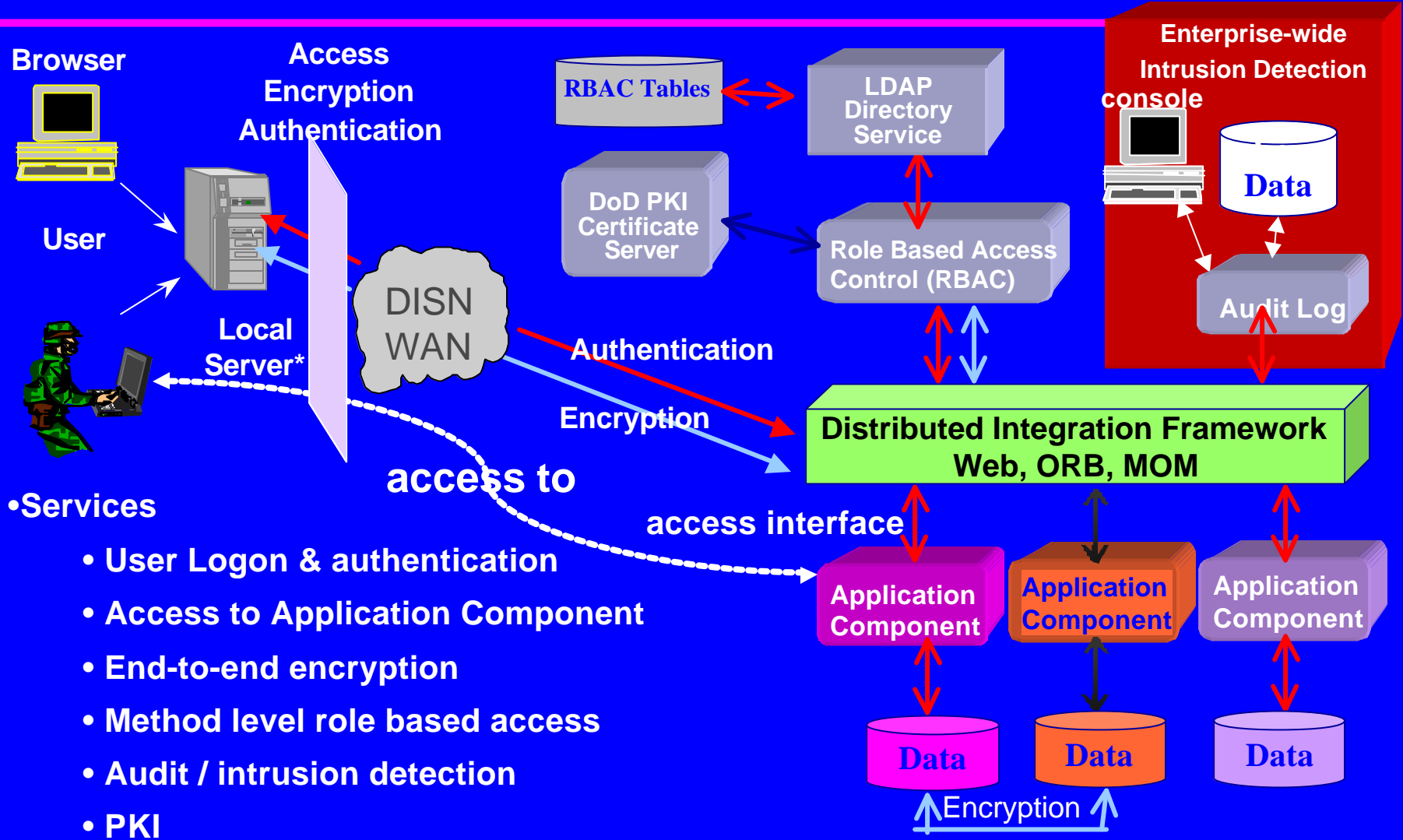
Based on RM-ODP Concepts





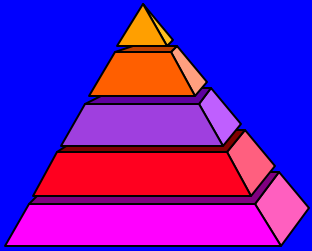
Access and Security Transparencies

Source: "GCSS-AF Security Architecture Approach", AFITC, Sep 99; with eSecurity briefing to GCSS-AF



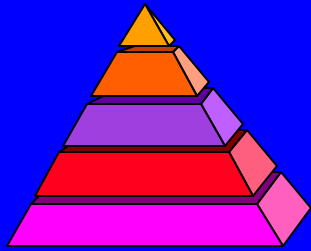
• Services

- User Logon & authentication
- Access to Application Component
- End-to-end encryption
- Method level role based access
- Audit / intrusion detection
- PKI
- Single Sign-On



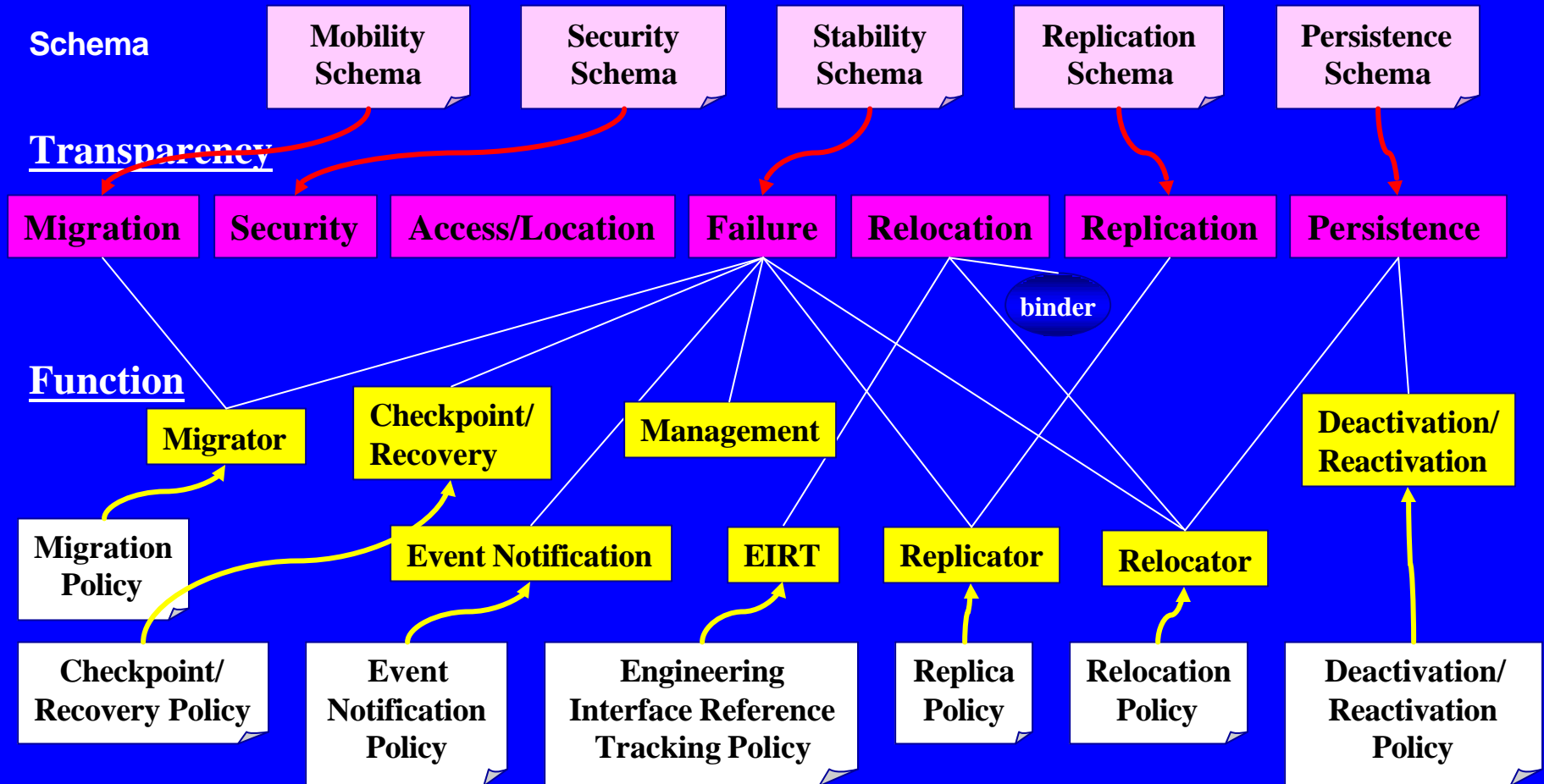
What about Semantic Behavior?

- ◆ How the infrastructure behaves is specified in a schema associated with the transparency and one or more policies associated with the infrastructure service or function
- ◆ Each interaction is associated with a contract, establishing the expected behavior across the interaction, and hence defined by one or more associated policies.
- ◆ Each policy identifies the role and information associated with the policy in terms of invariants, pre-conditions, post-conditions, and constraints
- ◆ A policy is is defined by a set of obligations, permissions, and prohibitions, assumed by a role in achieving a particular purpose.
 - Obligation prescribes what behavior is required, fulfilled by some action that achieve the behavior.
 - Permission prescribes what behavior may be allowed to occur.
 - Prohibition prescribes what behavior must not occur.



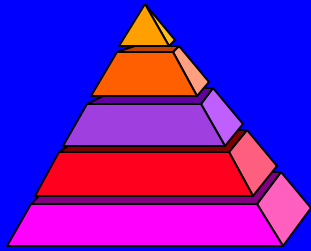
Relationship Among Transparencies, Functions, Schema, and Policies*

Based on RM-ODP Concepts



Policies

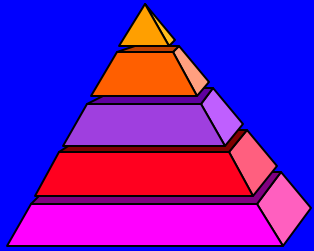
*[JP]



Critical Barriers to Transparency Architecture

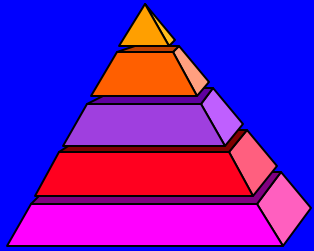
- ◆ **Inconsistency among views of transparency**
 - Primarily related to technical understanding of infrastructure services required
 - Transparency must be an up-front system requirement
 - Mismatch will occur if not architected in from beginning

- ◆ **Conflicting goals**
 - Technological issues: “too hard” or “no COTS solutions” or aversion of technologies
 - Business case : no clear ROI ... empirical data only
 - Organizational issues of architecting common infrastructure services for system of systems



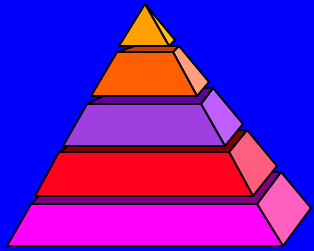
Open Is Critical to Transparency

- ◆ Meaning of each transparency needed, to define expected behavior as specified in a transparency specific schema
- ◆ Infrastructure mechanisms need well defined open interactions: semantics and interfacing
- ◆ Mechanisms need to be constrained by policy specifications
- ◆ Common architectural and design pattern(s) needed for architectural choices
- ◆ UML representation of open transparency solutions still in research
- ◆ RM-ODP provides specification of open concepts and mechanisms to achieve each identified transparency



Summary

- ◆ Integrated systems architects need to focus in on the aspects of distribution for a well-functioning system
- ◆ Distribution transparency is a capability that supports many of the system property requirements (e.g., the “ilities” of security, reliability, migratability, availability, fault-tolerance, integrity, enhanced performance, decreased latency, and others)
- ◆ Distribution transparency off-loads Mission Application development needs to system infrastructure services
- ◆ Transparency generally requires additional infrastructure services, along with precisely defined behavior of how those services support the needs of distribution transparency
- ◆ Distribution transparency solutions emerging in new specifications (e.g., CORBA 3.0 for failure, replication, persistence, location, security)
- ◆ COTS vendors building more transparency into products (e.g., security)
- ◆ Open distribution transparency concepts and mechanisms have been specified in RM-ODP; in research otherwise



References

- ◆ [RM-ODP] ITU-T Recommendation X.900 series | ISO/IEC 10746 series: Information technology - Open distributed processing", 1995.
- ◆ www.esecurityinc.com
- ◆ [FT-Corba] "Fault Tolerant CORBA" briefing, 16 Aug 99, orbos/99-08-27.
- ◆ [FT Corba 2] "Joint Revised Submission Fault Tolerant CORBA", OMG orbos/99-10-05, October 25, 1999.
- ◆ [JP] Janis Putman, "Open Distributed Software Architecture Using RM-ODP", ISBN 0-13-019116-7 1911F-3, copyright 2001 Prentice Hall, to be published.
- ◆ Janis Putman, "Model for Fault Tolerance and Policy from RM-ODP Expressed in UML/OCL" to appear in 3rd IEEE International Symposium on Object-oriented Real-time Distributed Computing (ISORC 2000) proceedings.
- ◆ Janis Putman, "General Framework for Fault Tolerance from ISO/ITU Reference Model for Open Distributed Processing (RM-ODP)", to appear in Fifth International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 99) proceedings.