Open Distributed Processing Reference Model

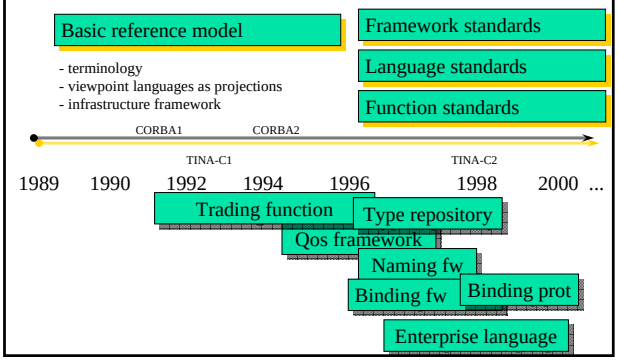
Outline

- Introduction
- Viewpoints
- System model: objects and infrastructure
- Services and functions, transparencies
- Usage
- Where to read more
- Trading function

Introduction -- RM-ODP Goals

- ODP mission in terms of research and industry needs
  - World-wide information services
  - Exploitation of new technology
  - Interoperation and portability
- ODP as a standardization effort
  - Additional concepts, consistent terminology
  - Meta-architecture, framework
  - Function standards

Contributions and Schedules



Introduction -- Audience and status

- ODP is aimed for architecture designers and tool developers
- basic reference model is stable (intentions and words), components under development (tools), marketing missing
- object model differs from OMA, TINA, and any object design methodology or object language; closer to component models

Who needs the standard?

- Needed by system specifiers
- Needed for communication between system specifiers
- Needed for communication between stakeholders and implementors
- Needed for a stable business functionality description
  - independent of technology and technology change
- Needed for mission critical business systems

The Reference Model of ODP

A framework for ODP standardization and system specification covering all aspects of distributed systems ("enterprise" context, functionality, infrastructure, technology) and comprising

- a comprehensive and coherent set of object-oriented modelling concepts
- a structure for specifications in terms of viewpoints on a system
- a language (concepts and rules) for expressing each viewpoint specification

Applying ODP viewpoints to modelling of the B2B middleware

Enterprise vp: What? Why? To whom?

Information vp:
On what information?
Where and how that information is created?

Engineering vp:
By which tools?
How distributed?

Computational vp:
What operations?
Who provides?

Technology vp:
How is the engineering support implemented?

Overview on the viewpoints

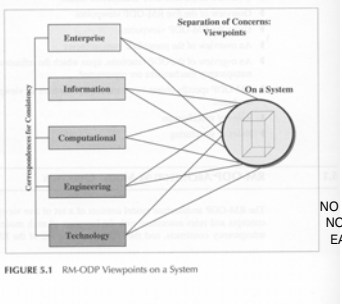
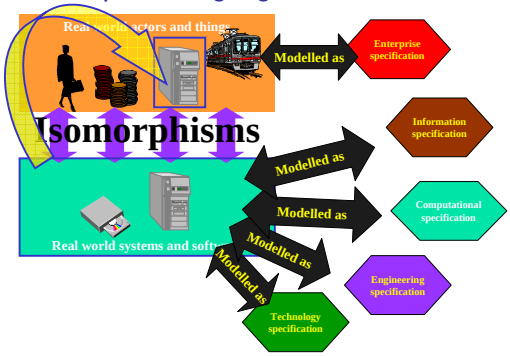


FIGURE 5.1 RM-ODP Viewpoints on a System

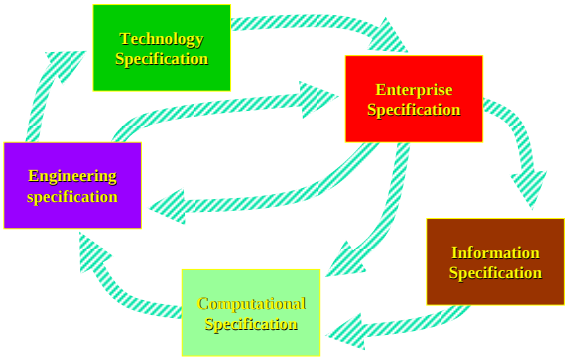
Enterprise viewpoint:
WHY?
Information viewpoint:
WHAT INFO?
Computational viewpoint:
WHAT LOGIC?
Engineering viewpoint:
HOW?
Technology viewpoint:
STANDARDS?

NO LAYERING, NO PROCESS PRESCRIBED
NO LANGUAGE BINDINGS PRESCRIBED
EACH VP SPEC. MAY HAVE DIFFERENT OBJECTS

Viewpoints and the Real World – the Enterprise Language is different!



An ODP based specification process



System model: Fundamental concepts

- Objects and interfaces
- Dynamic binding of interfaces
  - Late binding
  - Distribution transparency
  - QoS management
- Organisational concepts
  - domain, federation
- Meta-information management functions

Objects and interfaces

- specification vs. implementation objects
- introduction and instantiation
- separate hierarchies for types and templates
- separation of views to the shared interface: causality, heterogeneity
- multiple interfaces
- operation and stream interfaces

Modeling Concepts

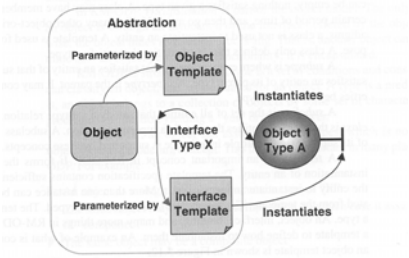
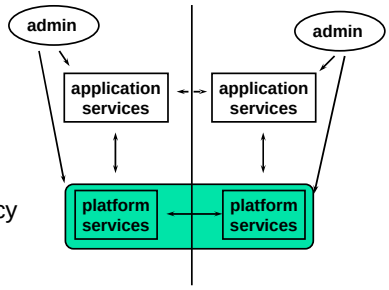


FIGURE 5.13 Object, Type, Interface, and Template Concepts

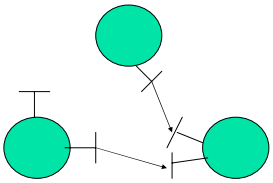
Organizational concepts

- object
- system
- X-community
- Y-domain
- X-federation
- Y-transparency



(Computational) object model

- interactions
  - interrogations
  - announcements
  - streams
- interface binding
  - interface type
  - client/server view
  - interface reference
  - properties of the binding



Another overview of viewpoints

- Enterprise – purpose and policy
  - A strategic view
- Information – shared information
  - A unifying view of information resources
- Computational – functional application design
  - An application designer view
- Engineering – structure of middleware
  - A view of communication and resource control
- Technology – known standards

The enterprise specification

- Specifies the roles played by an system in its organisational environment
- An object model of a social/commercial organisation in terms of:
  - *enterprise objects*
  - *communities* (of *enterprise objects*)
    - *objectives*
    - *behaviour*
      - *roles*
      - *processes*
    - *policy*

The information specification

- Specifies system behaviour abstracted from implementation
- An object model of the *system* describing the semantics of information and of information processing in the *system* in terms of:
  - *information objects*
  - *invariant schema* - predicates on *information objects* that must always be true
  - *static schema* - state of *information objects* at some *location in time*
  - *dynamic schema* - allowable state changes of *information objects*

11.9.2007 ODP-RM -- Lea Kutvonen 19

The computational specification

- Specifies computational structure in terms of units of distribution and portability and their interactions abstracted from the detail of how distribution is accomplished
- An object model of the ***system*** describing the structure of processing in terms of:
  - *computational objects*
  - *interfaces*: operations supported
  - *invocations*: operations invoked
  - *activities*: sequences of invocations
  - *computational bindings*
- Provides funticionts for partner discovery
  - Trading: Mediates information about services
  - Type repository function
  - Naming framework with federation

11.9.2007 ODP-RM -- Lea Kutvonen 20

The engineering specification

- Specifies the mechanisms and services that provide the distribution transparencies and QoS constraints required by the system
- An object model of the system describing the infrastructure supporting the computational structure
  - *basic engineering objects*
  - *(infrastructure) engineering objects*
  - *clusters, capsules, nodes*
  - *channels*
  - *functions*

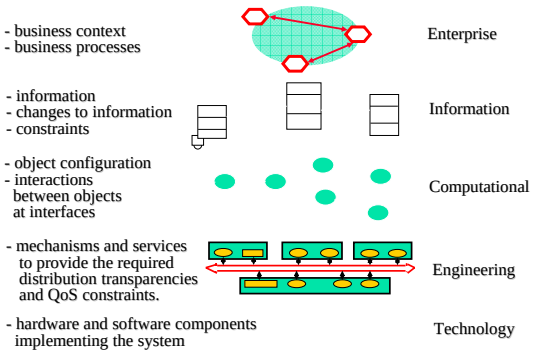
11.9.2007 ODP-RM -- Lea Kutvonen 21

The technology specification

- Specifies the procurable pieces from which the system is built.
- An object model of the system
  - defining the configuration of *technology objects* and the *interfaces* between them that comprise the ODP system
  - identifying *conformance points*

11.9.2007 ODP-RM -- Lea Kutvonen 22

An ODP system specification



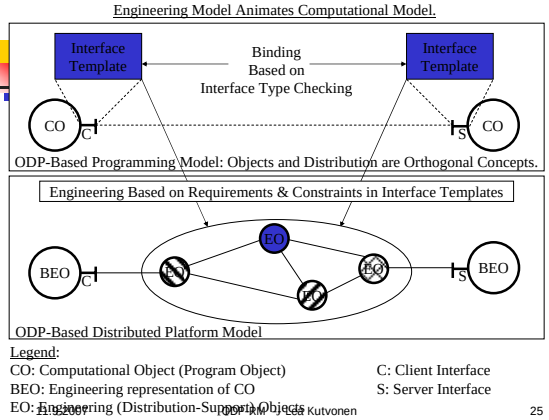
Essentials of computational and engineering viewpoints

- Computational viewpoint concepts of
  - explicit binding
  - distribution transparency
  - trading function for object discovery
- Engineering viewpoint concepts for processing and providing explicit binding realisation

11.9.2007 ODP-RM -- Lea Kutvonen 24

Engineering Mechanization of Computational Model

Engineering Model Animates Computational Model.

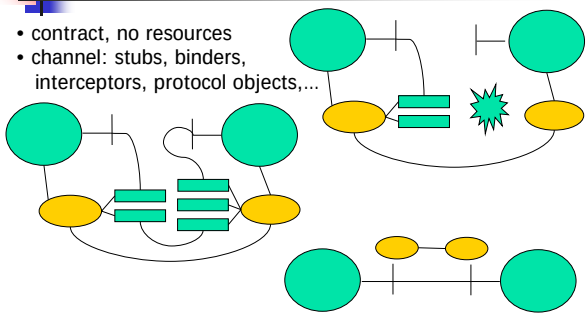


Legend:

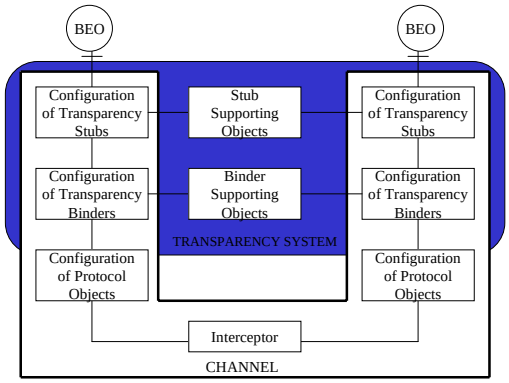
CO: Computational Object (Program Object)
BEO: Engineering representation of CO
EO: Engineering (Distribution-Support) Objects
C: Client Interface
S: Server Interface

Engineering model of communication

- contract, no resources
- channel: stubs, binders, interceptors, protocol objects,...



Simplified Generic Channel Structure



Stub objects involve application semantics and add further interactions and/or information to the interactions between basic engineering objects to support distribution transparency such as *access transparency*, *transaction transparency*.

Binder objects control and maintain the binding between distributed *basic engineering objects* (in case they are relocated, reactivated or replicated at a different *node*) contributing to the provision of distribution transparencies such as *location transparency*, *migration transparency*, *replication transparency*, *failure transparency*, *resource transparency*, etc.

Protocol objects interact with other protocol objects in the channel to provide communication between *basic engineering objects*.

Realisation of ODP binding

ODP	CORBA
▪ trading of services	▪ trading of object instances
▪ type conformance ensured by negotiation during late binding process	▪ type conformance ensured by inheritance
▪ heterogeneity	▪ polymorphism
▪ dynamic type repository	▪ static interface repository

Distribution transparency

ODP	CORBA
▪ federation transparency	▪ access transparency
▪ selective transparencies	▪ location transparency
▪ access	
▪ location	
▪ relocation	
▪ migration	
▪ failure	
▪ persistence	
▪ replication	
▪ transaction	

DISTRIBUTION TRANSPARENCY

Distribution Transparency: Hiding distribution-related details from the application programmer/designer.

Selective Transparency: ODP application designer/programmer can select the level of transparency needed in the design and have full control over other aspects of distribution by turning off some transparencies.

ODP Distribution Transparencies

Transparency	Central Issue	Result of Transparency
Access	The method of access to objects (invocation mechanism and data representation)	Clients need not be aware of *access* mechanisms at the server interface (see CORBA).
Concurrency	Concurrent access to objects in the distributed system.	Clients are masked from the effects of concurrent access to the server interface.

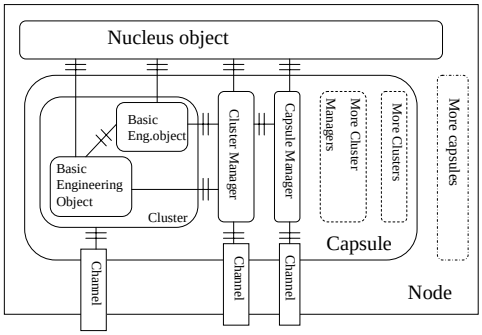
ODP Distribution Transparencies (Cont'd)

Transparency	Central Issue	Result of Transparency
Location	Location of object in the distributed system	Clients are unaware of the physical location of the server.
Migration	Dynamic re-location of objects during the "bind-session".	Clients are unaware of the dynamic migration of the server.
Replication	Multiple invocations on replicated objects, multiple responses, and consistency of replicated data.	Client invokes a replicated server group as if it were a single server. Distribution of requests, colation of responses, consistency of data, and membership changes are hidden.
Resource	Resource management policies of the *node* (deactivation and reactivation of objects).	Client unaware of the deactivation and reactivation of the server.
Failure	Partial failure of object in the *node*.	Client unaware of the failure of the server and its subsequent reactivation (possibly at another node).

ODP Distribution Transparencies (Cont'd)

Transparency	Central Issue	Result of Transparency
Transaction	Coordination required to satisfy transactional (ACID) properties of operations	Clients unaware of coordination activities among a configuration of objects required for ACIDity.
Federation	Pan-organizational boundaries.	Clients unaware of interactions crossing administrative and technology boundaries.

Engineering model for control



Engineering Object	System representation
Node	Single computer system, network of workstations managed by a distributed operating system, any autonomous information processing system with independent *nucleus* resources and failure characteristics.
Nucleus	Processing, storage, and communication resources of a *node*.
Capsule	The concept of address space in operating systems.
Cluster	The concept of 'linked' modules to form an executable program image.
BEO	The program module which may not be executed in isolation.
Channel	The run time 'binding' between distributed BEOs
Transparency object	Special purpose modules which enhance the operating system environment of the *node* and can be dynamically linked into the distributed application program

11.9.2007 ODP-RM -- Lea Kutvonen 35

Engineering viewpoint functions

- Node/Object/Cluster/Capsule management
- Coordination functions:
  - event notification, checkpointing and recovery, de/reactivation
  - group, replication, migration, tracking
  - transaction
- Security functions
  - Access control, audit, authentication, integrity, confidentiality, nonrepudiation, key management

11.9.2007 ODP-RM -- Lea Kutvonen 36

Relationship between Computational and Engineering Model

	Computational Model	Engineering Model
1	*Service-oriented view*	*System-oriented view*
2	*Focus on applications.* The focus is on the *functionality* of the distributed application.	*Focus on mechanisms for application support:* The focus is on the subsequent manifestation of the application on the distributed platform.
3	Computational model provides *distribution-transparent interaction* semantics for application components.	Engineering model provides mechanisms for the *realization* (support) of distribution transparent interactions identified in the computational model.
4	*Application-designer's view:* Computational model hides distribution details from the application programmers and designers.	*System-designer's view:* Engineering model reveals the mechanisms which regulate and enable distribution between application components.
5	*Programming support environment:* Computational model provides a language-independent distributed programming environment capable of building and executing distributed applications. It constitutes an abstract (programming) machine, whose realization is the purpose of an engineering model.	*Distributed execution support environment*: The engineering model, that describes the structure and organization of distribution support services, constitutes a *virtual machine* model for executing distributed programs conforming to the computational model. It provides a machine-independent execution environment.

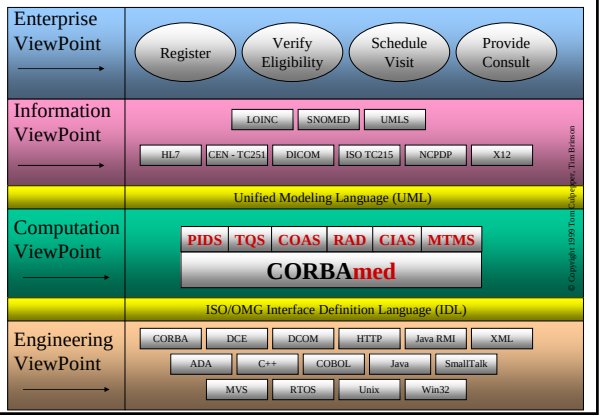
Alternative notation languages

A standard of UML for ODP system specifications - X.906 | ISO/IEC 19793 covering:

- definition of a set of UML profiles for expressing a system specification in terms of ODP viewpoint specifications
- relationships between the resultant ODP viewpoint specifications
- relationships between a system specification using ODP viewpoint specifications and the OMG Model Driven Architecture
- FCD May 2005; FDIS Dec 2005; IS May 2006

11.9.2007 ODP-RM -- Lea Kutvonen 38

Healthcare Standards Roadmap



Survey accomplished by Mitre Corporation, 1999

Peer Review:

- Formal Standards Bodies (that created RM-ODP)
  - International Standards Organization (ISO)
  - ITU-T/CCITT - Formal Telecomm Standards
- Object Management Group (OMG)
  - Aligned with OMA and RFP
  - Required of Vertical Domains and Object Services
- Mission Critical Systems Industries
  - Telecommunications Industry : Used by TINA Telecommunications Consortium and Telecomm companies: ATT, Lucent, Nortel, GTE, ...
  - Financial Industry: Merrill Lynch, Morgan Stanley, ...
  - Manufacturing: Product Data
  - Health Industry: US GCPR, VHA, Umich, GEHR, NLM VHP, I2, CORBAmed, HL7, X12, etc.
  - Governments: UK Logistics, UK C3I Methods for C3I Interoperability, Norway C2, Australia C2, US C4I, P4I, etc.

11.9.2007 ODP-RM -- Lea Kutvonen 40

Usage

- Viewpoints can be used in software engineering process; part of MDA
- functions like trading developed in liaison with OMG -> CORBA includes
- framework standards direct middleware standards in future

11.9.2007 ODP-RM -- Lea Kutvonen 41

ODP system specifications and the OMG Model Driven Architecture®

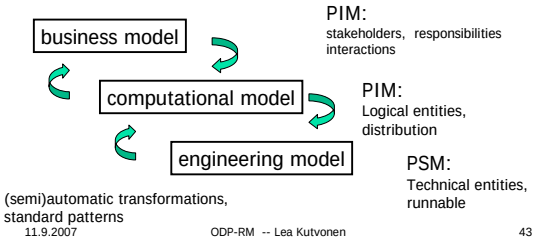
A system specification that is compliant with the RM-ODP also satisfies the requirements of the MDA®. Specifically:

- the enterprise specification is a computation independent model (CIM)
- the information, computational and engineering specifications together form a platform independent model (PIM), where clause 8 of the RM-ODP Part 3 defines a virtual machine which is the context for platform independence
- the technology specification is a platform specific model (PSM)
- the correspondences between the viewpoint specifications express the transformations by means of which one model is derived from another.

11.9.2007 ODP-RM -- Lea Kutvonen 42

Usage

- MDA process (Model driven architecture) by OMG

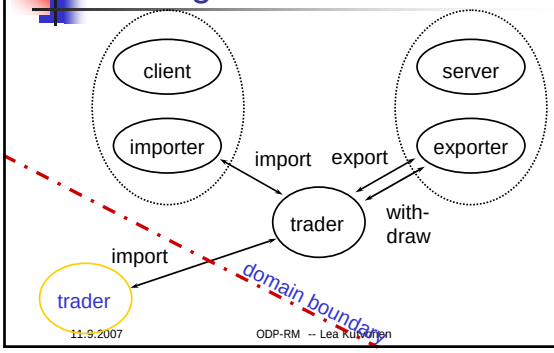


Where to read more?

- Some tutorials:
  - IS10746-1 (not quite consistent w. Part 2&3)
  - Linington-95, Raymond-95, Blair/Stefani-97
  - Putman: Architecting with RM-ODP
  - CORBA, TINA, ANSA, etc. **DO NOT** share the set of concepts nor goals
- ODP-group in Finland
  - Lea.Kutvonen@cs.Helsinki.FI

Open Distributed Processing Reference Model – Trading service

Trading service



Meklauspalvelu

- tarjoukset
  - ominaisuusarvoja (nimi, arvo)
  - ominaisuudet staattisia tai dynaamisia
  - proxy offer
- kyselyt
  - valintaehdot (aritmetiikka, vertailut, joukko-oppi)
  - järjestysehdot (min, max, with, first, random)
  - etsintärajoitteet (alueet, etsintäsyvyys, aika, hinta)
  - palautusohjeet (attribuutit, tarjousten lkm)

Meklauspalvelu

- valinta
  - pyytäjän antamat valintaehdot
  - meklareiden sisäiset valintasäännöt
  - valintaprosessin rajoitteet
  - meklariverkoston tila

Meklauspalvelun merkitys

- Late binding -tuki
  - paikallistetaan palvelun tuottava objekti vasta kutsuaikana
  - antaa mahdollisuuden vaihtaa joustavasti palvelun tuottavaa objektia
- Siirtyminen komponenttiperustaisiin järjestelmiin

Tyyppienhallintapalvelu (Type repository function)

- järjestää globaalisti saataville paikallisesti määriteltyjä tyyppejä
- tyyppimäärittely: nimi, toiminnallisuus, attribuutit
- esittää tyyppien välisiä suhteita, mm. alityyppi
- järjestää tyypit ODP-käsitteiden mukaisiin luokkiin
  - interface, service, operation, parameter, data, ...
  - behaviour, stream, type, template, ...