An ODP-Based Type Manager for Trading Services

GSIT POSTECH

YongHoi Joo

Contents

Introduction

- **Distributed Computing Systems**
  - span heterogeneous platform
  - require a variety of distributed services
  - distribution transparency
  - reliability and availability will be critical
- **Open Distributed Processing**
- **Trading Function**

Introduction

- **Key Issue**
  - limitations of current ODP trader
  - basis for common understanding of function and semantic of services
- **Type Management**
- **Goal**
  - present the type manager for trading services
  - using ODP viewpoint

Overview

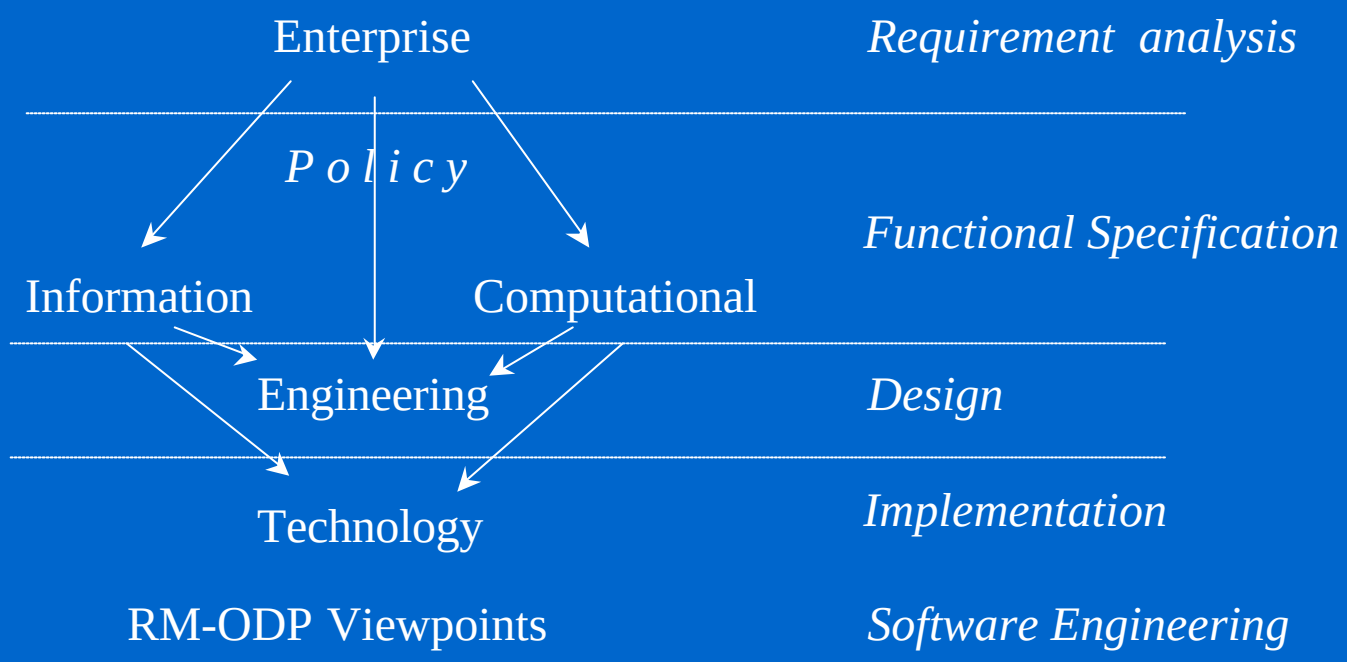
- **ODP**
  - distribution transparency sharing services over heterogeneous platforms.
- **RM-ODP**
  - ISO / ITU-T
  - framework for the standardization of ODP
  - architecture which supports distribution, interworking, interoperability and portability

Overviews

- **Viewpoints**
  - Enterprise (purpose, scope and policies)
  - Information (semantics of information)
  - Computational (functional decomposition)
  - Engineering (infrastructure)
  - Technology (choice of technology for implement')

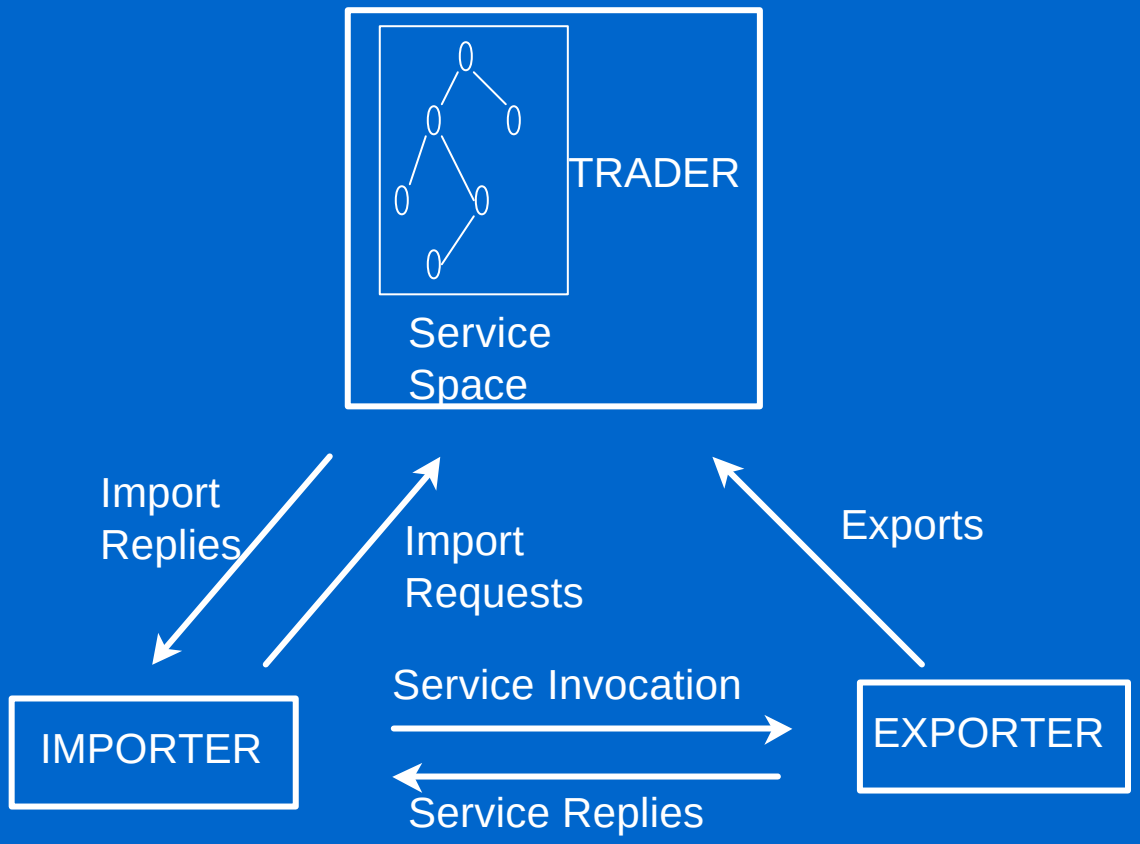
Overview

- RM-ODP Viewpoints and Software Engineering



Overview

- **ODP Trading Function**

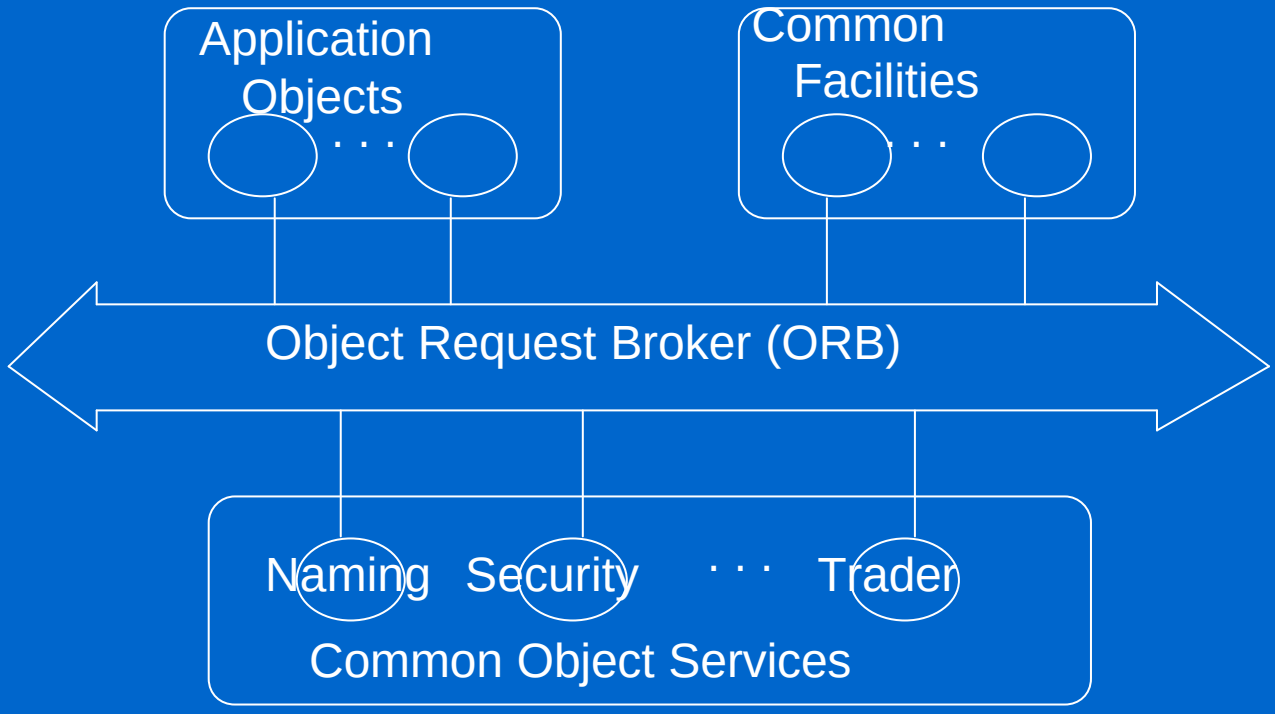


Related Work

- **TRADE Project**
  - TRADEr : service offer manager, service selection manager, type manager etc.
  - Integrating Trading Into DCE
  - extensions of the service type
- **ANSA Project**
  - ANSAware
  - type management function(naming and relationship), not type description function

Related Work

- **OMA / CORBA**

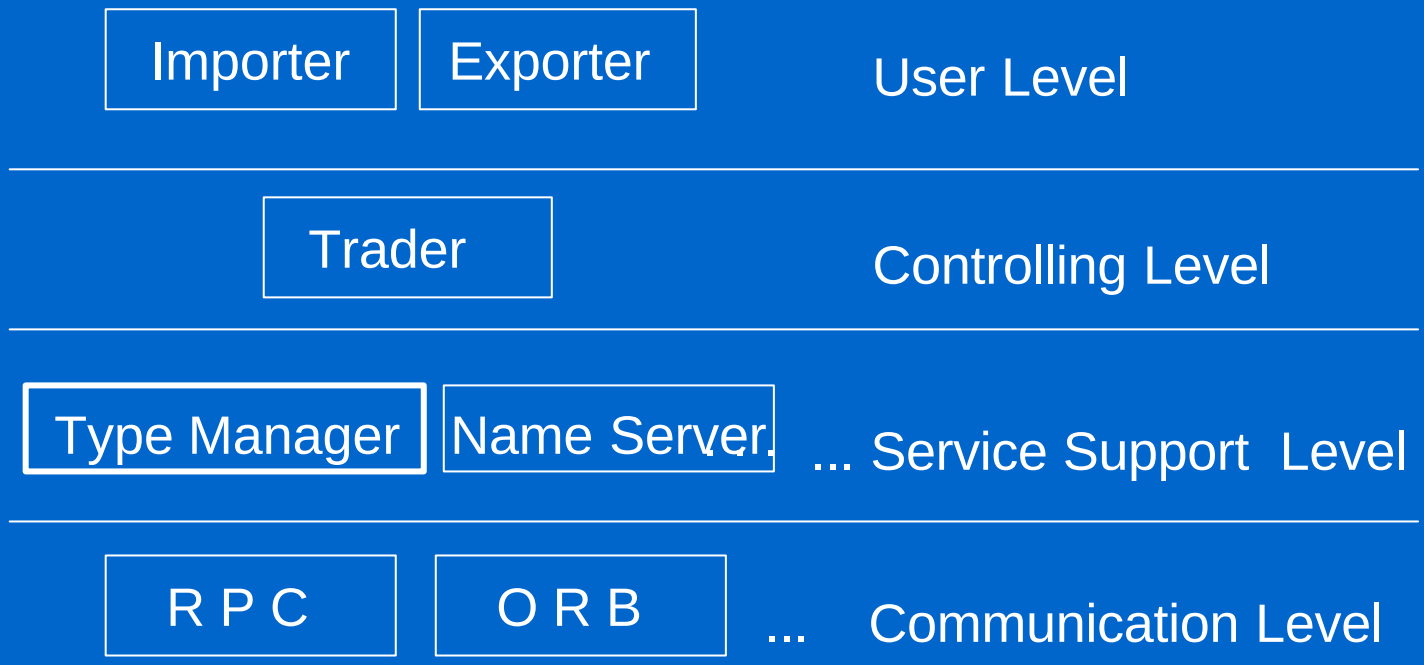


Enterprise Viewpoint

- **Type Manager Purpose**
  - type safety
  - interworking
  - resource discovery
  - system evolution

Enterprise Viewpoint

- **Position**



Enterprise Viewpoint

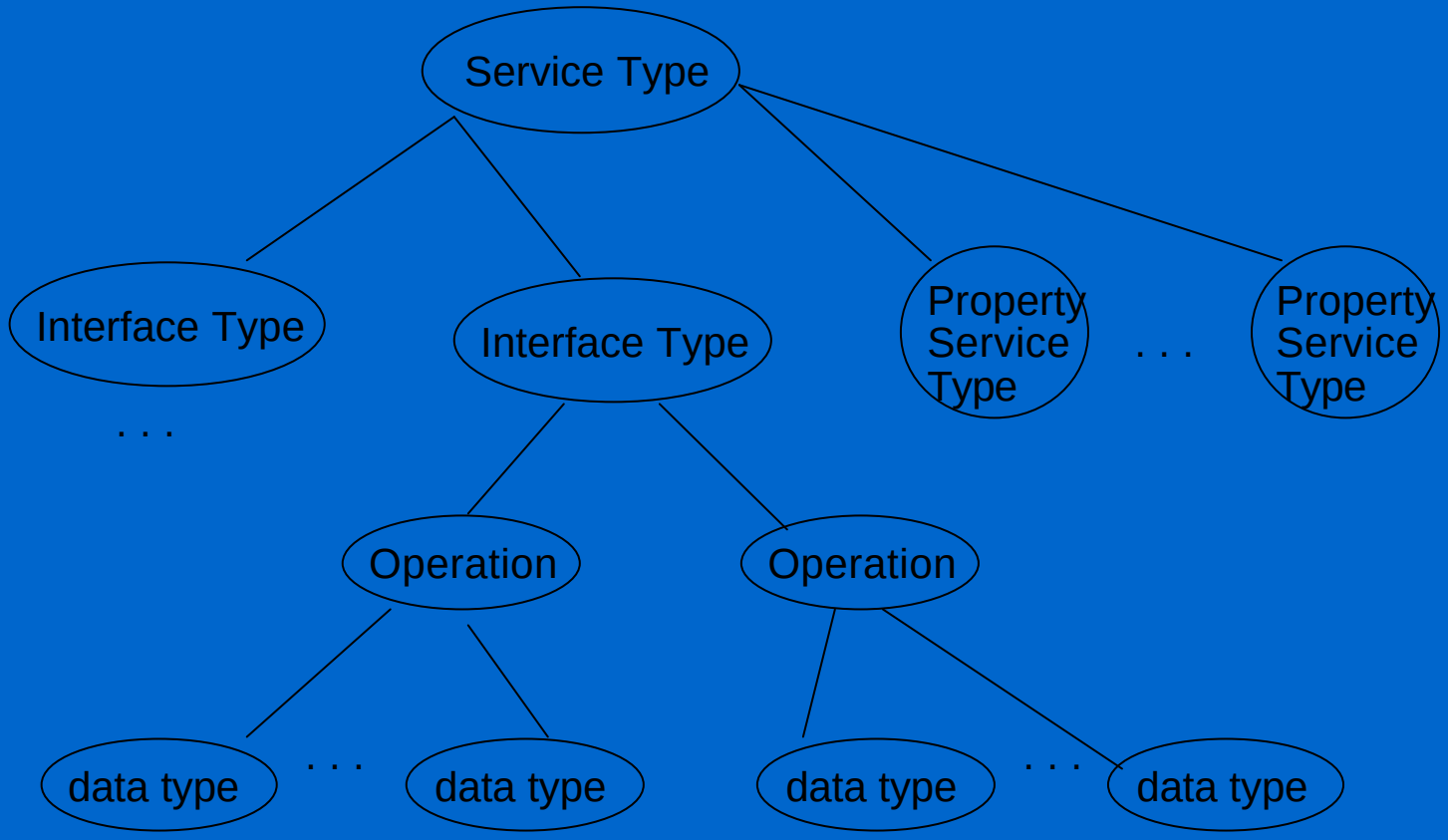
- **Requirements**
  - run time type checking
  - type matching
  - dynamic type selection
- **Policies**
  - Independent Object
  - Autonomous
  - Service Type Model
  - Interworking

Information Viewpoint

- **Service Type Description Model**
  - Object Oriented
  - interested in behavioral types
  - Service Type : interface types, set of service Prop’
  - Interface Type : set of operations
  - Operation type : operation names, name and types of input/results
  - Data type : basic data types, constructed

Information Viewpoint

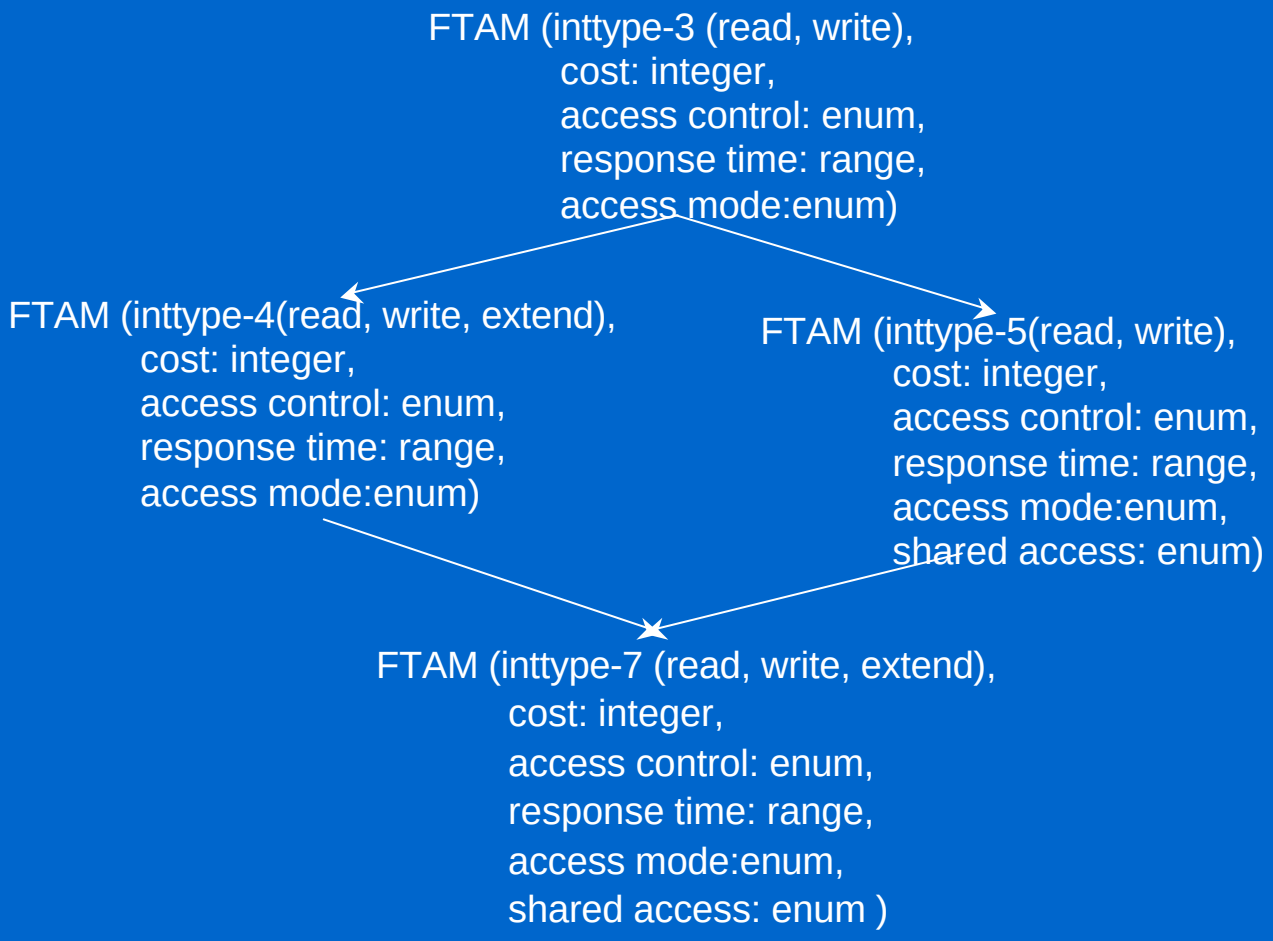
– Service Type Hierarchies



Information Viewpoint

- **Relationships**
  - support for dynamic type matching
  - represent pre-existing relationships
- **Subtype**
- **Conversion**
- **Relationships can arise by**
  - the user supplies this information
  - they are derived from the type description

Information Viewpoint



Information Viewpoint

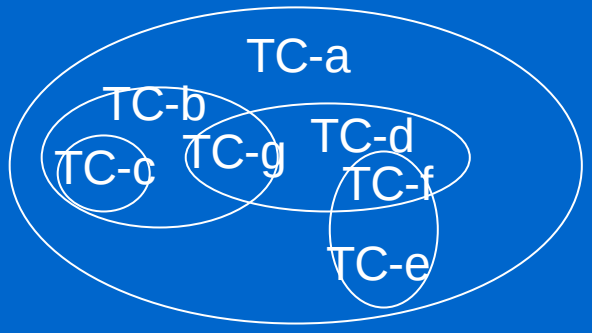
- **Type Declaration**

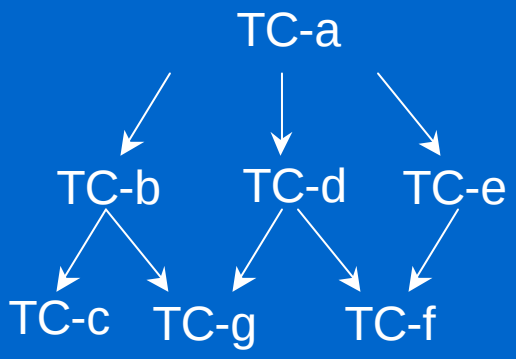
```
servicetype PrintService {
        normal Cost integer;
        fixed_normal Response integer;
        read_only Kind string;
        interfacetype Printing{
                open();
                write();
                close();
        }

    relationship  RelationshipName{
            relationkind subtype;
            sourcetype PrintService;
            targettype  DotPrint;
    }
```

Information Viewpoint

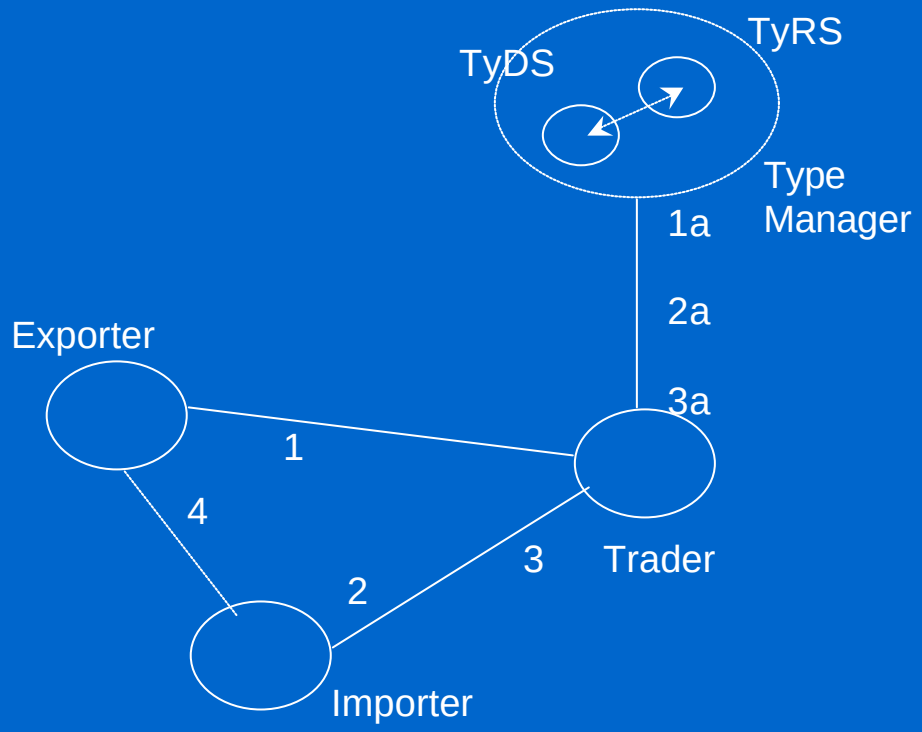
- **contexts**
  - some aspects of the characteristics of group itself
  - some common properties of the services
  - security, naming
  - context structure





Computational Viewpoint

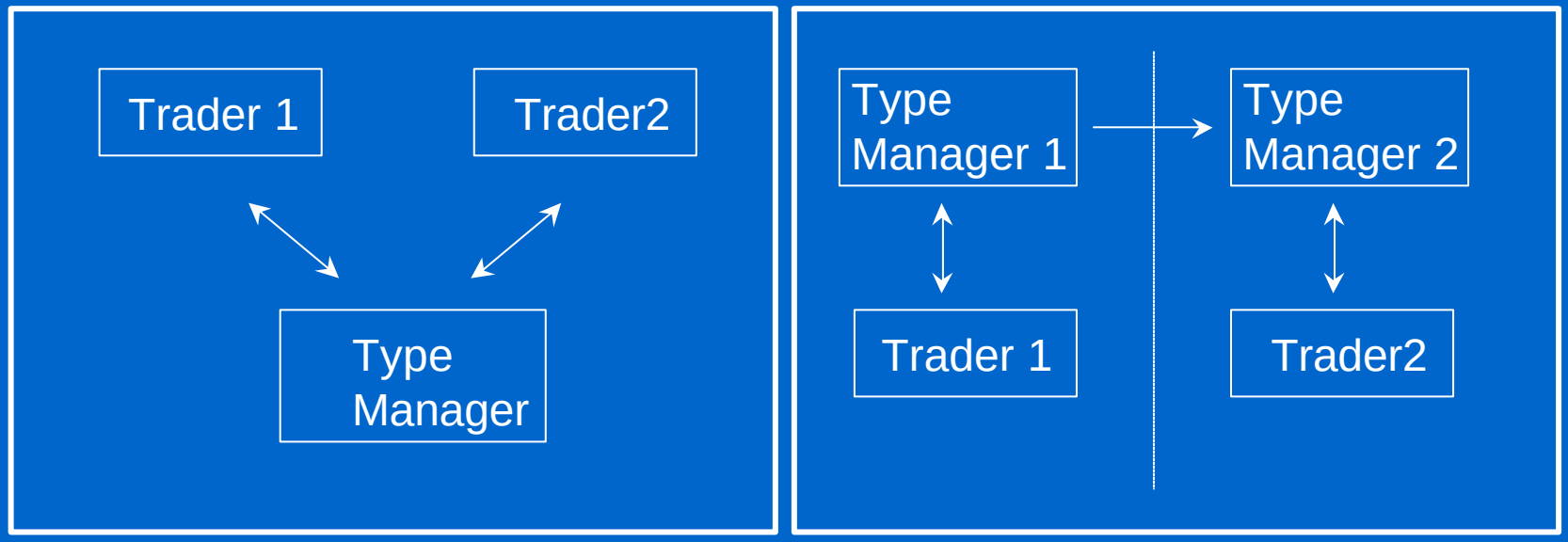
- **Interaction Model**



Computational Viewpoint

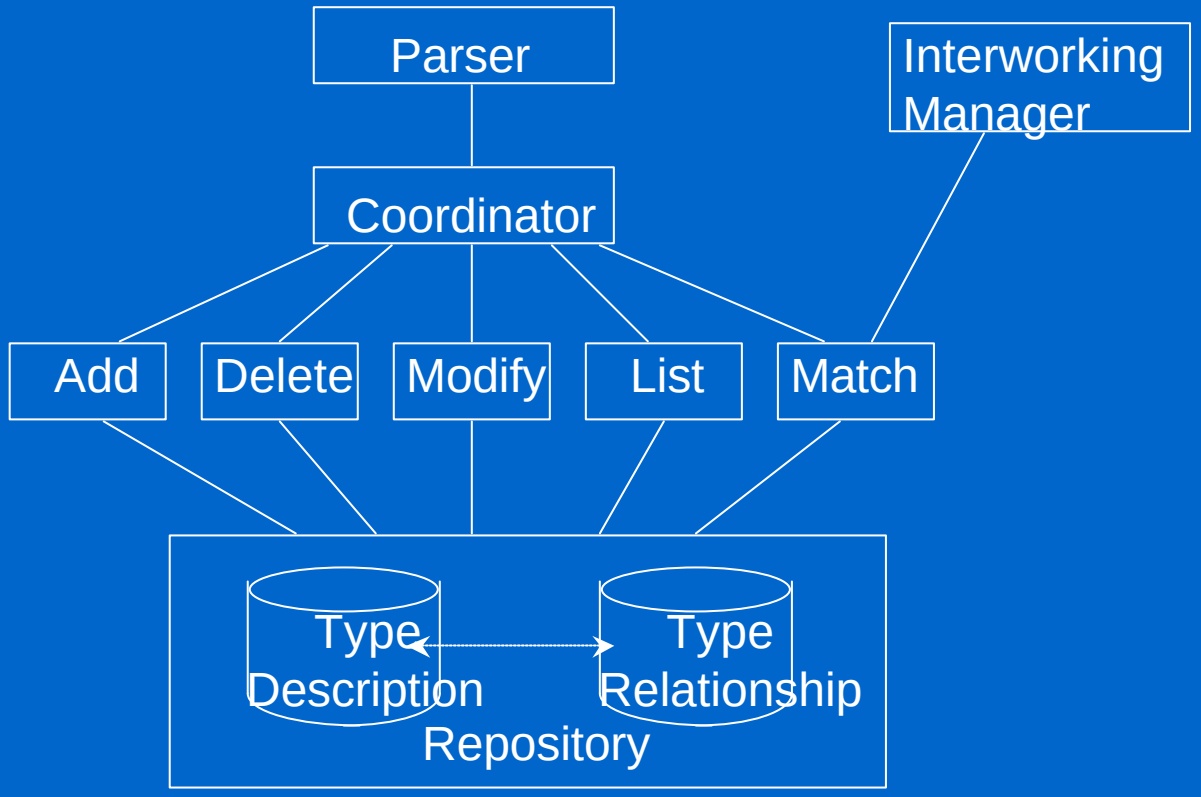
- **Operations on types**
  - add / delete / modify type description
  - add / delete / modify type relationships
  - type match / list
- **Type Manager based Trader Interworking**
  - Simple Form
  - Interworking between Type Manager

Computational Viewpoint



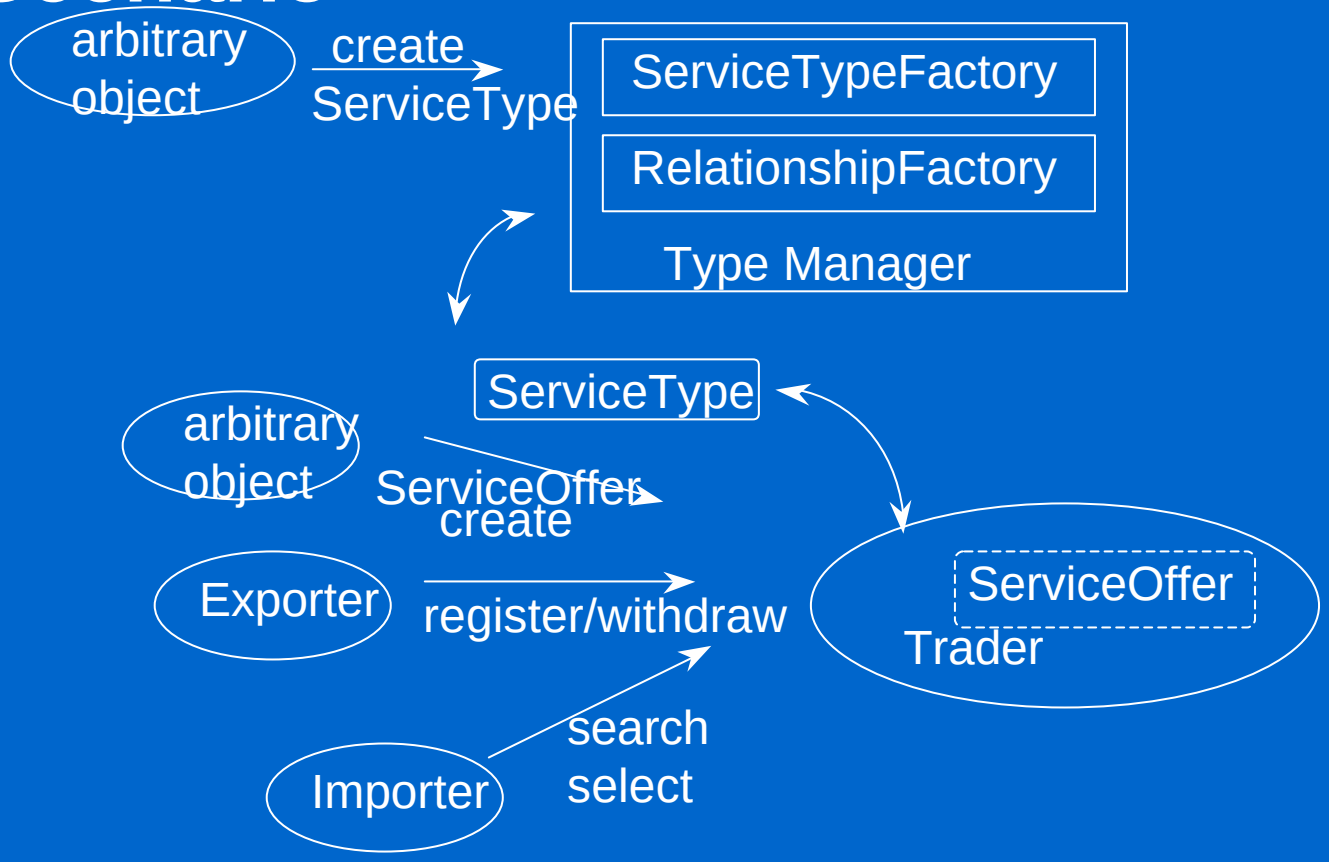
Engineering Viewpoint

- **Architecture**



Type Manager Interface

- **Scenario**



Type Manager Interface

- **Service Type Factory**

```
IncarnationNumber add_servicetype (
        in ServiceTypeName name,
        in ContextName context_name,
        in InterfaceStructSeq inters,
        in PropStructSeq props
) raises (
        InvalidServiceType,
        InvalidContextName,
        InvalidInterfaceName,
        InvalidProperty
);
void delete_servicetype ();
void modify_servicetype();
ServiceTypeNameSeq list_servicetype();
```

Type Manager Interface

- **Relationship Factory**

```
void add_relationship (
        in RelationshipName r_name,
        in Context_Name context_name,
        in SourceType f_name,
        in TargetType t_name,
        in ServiceRelKind rel_type
) raises (
        InvalidRelationship,
        InvalidRelationshipKind,
        InvalidContextName,
        UnknownserviceType
);

void delete_relationship();

RelStruct find_Relationship();
```

Type Manager Interface

```
TypeStruct matches (
    in ServiceTypeName name,
    in ContextName context_name
) raises (
    InvalidServiceType,
    invlaidContextName
);
```

Conclusion & Future Works

- **Qualitative Trading Service**
- **Service Type Declaration Language**
- **Clear Interface Specification**
- **Future Works**
  - Extend type relationships
  - matching algorithms
  - Interworking (Link management, operations)