An ODP-Based Type Manager for Trading Services

GSIT POSTECH
YongHoi Joo
Contents

• Introduction
• Overview
• Related Work
• Enterprise Viewpoint
• Information Viewpoint
• Computational Viewpoint
• Engineering Viewpoint
• Interface Design
• Conclusion & Future Work
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 implementation)
Overview

– RM-ODP Viewpoints and Software Engineering

Enterprise

Policy

Information

Computational

Engineering

Technology

RM-ODP Viewpoints

Requirement analysis

Functional Specification

Design

Implementation

Software Engineering
Overview

• ODP Trading Function

[Diagram showing import and export processes between IMPORTER, TRADER, and EXPORTER]
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

Importer  | Exporter  | User Level
-----------|-----------|-------------
Trader     |           | Controlling Level
-----------|-----------|-------------
Type Manager  | Name Server, | ... Service Support Level
-----------|-----------|-------------
RPC        | ORB       | ... Communication Level
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

FTAM (inttype-3 (read, write),
cost: integer,
access control: enum,
response time: range,
access mode: enum)

FTAM (inttype-4 (read, write, extend),
cost: integer,
access control: enum,
response time: range,
access mode: enum)

FTAM (inttype-5 (read, write),
cost: integer,
access control: enum,
response time: range,
access mode: enum,
shared access: enum)

FTAM (inttype-7 (read, write, extend),
cost: integer,
access control: enum,
response time: range,
access mode: enum,
shared access: enum)
Information Viewpoint

• **Type Declaration**

```plaintext
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

Trader 1

Trader 2

Type Manager

Type Manager 1

Type Manager 2

Trader 1

Trader 2
Engineering Viewpoint

- Architecture

Parser

Coordinator

Add Delete Modify List Match

Type Description Repository

Type Relationship

Interworking Manager
Type Manager Interface

**Scenario**

- Arbitrary object
- ServiceTypeFactory
- RelationshipFactory
- Type Manager
- ServiceType
- ServiceOffer
- Exporter
- Trader
- Importer
- create ServiceType
- register/withdraw ServiceOffer
- select
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**

  ```java
  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,  
    UnknownSourceType  
  );
  
  void delete_relationship();
  
  RelStruct find_Relationship();
  ```
Type Manager Interface

TypeStruct matches (  
   in ServiceTypeName name,  
   in ContextName context_name  
) raises (  
   InvalidServiceType,  
   invalidContextName  
);
Conclusion & Future Works

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