Federated Trading Model for Object Interoperation in DCE

Li Gui Yin Chaowan
Shenyang Institute of Automation
The Chinese Academy of Sciences, Shenyang 110015,
P.R.China

Zheng Huaiyuan
Department of Computer Science & Engineering
Northeastern University, Shenyang 110003
P.R.China

Abstract In this paper, the federated trading model, based on the DCE’s Directory Service and the ODP’s Trading Service, for object interoperation in DCE is presented. The implementation architecture of the federated trading model is discussed.

Keywords: DCE, ODP, Trader, FTM (Federated Trading Model)

I. INTRODUCTION

Distributed Computing Environment (DCE) is a software development and running environment which supports the creation and operation of distributed application can be seen as being composed of objects, which interact using the client/server paradigm. An object which takes the role of a service provider is known as a server. An object which takes the role of a service user is known as a client. The use of Remote Procedure Calls (RPC) or Object Request Broker (ORB) allows clients to invoke object services on remote server. The DCE Directory Service allows client, at runtime, to find the current location of a server by searching on the server’s name. That is the DCE Directory provides a dynamic white pages service. The Basic Reference Model of Open Distributed Processing (RM-ODP) is being developed by the ISO/IT-U. RM-ODP is a meta-standard which prescribes a framework for the development of other application standards in the area of Open Distributed Processing (ODP). RM-ODP specifies an architecture that organizes the many divergent components of an ODP system into a coherent whole. A trader is a key component of this ODP architecture. The ODP Trader allows clients to find services information based on knowing attributes of a service at runtime. That is the ODP Trader provides a dynamic yellow pages service[1].

In any large distributed environment, supporting environment that include hardware and software and applications are frequently changing. Additions of new services and deletions/modification of existing services are be expected. Thus, late binding and dynamic administration and selection of services based on service characteristics become essential functionality for any large distributed systems.

The DCE Directory Service provides a dynamic white page service, but DCE currently lacks a yellow pages service. A dynamic yellow pages service allows clients, at runtime, to find a required service based on the service’s attributes. The ODP Trader provides such a service.

In this paper, we present the federated trading model based on the ODP Trader for advertising and discovering of object services dynamically in DCE. A trader is an object to which another object can advertise (export) its services and from which another object can discovers (import) its needs from the set of advertised services in DCE. Federation is the cooperation mechanism among heterogeneous, distributed and autonomous traders. Each component trader manages its own set of federated trader. The users can only import and export services indirectly to and from remote traders via a user’s local trader. A federation contract is used to document the agreement between two federating traders. The contracts also contain mapping functions for requests and results, to be expressed in a canonical form for communication between traders.

This paper is organized as follows. Section II introduce and analyze the ODP trader services. Section III presents the DCE relevant services. Section IV discuss the Federated Trading Model (FTM) in DCE and presents the implement architecture of FTM in DCE. Finally, in Section V, we conclude this paper.

II. THE ODP TRADING SERVICES
A trader is a key object to which another object can advertise its services and from which another object can buy its needs from the set of advertised services in a distributed environment [2]. The main task of a trader function is the mediation and management of services in open distributed systems. For this purpose, the trader first offers mechanisms for arranging and categorizing various service types, and then supports potential service clients with specific service selection strategies. Thus, the functionality of a trader object can be compared to, e.g., a yellow pages service which categorize service kinds and provides service selection support based on different service properties. The most important formal concept underlying such a trading function is the notion of a service type. A service type may contain interface type which specify the operational service interfaces in terms of operation signatures as well as service property type which add additional semantic details to the service type description. A second important mechanism for service structuring in open distributed systems is based on service contexts in which service offers and be grouped and located. The interactions of clients and servers with a trader object in an open distributed environment are shown in Fig. 1.

According to the ODP trader function, When a server wishes to advertise its service, it registers a service offer with the trader (step (1)). A services offer contains the service type, the properties values of a services that a server provides and the context in which the service shall be exported. The process of advertising a service is called exporting, and the object that registers the service with the trader is known as the exporter. Service offers are stored by the trader in its database and can be structured into groups called context. An exporter can withdraw a service offer or replace a service offer when the characteristics of a service change (step(2)).

When a client requires a service, it first issues a service request to the trader to find a suitable service (step(3)). Such a service request contains the desired service type, the desired service properties, and a search context. Based on this information, the trader then determines appropriate service providers and selects — if necessary — the best matching service offer. Subsequently, the necessary binding information is returned to the client (step(4)). This process is called importing, and the object that requests the service is known as the importer. After a suitable service is selected, the client can interact with the selected server by invoking the operations provided by the service (step(5), (6)).

III. DCE SERVICES

Technologies capable of supporting a distributed environment have been available for at least the last decade. DCE from the Open Software Foundation (OSF) provides an integrated set of support services and application programming interfaces for the development of distributed application in open heterogeneous environments. The main goal of DCE is to provide users and applications of distributed computing environments with a homogeneous view which hides much of the complexity and heterogeneity of the underlying hardware and system software components. Therefore, DCE provides ways to develop platform-independent distributed applications, based on de-facto standardized application programming interfaces. Basic DCE services are the Remote Procedure Call (DCE RPC), The Thread Service, the Cell Directory Service (CDS), The X.500 compliant Global Directory Service (GDS), the Security Service, and the Distributed Time Service (DTS). Additional services are the Distributed File Service (DFS) and the Diskless Client Support (DSS), each of these services provides a single unified application programming interface.

A. Service Management and Access in DCE

The basic unit of service management and structuring in
DCE is the so-called **cell**. Each cell includes its own Security Service, a Cell Directory Service, and a distributed Time Service. The Cell Directory Service plays a central role for storing all information concerning actual services of the DCE cell (e.g. configuration and binding information). Access to information of foreign cells is provided by an X.500 compliant Global Directory Service which offers a worldwide available name space connecting several different organization domains.

Binding is the process of connecting two remote objects together by establishing a logical channel for interaction between them. In the client-server paradigm, this interaction is uni-directional (i.e. the client invokes operations on the server). Thus, the client is responsible for establishing the channel.

In the following, we give a concrete example of how to establish a binding between a client and a server provider using the DCE Cell Directory Service[4]. In a DCE based open system environment, a server has to do the followings in order to register a new interface: Firstly, to inform the RPC runtime system about the offered interface type using an interface handle. Secondly, binding vectors are generated which contain binding information. These along with the interface type and the object identification are then registered at the local host's **Endpoint Mapper** which manages the mappings from interface and object identifier to communication endpoints of current running service providers at the local host. Finally, the same information are registered in the DCE Cell Directory Service under a distinct entry name.

In order to obtain the necessary binding information from a server offering a desired interface, a client has to execute the followings: First, the DCE Cell Directory Service must be called in order to obtain server binding handles of a distinct server instance, which include the server entry name, the interface identifier, and the object identification have to be supplied. The client can then bind to the server and start to execute its remote procedure call by using these binding handles. Subsequent the remote procedure calls can be transferred directly to the corresponding service provider.

The process of obtaining a binding handle at, or before, compile-time is known as static binding. The process of obtaining a binding handle at runtime is known as dynamic binding.

**B. DCE Support for Service Trading**

Although DCE provides several useful prerequisites for development of open distributed system application, it still lacks some important features, especially in order to support a trading function as, e.g. specified by ODP. Currently, there is no support for service mediation in DCE, i.e. the clients are entirely responsible for the selection of appropriate service offerings.

**IV. FEDERATED TRADING SERVICE FOR DCE**

In a large distributed environment, a federated trading is a collection of cooperating but autonomous component traders. By federating, an advertisement of a component trader will be known to a wider audience and a client of a component trader will have a wider market from which to choose its needs.

A federation of traders can also be formed when the number of objects administered by a trader has become so large that it no longer become possible for the administration to provide efficient service to the trader’s user. By partitioning the administrative responsibilities of the trader database into autonomous but cooperating traders, federated trading can be formed.

A federation of traders allows member traders to have controlled and partial sharing of information. The amount of information shared between traders will depend upon the amount of cooperation between the autonomous components. Each individual component of the federation must be able to carry out its normal local operations without external interference.

Ideally, in a federation, mechanisms must exist to allow cooperation among traders that are autonomous, heterogeneous and distributed. Furthermore, the goals of Open Distributed Processing (ODP) mandate the transparency of distribution so that a user of a federated trader will associate with only one trader and transparently
access other traders via that trader. Different models of federation provide different degrees of autonomy and transparency for each cooperating trader.

A. A FEDERATED TRADING MODEL

The federated model is a decentralized federation of traders[2]. Each trader manages its own set of federated traders. To be part of a federation, a trader must be able to import from at least one other trader or be able to export into at least one other trader in the federation; A trader that imports from a remote trader has an import contract with that remote trader. A distinct import contract exists for each remote trader that the local trader imports from. Each import contract states the service types and type structures available in the remote trader, and the rules for mapping requests of the local trader to the remote requests understandable by the remote trader; A trader that exports into a remote trader has an export contract with the remote trader. There is an export contract for each remote trader that the trader exports to. Each export contract of a trader states the extent of allowed access of a remote trader to the local trader’s database, and the service types and type structure. It also states the rules for mapping requests and results between the local trader and the remote trader.

A client of a trader search for a service in the local trader database and, if necessary, the local trader searches through its import contracts. If the required service type is available in a remote trader, then the local trader maps the request into a remote request and sends it to the remote trader. The remote trader initiates a search of its database, constrained by the corresponding export contract. When the search is completed, the results are returned to the importing trader.

Figure 2 shows the interconnection between traders via the contract. The dotted arrows show a search path from one trader to another using negotiated information contained in the contract. Figure 3 shows the type directory hierarchy seen by users of trader 1.

B. A IMPLEMENTATION ARCHITECTURE OF FEDERATED TRADING

According to the trader’s role in service management, selection, and access in open distributed system, the federated trader's core component are the service offer manager, the service selection manager, the dynamic type manager, the access control manager, and the federating manager. The implementation architecture of the federated trader is shown in Fig. 4.

The dynamic type manager is one important component of a federated trader. It provides the basis for a common understanding and for comparison of services types as a main structuring technique for service requests and offerings in open distributed environments. Different notions of service type serve in this context as formal abstractions of service characteristics, i.e. common properties of classes of service.
instances of a distinct service type.

In general, several levels of type management can be distinguished within a trader's type manager. Some possible steps in such a continuum from simple name-based service type descriptions up to full semantic specification of a service type are discussed in [4]. They are used as a basis for step-wise extension of type description and management functions as already available in DCE, up to what is really needed for a future trader's dynamic type manager component.

An other important core component of the federated trader is the service offer manager. The service offer management is entirely based on the DCE Cell Directory and Global Directory services. The respective interface contains operations for a variety of service offer management functions, e.g. inserting, deleting, reading, and modifying of service offers. Also attribute based search operations are provide which facilitates the implementation of the trader's selection strategies. For a service provider to advertise a service offer in the trader, it is possible either to use CDS entry names, or to use X.500 entry names. service offers are stored in a special format representing the offered service type and all corresponding interface types, the current values of the static service attributes, the interface reference for server binding, and, optionally, a service type description. This service type description can be used by programmers to develop corresponding DCE client applications and to generate the RPC stubs necessary for communicating with the DCE server.

The service offer manager and dynamic type manager discussed above are mainly responsible for structuring, storing and managing the service type, the other components of the federated trader includes the access control manager and the service select manager and the federating manager. The access control manager is responsible for safe type checking of the request of the importer based on the DCE Security Service. The service select manager is responsible for searching for the required service type based on the importer's select constraints. The federating manager is responsible for creation and administration and execution of the federation with other traders. For example, if the local type system does not have the required service type, the federating manager will import the required service type from other trader based on the federated contract.

V. CONCLUSION

The DCE Directory Service provides a dynamic white page service while the ODP trader provides a dynamic yellow pages service. In this paper, we present the federated trading model for object interoperation in DCE, and discuss the implement architecture of the federated trader. This architecture is based on the DCE relevant service such as the DCE binding services and the DCE Cell Directory Service and Global Directory Service. A concrete implementation goal is an orthogonal and smooth integration of federated trader functions into, in particular, already available service registration and management mechanisms in DCE.

VI. REFERENCE