Federating Traders: an ODP Adventure

Mirion Bearman
Information Sciences and Engineering
University of Canberra
Belconnen ACT 2616
Australia

Kerry Raymond
Centre for Information Technology Research
University of Queensland
St Lucia QLD 4072
Australia

Abstract

In this paper we propose a model of decentralized federation of traders. Each component trader manages its own set of federated traders. A federation contract is used to document the agreement between two federating traders. The importer's portion of the federation contract forms an import contract that states the service types available in the remote trader. The exporter's portion of the federation contract forms the corresponding export contract that states the extent of allowed access of a remote trader to the local trader's database. The contracts also contain mapping functions for requests and results, to be expressed in a canonical form for communication between traders.

There are two protocols: one for a user with its trader and another for a trader with another trader in the federation. A user has access only to the protocols of its local trader.

We present our model using the five viewpoints of ODP, and demonstrate the need to recursively apply the five viewpoints to engineering mechanisms of a system. We learn that there is a need for a third-party security mechanism and a separate type management function in ODP. We also discover that transparency does not imply secrecy. Finally, our experience confirms that while separation and location can be made transparent, autonomy and incremental change cannot be made completely transparent in an Open Distributed Processing environment.

Keyword Codes: C.2.0; C.2.4
Keywords: Computer-Communications Networks, General; Distributed Systems

1 Introduction

A trader is an object to which another object can advertise (export) its services and from which another object can buy (import) its needs from the set of advertised services in a distributed environment [1,8]. A federated trader is a collection of cooperating but autonomous component traders [2,6]. 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 be formed when component traders wish to:

- Offer services to a wider community by advertising services for sale,
- Have the opportunity to buy by reading what is for sale, or
- Have the opportunity to buy from a wider community by advertising services to buy.

This assumes that there exists a known mechanism of communication between the traders that wish to federate.

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 becomes possible for the administration to provide efficient service to the trader's users. By partitioning the administrative responsibilities of the trader database into autonomous but cooperating traders, federated traders 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 [3] 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.

In this paper, we discuss some fundamental issues of federated traders. We then present a model of federation using the five viewpoints of ODP [4]. Our model provides a decentralized federation which corresponds closely to that of the 'loosely coupled' federated distributed database of Sheth and Larsen [5]. We conclude by outlining the lessons learned about ODP from our study of federated traders.

2 Fundamental Issues of Federated Traders

A federation of traders consists of autonomous and heterogeneous traders in a distributed environment. However, each trader must provide its users with transparency of distribution and heterogeneity, while maintaining its autonomy. Fundamental issues of federated traders include separation, heterogeneity, autonomy, and transparency. Unfortunately, these issues often present conflicting requirements in a federation. We will now briefly describe each issue.

2.1 Separation

The database of each trader will (usually) be stored on different computer systems that are physically distributed and connected by communication networks. Management of

individual components, as well as the management of the federation of components, will also be distributed (but might be logically centralized). Such separation can often cause delay in access times and consequent loss of performance. Unfortunately, introducing replication to improve access times caused by separation creates the problem of maintaining consistency among copies. However, separation does provide increased availability and reliability of the system as a whole.

2.2 Heterogeneity

Each trader can have its own mechanisms for access control, concurrency control, and recovery. Each trader can have its own directory hierarchy [7], and can support different service types and type hierarchy. A given service type might have different optional attributes at each trader. Also there can be differences in semantics of services and attributes. For example, the quality of printing might be expressed as a digit of 0 to 9, where 0 means the best quality in one trader but the worst in another.

2.3 Autonomy

Each trader can support its own service types, services, and directories. It decides on the representation of its data (e.g. cost can be represented as an integer or a string). It decides on the semantics of its own data. When a trader federates, it must be able to maintain its existing exported services and service types, and be able to carry out local operations without external interference. Finally, a trader decides when to join or leave a federation and how much of its database it will share with other traders.

However, it should be noted that it is also a trader's autonomous choice to give up some of its rights if it is consistent with its enterprise policy.

2.4 Transparency

The distribution transparency ensures that when a user accesses data from a remote trader, the user does not need to know where the data is or whether that data is distributed. The heterogeneity transparency ensures that a user's request will be in the same format and will have the same meaning when accessing a local or a remote trader. Similarly, the results of the request will be expressed in the same format and will have the same meaning whether the result is supplied by a local or a remote trader.

To achieve such transparencies, mapping functions are required between the local and remote traders. For example, requests and results from both traders can be translated into a canonical form for communication between the traders.

There will be two protocols: one for a user with its trader and another for a trader with another trader in the federation. A user has access only to the protocols of its local trader.

3 Motivation for Federation

It is possible for a user to import from more than one trader without federation. A user can be associated with a set of traders and be a direct client of each trader. This means that the user must know the syntax and semantics of the service types known to each trader. It also involves unnecessary accesses to all trader databases.

A trader can also be a client of another trader. A user can access a remote trader via its local trader which acts as a client to the remote trader. Again there are the problems of syntax, semantics, and inefficiency.

However, if the traders are suitably federated, then the user need only be a direct client of one trader, accessing the other traders indirectly via the federation. There are a number of benefits of the federated approach.

Domain Organization Federation allows a trader to organize its enlarged trading domain (the collection of domains and sub-domains visible from itself and other traders) into a single directory hierarchy, making the federation location and separation transparent to its users. Figure 1 shows the directory hierarchy as seen by a user of Trader 1. The user need not be aware that parts of the directory hierarchy are managed by other traders. (The shaded triangles indicate sub-directories of traders in a federation.)

Heterogeneity Federation provides a canonical data type mechanism to/from which local types and local type instances can be translated for transmission between traders.

Security Federation provides for access control and accounting in operations executed at remote traders to be performed on the basis of the end-user as the client rather than the intermediate trader(s).

Efficiency Federation allows traders to negotiate the service types that are permitted to pass between them and to advise on the availability of instances of these service types in order to reduce unnecessary accesses via the federation.

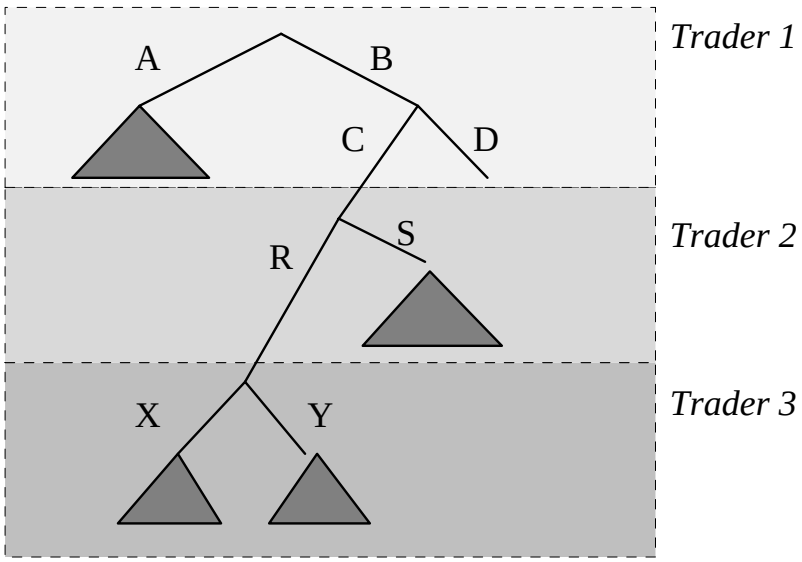

Figure 1: Directory Hierarchy seen by Users of Trader 1

4 A Model for Federated Traders

Our model is a decentralized federation of traders. 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 requests understandable by the remote trader (and mapping functions for results from remote trader to local 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 structures. It also states the rules for mapping requests and results between the local trader and the remote trader.

For each export contract, there exists a corresponding import contract in the remote trader.

Thus, an exporter's view of a federation is the set of export contracts with different remote traders while an importer's view of a federation is the set of import contracts with remote traders.

A client of a trader searches 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 (transformed if necessary) are returned to the importing trader.

Figure 2 shows the interface between traders via the contracts that implement the directory hierarchy view shown in Figure 1. The dotted arrows show a search path from one trader to another using negotiated information contained in the contracts.

4.1 Enterprise Viewpoint

In the enterprise viewpoint, the policy requirements for establishing imports and exports between traders are established. A trader can import from more than one remote trader and it can export into more than one remote trader. Each trader manages its own federation. There is no need for a central federation agent. The federated traders must be able to provide transparent access to users of heterogeneous, distributed and autonomous traders. The users can only import and export services indirectly to and from remote traders via a user's local trader. Known communication paths exist between potential traders of a federation and standardized service types are known to all. Note there can be unstandardized service types and there should be mechanisms to manipulate instances of these types through federation.

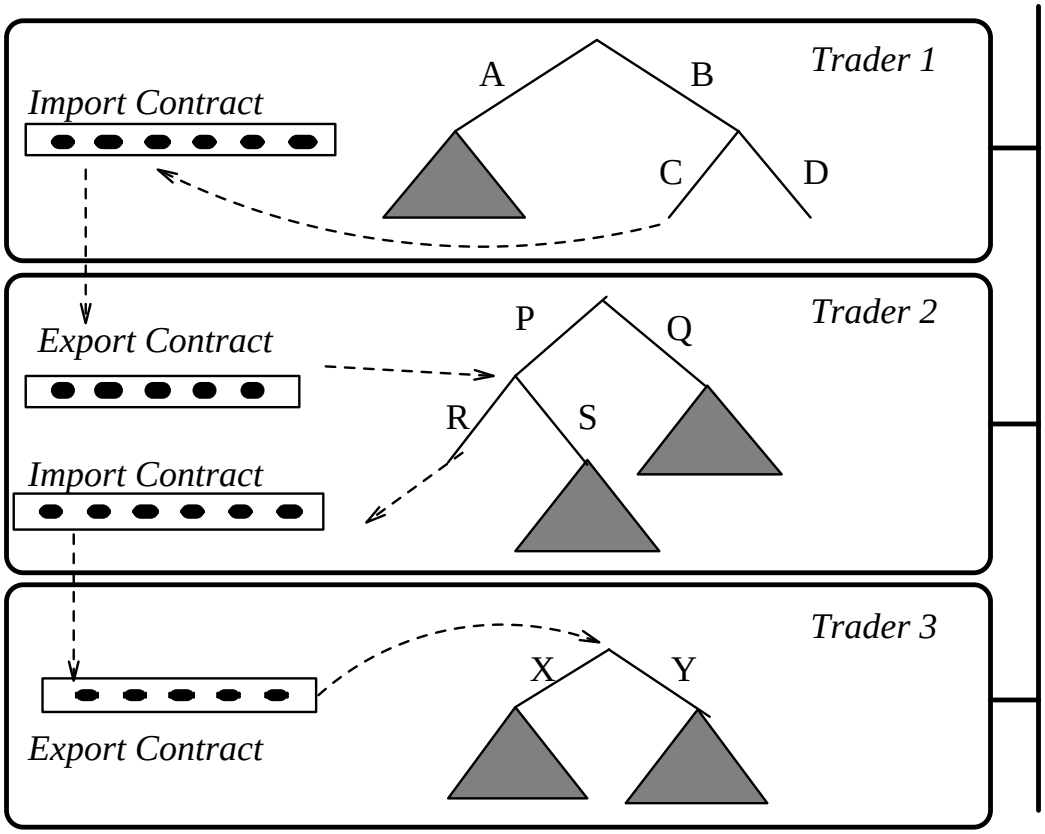

Figure 2: Engineering View of Decentralized Federation

4.2 Information Viewpoint

For federation, there must be a method to describe :

- Trader identification
- What is available for export by a trader
- Types of services accessible by an importing trader
- Restrictions on access to an exporting trader's database

This information is provided by catalogues, federation contracts, import contracts, and export contracts, which are described below.

4.2.1 Catalogue

A catalogue (a name taken from [6]) contains the descriptions of directories [7] that are available for export by a trader and the descriptions of the associated service types.

A catalogue can be represented as a standardized service type. Each entry in the catalogue contains:

- an identifier
- the portion of the directory hierarchy that the importer can access (directory name)
- a list of service type names (expressed in a canonical form) that are currently available for export
- textual description of the directory and the service types
- chaining flag (indicates if it can be chained for further exports)
- list of permitted federated operations (as some federated operations might unacceptably restrict the autonomy of a trader)

The type structure is not included in the catalogue as it is assumed that the type structure can be obtained from the type manager.

The chaining flag is used in the import contract to indicate if chaining is permitted. If trader A has an import contract with trader B and B has an import contract with trader C, then trader A can indirectly access trader C via the two import contracts if the contract between B and C permits chaining.

When a trader wishes to export, it first prepares a catalogue. It can send the catalogue to a specific trader by acting as a user to that trader and exporting the catalogue. That is, a trader will act as a client and export the standardized CATALOGUE service type to another trader in the hope of establishing a federation. While some of the fields of a catalogue entry are not essential to the importing trader, they contain information which might be used by some importing traders or their administrators.

Thus, when a trader wishes to export to other traders, it must first prepare a catalogue. And, if a trader wishes to import, it must first obtain a catalogue.

The exporting trader stores its catalogue within its directory hierarchy [7].

4.2.2 Federation Contract

In principle, a federation contract documents the agreement to federate between two traders. This means that the importing trader can perform the agreed federated operations on the agreed service types in the agreed directory of the exporting trader.

Federation contracts are subsets of the published catalogue and must be negotiated between traders to establish what, where, and how services can be shared between traders.

The importing trader uses the published catalogue to form a federation contract by selecting entries and possibly further selecting within each entry. For example, the importing trader might not desire all the federated operations offered by the exporter or not want the ability to chain.

A federation contract contains:

- globally unique contract identifier (e.g. $< ImporterId, ExporterId, Count >$)
- identification of the two traders (assuming it isn't contained in the contract identifier)
- a directory name of the exporter
- a list of service type names that are currently available for export
- a list of service type names that are required by the importer
- the exporter's willingness to chain
- the importer's willingness to chain
- the federation operations offered by the exporter
- the federation operations wanted by the importer

In practice, a federation contract is divided into an import contract held by the importer and a corresponding export contract held by the exporter as shown in Figure 3.

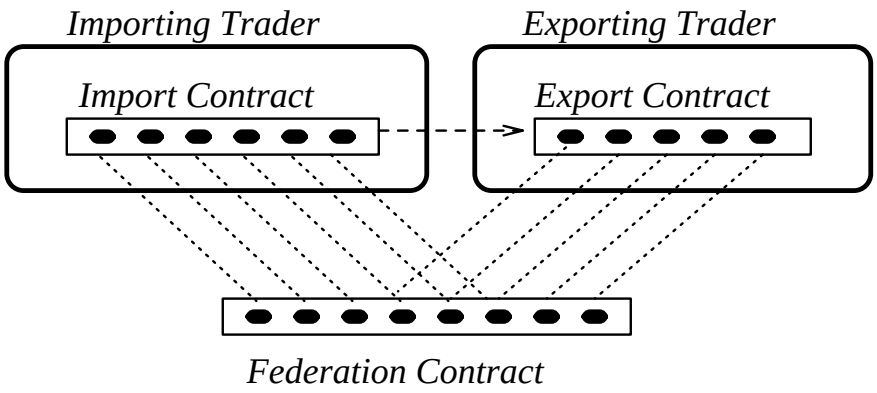

Figure 3: Import, Export, and Federation Contracts

4.2.3 Import Contract

An import contract is an agreement between an importing trader and an exporting trader. The import contract is kept by the importing trader and forms the importer's portion of the federation contract. This contract contains:

- contract identifier
- identification of the two traders
- a list of service type names that are currently available for export ('Available')
- a list of service type names that are required by the importer ('Wanted')
- the exporter's willingness to chain ('Exporter Chaining')
- the importer's willingness to chain ('Importer Chaining')
- the federation operations offered by the exporter ('Exporter Operations')
- the federation operations wanted by the importer ('Importer Operations')
- the mapping functions between local and canonical service types

The contract permits chaining only when both importer and exporter are willing ('Exporter Chaining' *and* 'Importing Chaining'). The federation operations available through the contract are the intersection of the operations offered by the exporter and wanted by the importer ('Exporter Operations' $\cap$ 'Importer Operations').

The importing trader stores the import contract within its directory hierarchy using the standardized IMPORT-CONTRACT service type.

4.2.4 Export Contract

An export contract is also an agreement between an importing trader and an exporting trader. However, the export contract is kept by the exporting trader and forms the exporter's portion of the federation contract. This contract contains:

- contract identifier
- identification of the two traders
- the directory name within the exporter
- a list of service type names held in the directory ('Held')
- a list of service type names that the exporter is willing to make visible to the importer through the contract ('Willing')
- the exporter's willingness to chain ('Exporter Chaining')
- the federation operations offered by the exporter ('Exporter Operations')
- the mapping functions between local and canonical service types

The intersection of the service types held in the directory ('Held') and those that the exporter is willing to make visible ('Willing') yields the available service types ('Available' of the import contract).

The exporting trader stores the export contract within its directory hierarchy using the standardized EXPORT-CONTRACT service type.

4.2.5 Federations

The import federation of a trader is the set of traders visible to that trader through import contracts (including those reachable by chaining).

The export federation of a trader is the set of traders from which that trader is visible through export contracts (including those accessible by chaining).

The type domain of a federated trader is extended to include the service types accessible through import federation.

Therefore the total trading scope of a trader includes the trader database of each member in the federation as constrained by the export contracts.

4.3 Computational Viewpoint

In the computational viewpoint, trading between traders in the federation involves exports and imports of services of a given type. However, before any federated interaction between the traders can take place, contracts between the traders must first be negotiated to establish what are to be imported and exported. Furthermore, it is important to note that a trader can act as a client of another trader as distinct from a trader acting on behalf of a user in a federation.

Thus, the operations between the traders in a federation can be separated into two categories:

- actions for federation, where a trader acts as a client of a remote trader dealing with the federation related service types of CATALOGUE, IMPORT-CONTRACT, and EXPORT-CONTRACT
- federated operations, which the trader carries out on behalf of its users

4.3.1 Distributing the Catalogue

For an exporting trader to initiate a federation, the exporting trader sends its catalogue to a potential importing trader. This catalogue is sent by the exporting trader acting as a client of the importing trader and using the Export primitive [2,7] of the importing trader and the standardized service type CATALOGUE.

The operation will return an acknowledgement by the remote trader or an appropriate error indication. The successful execution of this action will result in the catalogue being known to the nominated trader. If the remote trader is interested in the exports listed in the catalogue, it can initiate contract negotiation by the Exchange Contracts primitive (see Section 4.3.3).

4.3.2 Requesting the Catalogue

A trader wishing to establish an import contract with a potential exporting trader must first obtain the exporting trader's catalogue. A request for the catalogue is sent by the importing trader acting as a client of the exporting trader and using the Search primitive [2,7] and the standardized service type CATALOGUE.

The successful execution of this action will return either

- the exporting trader's catalogue (which could be empty)

- no catalogue (the exporting trader had no catalogue)

An empty or missing catalogue indicates that the remote trader has nothing to export at present.

If the importing trader is interested in the exports listed in the catalogue, it can initiate contract negotiation by the Exchange Contracts primitive (see Section 4.3.3).

4.3.3 Establishing a Federation Contract

The Exchange Contracts primitive is used to negotiate federation contracts.

The Exchange Contracts primitive is initiated by the importing trader after creating a proposed federation contract (based on the exporter's catalogue).

This proposed federation contract, together with security information (e.g. the list of users of the importing trader), is sent by the importing trader acting as a client of the exporting trader and using the Exchange Contracts primitive of the exporting trader.

The exporting trader evaluates the proposed federation contract in the light of its internal access policy and replies with one of the following decisions:

- it accepts the contract as is, and confirms the establishment of the contract by completing the details of the federation contract and creating an export contract
- it refuses the contract outright
- it refuses the contract but proposes a further reduced contract

If the importing trader receives a positive response from the exporting trader, it creates the corresponding import contract.

If a negative response is received with a reduced contract, the importer can evaluate the reduced contract and if it is acceptable, can initiate the Exchange Contract primitive again with the reduced contract.

4.3.4 Listing Federation Contracts

An exporter with an export contract can request information about its corresponding import contract. The request is sent by the exporting trader acting as a client using the Search primitive of the importing trader and the standardized service type IMPORT-CONTRACT.

The successful execution of this Search will list:

- the current location of the import contract
- security information (e.g. the list of current users of the importing trader)
- the details of the import contract

An importer with an import contract can request information about its corresponding export contract. The request is sent by the importing trader acting as a client using the Search primitive of the exporting trader and the standardized service type EXPORT-CONTRACT.

The successful execution of this Search will list:

- the current location of the export contract
- the details of the export contract

Subject to access control, import and export contracts can be inspected by any user.

4.3.5 Modifying Federation Contracts

An exporter can modify the import contract in the following circumstances:

- Changes in the set of service types that are exported in the exporter's sub-tree nominated by this contract.
  'Held' in the export contract and 'Available' in the import contract are updated.
- Changes in the set of service types that the exporter is willing to make visible through this contract.
  'Willing' in the export contract and 'Available' in the import contract are updated.

- Changes in the set of service types known to the exporter can lead to changes in the set of service types that the exporter is willing to make visible through this contract.
  'Willing' in the export contract and 'Available' in the import contract might be updated.
- Changes in the enterprise policy with respect to chaining.
  'Exporter Chaining' is updated in both the export contract and the import contract.

- Changes in the enterprise policy with respect to permitted federation operations.
  'Exporter Operations' in export contract and 'Exporter Operations' in the import contract are updated.

The exporting trader modifies the import contract by acting as a client of the importing trader using the Modify primitive [2,7] of the importing trader. The exporting trader modifies its export contract locally with its Modify primitive.

The successful execution of the modification can alter the set of accessible types, the chaining, or the permitted operations agreed to by the contract.

An importer can modify the import contract in the following circumstances:

- Changes in the set of service types that the importer requires through this contract.
  'Wanted' in the import contract is updated.
- Changes in the set of service types known to the importer can lead to changes in the set of service types that the importer requires through this contract.
  'Wanted' in the import contract might be updated.
- Changes in the enterprise policy with respect to chaining.
  'Importer Chaining' in the import contract is updated.
- Changes in the enterprise policy with respect to permitted federation operations.
  'Importer Operations' in the import contract is updated.

These modifications are performed by the importer with its Modify primitive on its import contract locally.

4.3.6 Terminating Federation Contracts

An importer who no longer wishes to continue a particular federation contract can withdraw the export contract by acting as a client to the exporting trader and using the Withdraw primitive of the exporting trader. It also removes its import contract.

An exporter who no longer wishes to continue a particular federation contract can withdraw the import contract by acting as a client to the importing trader and using the Withdraw primitive of the importing trader. It also removes its export contract.

4.3.7 Federation Operations

Ideally, a trader on behalf of a client should be able to perform all standard trader operations (as given in [7]). However, some of these operations (e.g. Delete Directory) might compromise the autonomy of a remote trader and appear to be in conflict with the philosophy of federation. However, if a trader is willing to permit such actions, there is no reason to limit the operations available to federated users. Thus, the set of operations a trader can use on behalf of its clients depends on the enterprise policies of the federating traders and is negotiated during contract establishment.

For each standard trader operation, there is a corresponding federation operation denoted by F-*Operation*. The Invoke operation is an exception.

Although the Invoke operation might involve several traders in selecting services, there is only one invocation. This is done by the local trader, who is best placed to make the final choice of selected service for the client. Therefore there is no requirement for an F-Invoke operation.

The complete list of federation operations is:

- F-List, F-Search, F-Select for importers
- F-Export, F-Modify, F-Withdraw for exporters
- F-Add-Type, F-Delete-Type, F-List-Type for type management
- F-Add-Directory, F-Delete-Directory, F-List-Directory for directory management

In order to use the federation operations, it might be necessary to map local types and instances of local types into canonical forms. This is an essential difference between a trader acting as a client and a trader acting on behalf of a user.

4.4 Engineering Viewpoint

In the engineering viewpoint, details of the export contract and the import contract must be stored and retrievable as needed to implement operations specified in the computational viewpoint. Any translations between the local trader operations and the federated operations between traders are performed in the engineering viewpoint, invisible to the users. In this viewpoint it could also be decided to use ASN.1 to define the types, parameters and operations of the federated traders.

4.4.1 Links to Import Contracts

Pointers to the import contract are set up in the directory hierarchy of the importing trader when an import contract is established. These pointers are at the leaves of the directory hierarchy. For example, when a trader receives a Search request, it looks up its database to find suitable exports by accessing service directories (leaves) of its directory hierarchy. If one of the leaves is a pointer to an import contract, then the import contract is used to generate an equivalent federated operation (F-Search) (see Section 4.4.3) using the information in the import contract. The importing trader sends the request to the exporting trader. The address of the exporter can be found by using ODP identification common functions, such as the X.500 Directory Service [9]. (Note: The searching of the import contract could be restricted to those cases when the local trader fails to return a service to its client.)

4.4.2 Links to Export Contracts

When an exporting trader receives an F-Search request, it associates the search with the export contract identified by the requesting trader. The search is restricted to those directories of the exporter specified in the export contract. The resulting search list, which might be empty, is sent back to the importing trader. The importing/exporting traders do any mapping between the two systems according to the rules specified in the contracts.

4.4.3 Mapping of Trader Operations to Federated Operations

Many executions of local operations will not impact on the federation in any way. The following list outlines the circumstances in which a local operation will impact on the federation. However, only F-operations that are within the enterprise policies and have been agreed upon can take place.

Export If an Export operation nominates a directory accessible via an import contract, then it will be translated into an F-Export operation. If an Export operation increases the set of service types in a sub-tree associated with one or more export contracts, then the 'Held' field in the export contracts and the 'Available' field in the corresponding import contracts must be increased accordingly.

Withdraw If a Withdraw operation nominates a directory accessible via an import contract, then it will be translated into an F-Withdraw operation. If a Withdraw operation reduces the set of service types in a sub-tree associated with one or more export contracts, then the 'Held' field of those export contracts and the 'Available' field of the corresponding import contracts must be reduced accordingly.

Modify If a Modify operation nominates a directory accessible via an import contract, then it will be translated into an F-Modify operation.

List If a List operation nominates a directory accessible via an import contract, then it will be translated into an F-List operation.

Search If a Search operation nominates a directory from which one or more import contracts is reachable, then it can be translated into an F-Search operation for each import contract.

Select If a Select operation nominates a directory from which one or more import contracts is reachable, then it can be translated into an F-Select operation for each import contract.

Invoke If an Invoke operation nominates a directory from which one or more import contracts is reachable, then it can be translated into an *F-Select* operation for each import contract.

Add-Type If an Add-Type operation nominates a service type that is unavailable via one or more import/export contracts, then it will be translated into an F-Add-Type operation for each of the import and export contracts. This might lead to changes in the set of service types that a remote trader is willing to make visible through its export contracts and the set of service types it requires through its import contracts.

Delete-Type If a Delete-Type operation nominates a service type that is accessible via one or more import/export contracts, then it will be translated into an F-Delete-Type operation for each of those import and export contracts. This might lead to changes in the set of service types that a remote trader is willing to make visible through its export contracts and the set of service types it requires through its import contracts.

List-Type If a List-Type operation nominates a service type accessible via import/export contracts, than it will be translated into an F-List-Type operation for each of the import and export contracts.

Add-Directory If the Add-Directory operation nominates a directory accessible via an import contract, then it will be translated into an F-Add-Directory operation.

Delete-Directory If the Delete-Directory operation nominates a directory accessible via an import contract, then it will be translated into an F-Delete-Directory operation.

List-Directory If a List-Directory operation attempts to access directories accessible via one or more import contracts, then it will be translated into an F-List-Directory for each of these import contracts.

4.4.4 Fast Access to Contracts

If the export contracts contain the directory and export identifier [7] of its corresponding import contract, then listing and modifying the import contract can be more efficiently implemented. Similarly, if the import contract contains the directory and export identifier of its corresponding export contract, then listing the export contract can be more efficiently implemented.

4.5 Technology Viewpoint

Specific choices for representation of data can be made in this viewpoint. For example, ASN.1 Basic Encoding Rules could be used for the exchange of typed information.

4.5.1 Performance

To increase the performance of searching for an import from remote traders, 'import indexes' could be created by an importing trader to facilitate searching through the import contracts. An import index collects all the service type names that are currently available for import from the collection of import contracts of an importing trader. Each service type is associated with the location information of the corresponding export contract(s)

gathered from all the import contracts. When an importing trader wishes to import a service type from its federated traders, the importing trader will search through its import index to find the location of the export contract for the required service type. And, if there are more than one export contracts associated with the service type, then the importing trader can issue simultaneous F-Search or F-Select requests to the corresponding exporting traders. The index could also include mapping information between the local and the remote service type and the access location of the remote trader.

Thus, the index provides a quick mechanism to link a required service type with its exporting traders permitting parallel federated searches and/or selections. The index is kept current by being updated whenever new contracts are formed or old contracts are cancelled.

5 Conclusions

In this paper we have proposed a model of decentralized federation of traders. Trading partners form contracts that provide a transparent method of handling heterogeneity and an efficient method for searching. Each component trader manages its own federation by negotiating and maintaining its own import/export contracts. The model allows for complete autonomy of component traders. However, the individual maintenance of contracts does demand a high overhead.

Federated traders are an example of an ODP system and the study of federated traders provides insights in ODP itself. As a consequence of our study of federated traders, we make the following comments about ODP in general.

The distribution of trading information amongst several trading objects in a federation is not visible except in the engineering viewpoint [1]. In order to study federation we have used the five viewpoints of ODP. Any engineering mechanisms of the federation can be studied further using the five viewpoints again. It follows that in the general case within the ODP framework, it is necessary to recursively apply the five viewpoints to engineering mechanisms of a system.

We note that there is a need for third-party security mechanisms. When Trader X interacts with Trader Y on behalf of user Z, then Trader Y must be confident that Trader X is a true agent of user Z so that Trader Y can perform access control on the basis of User Z's privileges and restrictions. In our model, the use of chaining flags allows further indirect access of other traders. The security mechanisms must be effective at all levels of indirection.

We learn that transparency does not imply secrecy. Transparency means a user does not *need* to know. It does not mean that a user *must* not know. Sophisticated users could make effective use of non-transparent information if such non-transparency is permitted.

We have included operations for type management in our model in order to present a complete picture. However, we believe that such operations should not be the responsibility of the trading system but should be a separate common function of ODP.

Finally, our adventure confirms that while separation and location can be made transparent, autonomy and incremental change cannot be made completely transparent in an Open Distributed Processing environment.

Acknowledgements

The support of Telecom (Australia) Research Laboratories for this work is gratefully acknowledged by both authors. In addition, Mirion Bearman would like to acknowledge the Australian Centre for Unisys Software for its support.

References

[1] ISO/IEC JTC1/SC21/WG7 N310, "Working document on Topic 4.3 - Function and Interface Definitions", October 1990.

[2] ISO/IEC JTC1/SC21/WG7 N312, "Working document on the Trader", October 1990.

[3] ISO/IEC JTC1/SC21/WG7 N308, "Report on Topic 4.1 - Requirements for Structures and Functions", October 1990.

[4] ISO/IEC JTC1/SC21/WG7 N315, "Basic Reference Model of Open Distributed Processing - part 2; Descriptive Model", October 1990.

[5] A.P. Sheth and J.A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", ACM Computing Surveys, Vol 22, No 3, September 1990.

[6] R. Van der Linden and J. Sventek, "Trading in the Five Projections", ISA project, T24.1; Working Paper, July 1990.

[7] ISO/IEC JTC1/SC21/WG7 N328, "Proposed Working Document on the Trader", February 1991.

[8] ISO/IEC JTC1/SC21 N3801, "Support Environment for Open Distributed Processing", ECMA, August 1989.

[9] "The Directory", CCITT Recommendations of the X.500 series, December 1988.