Middleware Support for Open Distributed Applications[1]

W. Lamersdorf, M. Merz, K. Müller-Jones

[lamersd|merz|kmueller] @ informatik.uni-hamburg.de

University of Hamburg, Department of Computer Science
Vogt-Kölln-Straße 30, D-22527 Hamburg, Germany

Abstract

Todays global and high-speed communication networks provide an open system environment which allows to access a nearly unlimited multitude of services at continuously decreasing communication costs. Advanced communication networks thus provide a powerful basis for realizing, e.g., world-wide distributed "Electronic Markets" with increasing similarity to real (i.e. economic) open service markets. However, middleware support for such open distributed applications at an adequate level of abstraction (i.e. beyond elementary communication functions), has still to be designed and implemented. Therefore, this contribution first briefly reviews some typically characteristics of realistic open service markets and then presents an overview of specific system support for such open distributed applications as realized in the ongoing COSM and TRADE projects at Hamburg University.

1. Introduction

Todays global communication infrastructure allows to interconnect various types of networks and services at continously decreasing communication costs. Internet information service infrastructures such as the World Wide Web (WWW) owe their success to both stable and ubiquitous availability of communication end-points and simple infrastructure communication protocols.

System ("Middleware") support for distributed applications, however, has to specifically address problems of both the multitude and heterogeneity of accessible networks nodes, services and interfaces [1]. In the recent past, these problems have motivated support functions and standardization activities for, e.g., client/server "trading" or service "mediation" system components. *Trading* is the process of matching client service requests with corre-

---

[1] 1st intl. Workshop on High Speed Networkks and Open Distributed Platforms, St. Petersburg, Juni 95

sponding service offers which are accessible somewhere within a distributed open systems network. The specification of a trader function is currently carried out within the ISO "Open Distributed Processing" (ODP) standardization activities [2]. Respective system support functions for an open distributed "Trading and Coordination Environment" (TRADE) are designed, extended, and implemented in the corresponding research project at Hamburg University [3].

In general, client/server systems are considered to be *open* as long as they interoperate on the basis of standardized communication mechanisms. From an *application* point of view, however, they may still appear as *closed* systems whenever specific details of the respective cooperation partners have to be known in advance. Approaches to client/server cooperation in open distributed environments - as currently discussed in ODP address this problem by providing the necessary knowledge through dedicated "trading" functions; such cooperations are, however, limited to fairly *close* (i.e. well-informed about each other, predefined, or 'standardized') client/server relationships.

In more general scenarios of *dynamically changing*, unrestricted *global* communication infrastructures, however, various and heterogeneous kinds of service offers and requests may typically arise at *arbitrary* points in time and space in a distributed network system. Based on the assumptions of the traditional service "trading" scenario, this would - for any global trader or specific client application - mean to have dedicated knowledge of and agreement with the different service types available in the network. Of course, this is not feasible beyond the limitations of relatively small networks (resp. service) boundaries and hinder new, alternate, competing, or complementary service providers to enter such a distributed service environment - to the benefit of the clients - freely and with without utilization delay.

In order to overcome such mismatches between - on the one hand side - flexibility requirements of distributed system (client) users and - on the other - the necessity to closely (and correctly!) align suitable networks components to one another based on agreed service characteristics and cooperation protocols, a generic system infrastructure for a "Common Open Service Market" (COSM) is designed and stepwise realized in a respective project at Hamburg University [4, 5, 6]. COSM supports a *user-oriented* cooperation schema which allows clients to stay independent from the details of specific service providers. In this schema, the human user is directly involved in (yet unknown) service selection and application specific interaction. Based on a common (i.e. uniformly defined) *Service Interface Description (SID)*, the proposed open system support infrastructure allows to utilize service-specific middleware components like *Generic Clients*, *Service*

Browsers, etc. In the resulting system infrastructure, *any* appropriate service provider can be selected in a flexible manner based on human end-user interaction.

Therefore, the question jointly addressed by both the COSM and the TRADE projects respectively is how the two alternate approaches to matching client requests with appropriate service offers in large-scale open system environments can be best combined: the basic idea is to *integrate* both approaches to system support for - on the one hand side - flexible *human*-oriented, interactive service selection mechanisms (COSM), and - on the one hand side - automatic *trading*-oriented (TRADE) service selection into a common distributed open system infrastructure ("Middleware").

1.1. Electronic Markets

Information service *infrastructures*, also called *Middleware Platforms,* support interaction and cooperation between their potential user types - clients and servers. In addition, they may provide *auxiliary services* like connection support, billing, encryption, directory, authorisation, and authentication, etc.

Before examining existing information service infrastructures for their suitability to establish such an *Electronic Market System* (EMS) [7], some critical success factors for such a system shall be briefly reviewed [8]:

- *Search costs* as well as service utilization costs for clients should be reduced to a lowest possible scale,
- *Switching costs* which arise when, e.g., customers plan to examine other services than the currently accessed one should be minimized. High switching costs may restrict clients from evaluating these services or gathering new ones,
- *Adaptation costs* concerning both client and server should be minimized as well. Such set-up costs arise at the EMS infrastructure providers and comprise, e.g., fixed subscription fees, service offer fees for providers, software purchase and installation costs for clients or servers, etc.
- Adequate *communication standards* are vital for open systems communications; but the more standardization involves application-level aspects the more restrictions are imposed on service individuality. A distinction is therefore made between *representation standards* which rather prescribe syntactical aspects of data exchanged and *content standards* which address service semantics. Application level standardization should therefore focus rather on data representation acpects.
- *Dynamic emergence of value chains*: In a scenario of high numbers of on-line services offered there is a rising demand for transparency, selection support, and service

evaluation. EMS architectures should allow or even encourage the existence of value-added services (VASs) to customize, enrich, or combine existing ones and to flexibly close occuring gaps between supply and demand.

- *Decentralization level* of the EMS infrastructure: Despite physical decentralization, EMSs may be logically centralized in the sense that they are involved in each single market transaction as a central control unit. Auxiliary services (like directory, notary, clearing, and repository services) are then provided exclusively by the central infrastructure provider.
- *Security, and trust*: Both cooperating parties, client and server, establish a legal contract when, e.g., a reservation is made. Beyond fundamental security requirements, a *trusted* service is required to assure the identity and reliability of the respective contracting partners. This service may either be provided by the EMS infrastructure itself or by specific third-party services (acting as a 'notary' function).

Apart from transaction cost reductions, flexibility in mapping real world activities to an electronic market system appears as a fundamental factor for the success of EMSs. Current commercial infrastructures like CompuServe or the German BTX system lack this ability: *Every* transaction involves 3 parties: the client, the actual service provider, and the infrastructure provider. Current developments in research, however, rather emphasize peer-to-peer communications infrastructures where auxiliary services may only be *optionally* involved and selected from both contracting partners. In principle, these services can thus be provided by third parties and are therefore allowed to evolve more rapidly in order to respond faster to changing market demands.

The following chapter shows how the "COSM" research prototype system supports EMS applications in open distributed systems in the more general framework of the common COSM and TRADE projects at Hamburg University [3, 5, 6, 8].

2. The COSM Project: Assumptions and Architecture

2.1. The COSM Architecture

The COSM infrastructure allows service providers in open EMSs to offer newly created services without previous content standardization.* The specific advantage of the COSM architecture lies in increased *design* and *execution autonomy* of service providers: Service

* When, however, such innovative services have gained maturity and coherence among competing providers, an a-posteriori standardization based on a uniform service type may take place later.

providers are not obliged to adhere to existing service interface standards for their own operational interface.

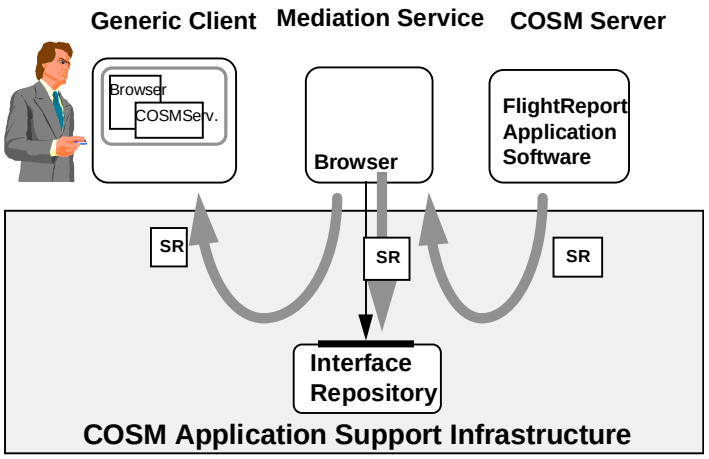

Fig. 1: Basic COSM components

The architectural model of COSM is based on the following main components (Fig. 1):

- A *Generic Client* (GC) component supports (client) users in service discovery, service access and binding, and in the inspection of service descriptions at run-time.
- A common *Service Representation* (SR) which contains several descriptional components (defining, e.g., the operational service interface, the GC user interface layout, the interrelation between user interface and remote procedure invocation, human-understandable descriptions of service functionality, billing information on the charge of procedure invocations, a definition of the legal order of invocation sequences, etc.).
- *Service Providers* implement dedicated functionalities and are accessible on-line in the COSM network. They supply their respective SRs to their potential users (possibly via GC or mediating services).
- SR *Browsers* and *Repositories* support users to inspect and store service descriptions.
- *Traders* identify appropriate service suppliers more efficiently by automating the search for best suitable services based on given service description criteria.
- Finally, *Mediation Services* provide application specific tasks like, e.g., collecting news articles for their customers.

When binding to a server, a GC receives the SR and generates the corresponding user interface automatically. The user can then inspect the information provided by the SR and familiarize with the described service functionality. A binding to the server can then be

established (or released if this information does not describe the kind of service the user was looking for). In COSM, the user interface representation of a remote services site is standardized at the GC, layout and content, however, may vary from service to service. This interface enables users to execute remote operation calls just by filling out forms and pressing corresponding buttons automatically.

Service references are data values which can be transferred across the network. On this basis, dedicated COSM services are enabled to act as directory services allowing servers to register their service representation and clients to obtain detailed information about the requested services. Such mediation services may act on behalf of their clients to evaluate the quality of services offered, or they may act on behalf of servers to "advertize" registered entries.

In a commercial environment, interaction with remote servers frequently also implies a *contract establishment* as well as a payment flow for service utilization. Therefore, current research also focusses on the integration of an anonymous electronic payment component [9]. Accordingly, servers may charge their clients either on a connection basis, by operation invocation, or by the amount of data transferred. Mediating services may thus charge either service registration operations (advertising services) or service reference look-up (directory services). Service look-up is free of charge for clients in the first case, server registration in the latter.

Despite of the fact that COSM servers are defined by their operational interface, they constitute a hybrid system consisting of the on-line accessible software part and human users who may take part in a servers response activity. Mediators which evaluate remote services therefore carry out their task by letting users access remote servers for testing purposes and making this information available for clients afterwards.

2.2. Application Execution in COSM

A general problem of distributed application design is to find a suitable distribution level between the top-most presentation level and a remote file access as the most bottom solution of an application. Both extreme solutions imply high network and process load. Therefore, the COSM approach is to shift the function split between the client and server part of a distributed application to an optimal level that reduces load, on the one hand side, and allows the Generic Client component to remain service-independent, on the other. Accordingly, to interact with a remote server, the user binds to it by receiving a COSM SR at run-time. Therefore, an SR is both a capability to access that server and a description of the service interface and semantics.

Beyond the problems addressed in this contribution, the COSM project focusses on the following problems: embedding trusted third party services into the infrastructure, like notary services, which assure a secure and confidential legal support of contract settlement for client and service components on an electronic market [6].

The COSM *Generic Client* (GC) is a tool for human users to acquire remote server access and to support interactions between users and servers. Communication between client and server is based on a Dynamic Invocation Interface (DII) as defined by the OMG CORBA [10] specification. The DII dynamic typing mechanism immediately results from ad-hoc binding requirements at the COSM-application level. Although late binding generally implies the possibility of a type mismatch at run-time, communication in COSM is type-safe since RPC invocation parameter values are generated by means of SR operation signature descriptions. RPC data types correspond with user interface types in order to allow users to inspect and modify parameter and result values at the GC site. Besides standard DII data types, COSM provides special types for, e.g., service references, bulk data values, and ASCII files. By supporting service references as first-class data objects COSM servers are able to display directories of servers via the Generic Client to the user - similar to WWW hypertext references. Binding to a such referred server is effected directly from the user interface by selecting the according service reference. Loosely coupled chains of servers may emerge from this principle.

In real (economic) markets, *value chains* supply abstraction, sophistication, and transparency to their respective consumers. End consumers are usually not involved in mining raw materials, manufacturing parts, assembly, or packaging of the goods they purchase. Accordingly, package tourists are not willing to be involved in time consuming service selection, evaluation, and combination processes - they preferably rely on the specialization of dedicated firms like travel agencies, tour organizers, or organizations like the "better business bureau". Transformed into the context of the COSM approach, an important infrastructure design goal is to facilitate the emergence of *value-adding Services* (VAS) which are built on the basis of pre-existing ones. The only precondition for a reliable use of COSM VASs is a stable interface of the underlying services over time: Servers are allowed to introduce additional operations, to accept less parameters, or reduce the number of result values, but syntax and semantics of previous interface versions must be preserved, to conform in the sense of interfaces subtyping rules.

A comparison of COSM with, e.g., the *World Wide Web* (WWW) reveals some few but substantial differences between COSM and the WWW:

1. COSM allows to interactively access *software applications* while WWW supports *document exchange*. However, interactive forms are supported by WWW but such

"operation calls" only result in the delivery of a further document. Restrictions of WWW apply when results of previous operation calls are used as parameters for further ones (statefull servers).

2. The second difference concerns possibilities for *chaining* service providers to allow value chains to emerge. Since COSM servers present an operational interface, they allow for specific client software that *automatically* executes operation calls. This software does not provide a user interface but may act itself as a COSM server. In addition, in order to let client calls directly pass-through to the back-end server, such front-end servers may carry out specific computations on parameter and result values.

2.3. COSM Service Representation

The crucial data object to contain and transfer arbitrary service descriptions in COSM is the Service Representation (SR). At its most generic level, the SR is a container for data structures of arbitrary types at run-time. An SR-interpreter (as a component of the Generic Client) seeks descriptional components it is sensitive for after receiving the SR. Among these components are:

- a specification of *operation descriptions*, containing operation names and *parameter descriptors*. parameter descriptors refer to *data object* which contain actual values. These components roughly resemble interface definition clauses known from DCE or CORBA IDLs.
- A specification of the *user interface* to be generated for human users by the GC. It contains specifications for dialog boxes, data editors and push buttons. Data editors manipulate data objects,
- a specification of the service interface protocol, i.e. which operations are enabled to be invoked at a given state. Currently, this protocol description is based on a *finite state machine* (FSM) model and comprises a set of application states and transitions between them which refer to operation descriptions. State changes are effected by user-level events and execute, in turn, RPC invocations,
- informal description components, e.g., help texts to support human users,
- cost functions that inform users on service access and related operation invocation fees,
- any data values locally used by the GC.

Since the Generic Client software component is application-unspecific, state information like, e.g., window positions or counter variables are captured by the service representation. Therefore it is possible to store the SR persistently and to suspend interactions with the

remote server. This SR can be re-activated later - or from another network site - in order to resume the previous communication state.

Two benefits arise from an SR-based approach to implement open service access: first, *representations* are standardized within COSM, not interfaces. This includes a uniform, service-independent user interface presentation and allows type-safe data entry. Second, more application-specific functionality can be shifted to the client site. In fact, COSM servers are reduced to plain call libraries with corresponding SRs. Compared with, e.g., X-Windows, dialog control is shifted to the user part of a distributed application.

The Generic Client dialog management component connects presentation functions to the application call interface. Users are allowed to invoke remote operations by pressing corresponding buttons. In the case of state-prone services, buttons are disabled if the operation is not allowed to be invoked in the current application state.

As containers for arbitrary data type definitions and value components, SRs are extensible at run-time. This allows specialized environments to introduce dedicated component types for their own purposes. COSM applications that are not aware of these extensions will still be allowed to interpret the subset of SR components they are specialized on.

3. Service Access Support in TRADE

Besides support for an (electronic) open service market - as provided by the COSM infra-structure components - there is an additional need for system support of client/server interactions in which the characteristics of the services accessed are *well-known* in advance. Based on such a-priori knowledge, server selection, access and binding can, in such cases, be executed *automatically* by specific dedicated distributed middleware components.

3.1. Service Trading in ODP

In general, client support for locating, accessing, and using arbitrary services in open system environments emerges as one of the most interesting, complex, and practically relevant tasks of realizing realistic open distributed systems applications. Therefore, current standardization efforts for a trading function play an increasingly important role for open system integration in the context of ODP. In addition to ongoing ODP standardization activities at a specification level, however, (official and de-facto) standard system support platforms - such as the OSF Distributed Computing Environment (DCE) [11] - have emerged in order to support efficient development and portability of distributed system applications. Therefore, time seems now ready to analyse and evaluate

the use of such platforms also for developing efficient implementations of, e.g., an ODP trading function.

Based on rapid recent developments of telecommunication and networking technologies, users of distributed systems are confronted with multitudes and varieties of service offers in a principally world-wide open market of services. Faced with the complexity of such open distributed environments, one of the main tasks is to support users and application programs to locate and utilize such services in effective and efficient manners. In this context, one of the most promising efforts is to extend open (operating) system platforms by unified service trading or broking components as an important structuring technique for efficient design of open distributed systems.

Accordingly, the specification of a trading component is currently - among other issues - subject of the Open Distributed Processing (ODP) international standardization activities as a so-called ODP trading function. 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 to arrange and categorize various service types, and then supports potential clients with specific service selection strategies.

Thus, the functionality of a trader component 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 types which specify the operational service interfaces in terms of operation signatures as well as *service property types* 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 can be grouped and located (e.g. in a hierarchically organized name space).

According to the ODP trader function, the service provider, a service *exporter*, first registers its service by supplying the service type, the current service property values, and a context in which the service shall be exported. Then a service client, the *importer*, may ask for servers offering a specific service. Such an inquiry contains - among others - 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, and the client is finally able to directly execute remote operations offered by the service provider.

3.2. A TRADEr Implementation based on DCE

Beyond the ongoing standardization efforts for a unified trader component specification, it is, however, increasingly important to also examine possible *implementations* of such trading concepts based on distributed middleware functions. Such common foundations for various distributed application implementations are existing and evolving distributed systems architectures, as, e.g., proposed and developed by various vendors and consortia like OSF's DCE, OMG's CORBA, Sun (ONC+), and APM's ANSAware.

In particular, the integration of a trading component into these architectures has to be evaluated: A good starting point for such an evaluation is the *Distributed Computing Environment* (DCE) from the *Open Software Foundation* (OSF) which has gained wide commercial acceptance and is one of the important system platforms for distributed application development in the future.

Accordingly, the TRADE project is concerned with

- an analysis of DCE mechanisms to support service management and service access,
- the limitations of DCE for supporting service mediation,
- a proposal for an architecture of the *TRADEr*, a DCE trading component,
- details of a smooth integration of the TRADEr implementation into DCE,
- extensions of service type abstractions and their integration into the TRADEr,
- support for client/server mediation in a distributed open systems environment, and
- design and implementation of 'interworking' trader functions and protocols.

OSF DCE provides an integrated set of support services and application programming interfaces for the development of distributed applications in open heterogeneous environments. The main goal of DCE is to provide users and applications with a homogeneous view of distributed computing environment which hides much of the complexity 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.

One of the most important components of a composite trader function is the *type manager*. 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. Here, different notions of service types serve as - more or less - formal abstractions of service characteristics, i.e. common properties of classes of service instances of a distinct service type. Because of the significance of the service type concept, standard typing mechanisms, as known from modern programming languages, and -

increasingly important in the future - further extensions to such service type concepts have to be evaluated for the trader's type manager component.

A final extension to service (type) descriptions - as considered in the COSM/TRADE projects so far - is concerned with *coordination* of complex distributed services which comprise a set of more basic ones. It uses *Coloured Petri Nets* (CPN) as a formal description technique of such coordination problems and is currently evaluated for supporting workflow modeling and execution within complex open distributed client/server environments [12]. Here, CPNs are used for describing service coordination in order to support the execution of workflows in open service environment in an adequate manner. One approach currently evaluated in this context is to extend each service (instance) description with a CPN representation of its respective usage in a global workflow application.

4. Status and Outlook

4.1. COSM

The COSM project aims at improved system support for flexible client/server integration in distributed and heterogeneous open systems. An important goal is to support not only *specific, predefined* client/server cooperations but rather to design a *generic* architecture for flexible service management, access, and coordination in open systems. The COSM architecture and software infrastructure help to address and solve open distributed system design and implementation problems at the *infrastructure* level by providing flexible access and management of arbitrary remote services in general.

The COSM system architecture supports arbitrary client/server interactions in open distributed environments. The current *COSM/TRADE prototype system* has been developed on a cluster of RS/6000 AIX workstations. It basically consists of four different component types (Fig. 2):

- an extended *Dynamic Invocation Interface* (DII) which was built upon the Sun RPC and XDR interface: It allows to allocate lists of structured parameter objects and to reconstruct them at the receivers site after transmission.
- a *Service Representation Manager* which controls access to the binary SR data structure: The SR is organized as a contiguous memory allocation unit with local memory management.
- a Generic Client component which was built upon the previous system functions: It consists of a presentation layer, a dialog control unit and the invocation manager. In

this order, a GC displays GUI information, handles user input events and invokes remote procedure calls based on information extracted by the service representation manager.

- various example *Server Applications* for testing purposes: Several demo servers have been developed, for example an SR repository that allows servers to register their SR on the one hand, and Generic Client to access the repository database, on the other. Registered SRs are interpreted by the repository and stored among others in the database. Generic Client users, who are bound to the repository server may run queries against the database to obtain structured information on the SRs registered.

First practical experiences have been made with the COSM/TRADE infrastructure with a decentralized implementation of a collection of distributed services. It comprises system support tools like the Generic Client as well as Browsers, Repositories, and Trading and Mediation services (like an ODP-conform trader). As a consequence, the reduced effort to make existing program libraries or 'legacy systems' available for external access decreases set-up costs for service providers. The resulting lack of a centralized cost-intensive infra-structure nexus is compensated by reduced transaction costs and the possibility to involve auxiliary services like, e.g., notary services if both customer and supplier agree. Current COSM system infrastructure extensions also aim at supporting *coordination* functions of concurrent and distributed application services.

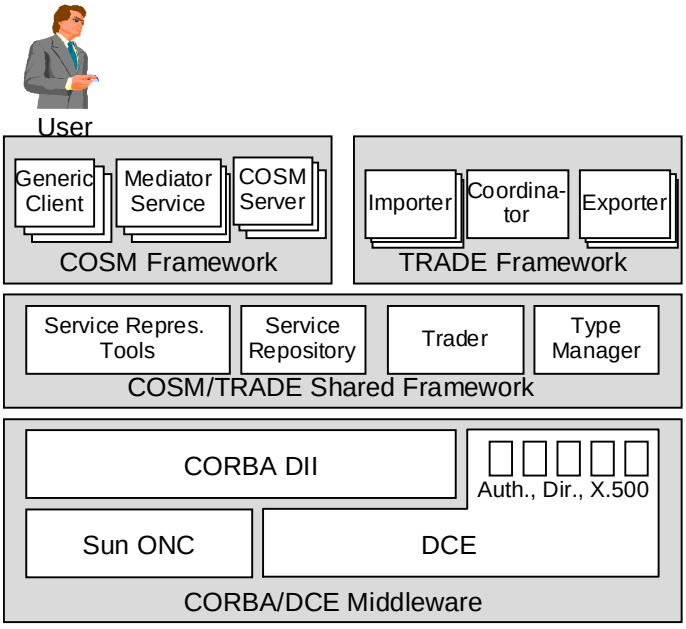

Fig. 2: Integration of the COSM/TRADE frameworks with CORBA/DCE Middleware

4.2. TRADE

The ongoing prototype work on the *TRADE project* is based on the importance of early evaluations of implementations of trading concepts in the context of existing distributed systems platforms as, for example, the OSF DCE. A concrete implementation goal is an orthogonal and smooth integration of basic trader functions into already available service registration and management mechanisms. The current *TRADE* prototype, implements extended ODP trader functions as part of a general system support environment for service access and management in open distributed systems. The prototype system is also realized within a heterogeneous cluster of interconnected SUN SPARC and IBM RS/6000 workstations; the TRADEr implementation has been developed on IBM RS/6000 workstations with AIX and DCE within a common DCE cell.

Related experiences from earlier TRADEr prototype implementations as available so far are twofold: On the one hand side, DCE already provides powerful functions for distributed application programming, many of which could also be used beneficially for the TRADEr's trading function implementations. On the other hand, however, DCE concepts already available for service (type) description are still unsatisfactory and corresponding DCE functions still lack important (especially: type management) mechanisms which are necessary for service management and mediation in open distributed environments. Therefore, extended type management functions have to be developed separately, based on modern programming language (polymorphic type) concepts as well as specific service description extensions (as, e.g., for protocol and workflow management specifications) as needed in open distributed environments.

In summary, experiences with the TRADEr prototype implementation have shown that trading functions can be smoothly integrated into DCE, but DCE components have to be extended substantially in order to support and use, in particular, realistic service descriptions based on modern type management functions. Extensions currently addressed in the TRADE project are: more elaborate type management functions, support for distributed trader interworking (currently based on X.500), and support for inclusion of dynamic attributes into extended open service descriptions. In particular, techniques to extend service descriptions with additional semantic information are still evaluated and will be integrated stepwise into the COSM/TRADE prototype implementation. In this context, early experiences with service type extensions like finite state machine description of service protocols have motivated recent work on using of more powerful description methods for coordinating composite services (*workflow management*) based on Coloured Petri Nets [12].

References

[1] V. Tschammer, A. Wolisz, J. Hall: "Support for Cooperation and Coherence in an Open Service Environment", Proc. Second IEEE Workshop on Future Trends of Distributed Computing Systems, Cairo, Egypt, IEEE Computer Society Press, Los Alamitos, CA, USA, 1990, pp. 222-228

[2] ISO/IEC JTC1 SC21 WG7: "Basic Reference Model of Open Distributed Processing", Working Documents No.s N7053 ('RM ODP') and N7047 ('Trading'), 1993

[3] K. Müller-Jones, M. Merz, W. Lamersdorf : "The TRADEr: Integrating Trading Into DCE", in: J. de Meer/ B. Reynolds/ J. Slonim (Eds.): Proc. IFIP 'International Conference on Open Distributed Processing' (ICODP'95), Brisbane, Australia, Chapman Hall, 1995

[4] M. Merz, W. Lamersdorf: Cooperation Support for an Open Service Market, eingereicht für: International Conf. on 'Open Distributed Processing', Berlin, 1993

[5] M. Merz, K. Müller, W. Lamersdorf: "Service Trading and Mediation in Distributed Computing Systems", Proc. IEEE International Conference on Distributed Computing Systems (ICDCS), L. Svobodova (Ed.), IEEE Computer Society Press, 1994, pp. 450-457, Los Alamitos 1994, pp. 450-457

[6] M. Merz, K. Müller, W. Lamersdorf: "Trusted Third Party Services in COSM", in "EM - Electronic Markets", No. 12, September '94, Volume 4

[7] T.W. Malone, J. Yates, R.I. Benjamin: "Electronic Markets and Electronic Hierarchies: Effects of Information Technology and Market Structures on Corporate Strategies", Communications of the ACM, (30:6), June 1987, pp. 484-497

[8] M. Merz, K. Müller, W. Lamersdorf : "Electronic Market Support for the Tourism Industry: Requirements and Architectures", in: W. Schertler, B. Schmid/A M. Tjoa/ H. Werthner (Eds.): Proc. Int. Konf. 'Information and Communications Technologies in Tourism' (ENTER95), Innsbruck , Österreich, Springer-Verlag, Wien New York, 1995, pp. 220-229

[9] G. Medvinsky, B. C. Neuman: Electronic Currency for the Internet, in: Electronic Markets, October 93, pp 23-24

[10] OMG: "The Object Request Broker: Architecture and Specification": OMG Document No. 91. 12. 1, Object Management Group, Framingham, MA, USA, 1991

[11] Open Software Foundation: "Introduction to OSF DCE" and "OSF DCE Application Development Reference", Prentice Hall, Englewood Cliffs, New Jersey, 1992/3

[12] M. Merz, D. Moldt, K. Müller, W. Lamersdorf: "Workflow Modelling and Execution with Coloured Petri Nets in COSM", in: Workshop on Appl. of Petri Nets to Protocols, Proc. 16th Intern. Conference on Application and Theory of Petri Nets1995