Distributed Systems Technology for Electronic Commerce Applications

Winfried Lamersdorf, Michael Merz, Tuan Tu

Distributed Systems Group
Department of Computer Science, Hamburg University, Germany
http://vsys-www informatik.uni-hamburg.de

Abstract. Based on the specific characteristics of electronic commerce (E-Commerce) requirements for an adequate system support, this contribution gives an overview of the respective distributed systems technology which is (or will be shortly) available for open and heterogeneous electronic commerce applications. Starting from basic communication mechanisms this includes (transactionally secure) remote procedure call and database access mechanisms, service trading and brokerage functions as well as security aspects including such as notary and non-repudiation functions. Further important elements of a system infrastructure for E-Commerce applications are: common middleware infrastructures, componentware techniques, distributed and mobile agent technologies etc. Increasingly new and important topics in this area are currently: workflow management support for compound and distributed E-Commerce services as well as negotiation protocols to support both the settlement and the fulfillment of electronic contracts in E-Commerce applications. In addition to an overview of the state of the art of the respective technology, the paper also presents briefly some aspects of related projects conducted by the authors jointly with international partners (sponsored by EU/ACTS, EU/ESPRIT, DFG) in order to realize some of the important new functions of a systems infrastructure for open distributed E-Commerce applications.

Keywords: Distributed Systems, Electronic Commerce, Electronic Contracting, Middleware, Workflow Management, Service Trading/Brokerage

1 Introduction

Electronic Commerce (E-Commerce) is frequently considered as *the* most important application area of open worldwide computer network infrastructures such as the Internet. Already existing global computer networks provide access to a nearly unbounded number of different functions and services. System support for such a complex distributed application area like E-Commerce comprises both an increasingly wide variety of functions and techniques already known from traditional communication, information, and cooperation support systems - as well as more specific functions in order to support, for example, service selection, trading, contract negotiation, security, and payment activities (to name just a few) [12, 9]. It has to fulfill strong flexibility and interoperability requirements on its respective software components – both at the system support as well as at the application level. It also reflects changing requirements and preferences of globally distributed, heterogeneous, cooperative organization structures.

1.1 Centralized vs. Decentralized Architectures

For example, many computer system customers demand today a re-centralization of enterprise computing systems in order to reduce roll-out efforts and maintenance costs. This development can be considered as a sober response to the idea of distributing any service at any time on any kind of heterogeneous hardware and operating system environment. Today, certain "low-level" services are accepted as inherently distributed such as DNS, NFS, HTTP, SMTP, etc. On the other hand, there are many others that were expected to be distributed applications in the early 90ies and before: distributed databases, shared editing, application-level extensions to the telecommunication infrastructure. However, they did not succeed in day-to-day practice by now. Why that?

One may argue that the first services are historically better understood since they have been developed for over 15 years, but there is another reason for the lacking success of the latter: the complexity of their respective specifications. A distributed database is not only itself a complex application but also the data models on top require immense conceptual and technological integration effort. If there exists a centralized alternative, that could be followed without facing prohibitive costs - this option would usually be chosen.

For a couple of years, however, the Internet established a drastically decentralized communication platform that flattens hierarchies, overcomes organizational borders and liberates small companies and individuals from high investment costs for communication services. This development stimulates the cooperation between business partners.

Finally, it is trivial to state that organizations are distributed across cities, regions, and countries. Since they coordinate the exchange of goods, services, and payments across their organizational boundaries and therefore across long distances, we find a natural need for a correlated support of electronic commerce applications by distributed systems. In this area, the distribution of applications is not only an option but a precondition for commercial life.

In the following, we first analyze the application fields for electronic commerce systems a bit more systematically and then provide some examples for components of an underlying system support technology.

1.2 Modeling Commercial Transactions

To provide a systematic classification of electronic commerce technology, two viewpoints could be taken: First, a distinction between businesses, consumers, and public authorities could be chosen for the respective roles of buyers and sellers of goods and services [6]. However, if we consider single persons as legal entities thus representing an equal market participant like a company, the borders between these categories will blur. For this reason we will follow the phase model for commercial transactions [19].

- In the first *information phase*, market participants offer product specifications, look for possible transaction partners, compare product specifications and prices, and evaluate offers.
- Then, after an initial contact has been established between some market participants, respective (service) offers and counter-offers are exchanged during the so called contract *negotiation phase*. This negotiation process may either lead to a situation where agreed terms and conditions have been reached or the negotiation is abandoned.
- In case of a contract establishment, first all participants commit their participation in the contract with their respective signatures, then the agreed assets are exchanged during the contract *performance (or: execution) phase*. The time-span of this phase may reach from a few seconds up to several years.

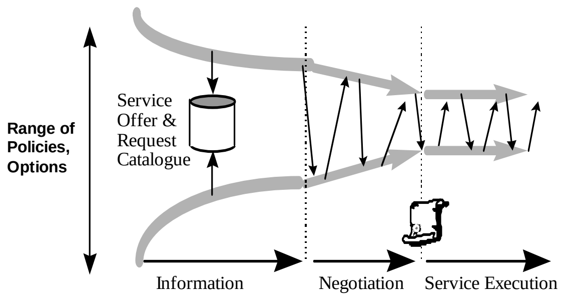

Fig. 1. Business Transactions' Phases

Following these phases, a clear separation of services can be given that are required for an electronic marketplace:

1. *Information phase*: This phase may be supported by (computer) functions like online catalogues, search engines, banner advertising
2. *Negotiation phase*: Here support for telecollaboration, negotiation protocols and strategies may be required.
3. *Execution phase*: During this phase, workflow management, business process integration among market participants, electronic payment systems, EDI-based message exchange functions, etc. may be provided in order to support the automatic execution of E-Commerce applications.

In between these phases, the following additional services may be required:

1. *Brokerage support* in order to select and match respective offers and inquiries, to form a (service providing) consortium or to set-up the negotiation session for all parties of the commercial transaction
2. *Signing support* to enter the execution phase by establishing a contract and ensuring for all parties to sign it. This process may be supported by trusted third parties such as certification authorities or electronic notaries.

In order to keep the model simple, yet without unrealistic abstraction, we consider any good as a service:

- A *payment* is a service provided by the customer. The result of the service is the transfer of data which is interpretable as a transfer of a value. This might either be an electronic coin or the settlement authorization between two bank accounts.
- A *tangible good* can be represented in the system as a service: It is selected, ordered and paid electronically and even the physical delivery is accompanied by a range of services and data communications that may be used by the commercial parties (transfers of electronic EDI documents, access to information on the delivery state, etc.).

1.3 Organization of the Rest of the Paper

The rest of this paper is organized as follows: In Section 2, we provide an overview of current system technology components for the respective phases as defined above. Afterwards, a system support reference architecture as developed in the EU/ESPRIT project COSMOS is presented as an example of a possible integration of these phases and technologies under a unified electronic contracting model. A perspective on future trends in system support for E-Commerce applications as well as an outlook of the COSMOS project is finally provided in Section 4.

2 Distributed Systems Support for Commercial Transactions

By following the steps of the phase model, requirements and solutions for distributed systems applications are discussed for the identified electronic commerce areas:

2.1 Catalogue services

To inform customers about the range of product and their specifications, catalogue services are used as a shared information system for both vendors and customers. A catalogue service also establishes a comfortable front-end for the following transaction phases, payment and product delivery. Internally, catalogues are supplied by the vendor's stock management system to keep the information displayed in synch with the physical warehouse.

Today, a catalogue service is deployed by a single vendor who intends to make accessible his range of tangible or intangible (‘soft’) goods to the customer. For small vendors, a catalogue may be hosted by a third party in the same way as the web server is provided by an Internet Service Provider. In this case, an individual vendor remains responsible for his own ‘shop’ in terms of presentation and product data. A unified settlement of payments and possibly the delivery of soft goods, however, can be centralized by the shopping mall provider.

As a next step, providers of catalogue services tend to further break down the effort for offering goods in a mall system by allowing vendors to enter single offers into the catalogue: this leads to an offer database, that allows competing market participants to register their offers in a suitable category of the offer database.

This concept has already been addressed several years ago by the ODP trader service [11], that not only suits for storing (*exporting* in ODP lingo) product specifications but also service specifications in a formal sense: services are understood as instances of a *service type* that includes *service attributes* and the *interface type*. A trader additionally provides exporting but also the matching of offers and inquiries. Later-on, this technology has been incorporated into CORBA standardization as the trader Object Service [16].

2.2 Service Brokerage

Usually, a catalogue access ends up in a purchase, which doesn’t require any further refinements of a contract or other terms. Usually the good is purchased at the offered price. However, if not only human users are involved, a formalized matchmaking service can be utilized to bring together customers and suppliers or even a group of market participants. In this case a brokerage service is used. To accomplish its task, a broker requires formal specifications of the services offered and demanded. Again, the ODP trader suits well for this activity: having already service specifications for different offer categories at hand, an *importing* client is only required to specify the required service by using an OQL (Object Query Language) statement.

The trader may also be applied to more than two participants in a commercial transaction by specifying a set of required services which are obtained step-by-step from the offer database.

Brokerage services are incorporated into different electronic commerce applications today. To name just a few examples:

- First of all, globally distributed trading/brokerage functions can generally be used for selecting the (according to pre-defined criteria) “best possible“ services in an open distributed service environment (such as, e.g., the Internet) [14].
- In the specific context of an open digital library (as an important, dedicated example of an open service market), e.g., the MEDOC project prototype uses a brokerage function for the matching of literature offered and demanded by users of electronic library systems [1].
- Another ‘Service Broker’ has been developed as a part of the EU/ACTS research project OSM (Open Service Model) [15] in the context of service selection for E-Commerce applications. The respective broker architecture is here based on the

OMG CORBA standardization of a Business Object Component Architecture (BOCA) which aims at tying together service offers that have been entered into a common catalogue [17].

2.3 Service Negotiation

Negotiation is the process of reaching an agreement for a service specification. This may take place either out-of-band, by letting market participants negotiate without electronic means, or it may be done on-line. In this case there are several stages of automation possible:

- *using collaboration tools*. In this case, human users are involved in the negotiation process. They use, e.g., a shared-editing tool that allows them to concurrently edit a document in a consistent way. The negotiation is free-form, i.e., there are no restrictions for the order of document accesses or the structure of the document.
- *Using negotiation protocols*. In this case, either human users or software components participate in the negotiation. The negotiation subject is still unstructured, i.e., the participants 'know' how to deal with it. The ordering of document accesses however is formalized and parameterized, .i.e., a negotiation protocol is applied to specify which party delivers which information at which stage to whom. The negotiation can be understood as a workflow process that is driven by a predefined process description.
- Using *formalized conversations* to further structure the negotiation protocol. Speech act theory (Specifically, the Knowledge Query and Manipulation Language, KQML [2]) provides a linguistic means to define formalized messages that relate - in the case of negotiations - to concepts such as 'offer', 'reject', 'propose', 'accept', etc. This further helps to tailor the involved software systems for the specific application of negotiation support: it may, e.g., react in a different way when it receives an offer instead of a proposal.
- Finally, *the complete negotiation process may be automated* (and therefore delegated to 'autonomous' software components) if the ontology for the negotiation subject has been standardized as well. In this case 'speed' and 'price' are features that a software component is able to reason about. Therefore, AI technologies are applied in this area for knowledge representation and for applying policies that have been defined to control negotiation strategies. Such an intelligent software agent is now capable at least to estimate 'price' and 'speed' and to trade-off their values in a reasonable way [21].

In today's real world, automated negotiation is not used so far for the following reason: Only if the service specification is kept simple (i.e., only a few Quality of Service attributes), a strategy module can be practically used for negotiating them. The more complex the specification becomes, the more effort needs to be spent for implementing policies and strategies for the agent. If a simple specification is sufficient, the service can be considered as a commodity on the other hand, i.e., a good that is offered by a large number of vendors on the market and for which an individual negotiation would come at prohibitively high costs.

Therefore, the practical integration of negotiation mechanisms won’t be feasible unless negotiation support is designed as an integral part of the overall software system.

2.4 Service Configuration

Services offered in an open and heterogeneous environment such as the Internet can only be competitive if they are flexible enough to adapt to a wide variety of user as well as technical requirements. Therefore, appropriate techniques are needed to (re-) configure an offered service dynamically according to the effective requirements in each case. For example, regarding the involvement of so-called “third party“ services – such as payment and notary functions – during a business-to-business transaction, as many options as possible should be supported and moreover, an option common to all transaction parties has to be determined and activated.

A very generic approach to provide system support for such kind of dynamic service configuration consists in using *policy management* mechanisms, which provide a formalization of arbitrary requirements in terms of policies which can be evaluated, compared, matched (or unified) and activated in an application independent way [20]. Policies can be added and activated fully automatically at run-time without changing the application code. The configuration effect is achieved by modifying externally accessible properties which are used as system parameters by the applications.

In a broader sense, flexible service configuration also means that arbitrary services should be so configurable that they can be easily plugged together to yield new functionality, i.e. that they can be used as building blocks to assemble new services “on the fly“. Providing technical support to fulfill this kind of requirements is precisely the objective of componentware techniques [8] which seem to be the right mean to face the growing challenges in the filed of E-Commerce, especially concerning the requirement of dynamic adaptability. However, there are still a lot of open questions and unsolved problems – for example, the *composition* of an application system out of components has to be distinguished from the *generation* of a new component out of existing ones since they have to fulfill, among other things, very different performance requirements – which are currently being investigated in projects such as DynamiCS [5] in order to make componentware technology applicable in practice.

2.5 Electronic Contract Signing

From the legal perspective, contracts don’t need to be signed. They become valid even if they are closed orally or by a concluding action. I.e., when a customer hits the ‘Buy’ button of an electronic shop application, it can be assumed that the consequences are well-known.

On the other hand, there are several reasons that promote the idea of involving an electronic contract into online transactions:

- *A written contract cannot be repudiated.* In the case of an electronic contract, this can be signed by the parties as well as by a *trusted third party*. This states who agreed on which terms and at which time. Any arbitration that may be required among these parties can be settled better if there is a version of the contract available that has been archived by a neutral auditor.

- The *legal framework* for online commercial transactions is being established in several countries now. Electronic signatures are at least accepted as an authentication means for the document signed. However, the management of a contract still requires a further harmonization of the national legislation for the participating countries.

  Complex legal situations can be better fixed by using a document as the common form of agreement. It is best practice today that commercial vendors display their terms and conditions as a part of their online presentation. However, it would clarify the legal situation if these documents are not displayed transitionally on the Internet but if they could be escrowed and archived at a third party (e.g. the Chamber of Commerce). This would allow the contracting parties to refer to this document even a long time after it has been replaced by a new one.

  Furthermore, some contracts may be negotiated and closed that require complex specifications such that it is essential to handle them in written form as a shared document. This applies to work plans as well as to complex relationships for obligations and right within consortia.
- In contrast to paper-based contracts, their electronic counterpart are *executable.* Structurally, such contracts incorporate clauses that determine the obligations and rights of each party. From the technical point of view, this can be interpreted for many contracts as an activity or a service that is to be provided at a certain time (payments, delivery of a good or a report, translating a document, or printing, binding and delivering books). Therefore, the *execution phase* of the commercial transaction is not only interpreted in the legal sense as the execution of a contract, but specifically in a technical sense by invoking the corresponding service through remote method invocations.

2.6 Electronic Contract Execution

This final transaction phase allows again to be considered from different angles.

- At least, the *legal execution* can be monitored at the human level - as it is the case in 'classical' commerce.
- Since deadlines and durations can be represented electronically, this information may be transferred to a workflow systems that sends notifications to the parties involved. These notifications refer to the actions the parties agreed upon, e.g., initiating a payment or performing an action. This can be called a 'supportive' workflow system.
- At the most sophisticated level, these actions may even be triggered by a workflow engine that performs method invocations at the different information systems which the parties made available for the others. In this case, a distributed computing infrastructure is assumed that easily enables market participants to be represented not

only through Web servers but also through distributed object-oriented software components [13]. Moreover, these components needs to be configurable at run-time in order to integrate them as a part of the commercial transactions

This final situation is only possible if a global network of related objects exists and if these objects can be inspected, refined, combined and integrated as dynamically as a contract is dynamically set-up and executed. A specific requirement is for this reason to transfer the workflow specification that all transaction parties agreed upon into the process definition that is required for a given workflow engine.

Therefore, this approach will only be successful if the workflow mechanism is not isolated from the previous mechanisms for negotiation and signing. For this reason the COSMOS reference architecture has been developed:

3 Case Study: the COSMOS Project

To accommodate the different functions mentioned above, a unified architecture is required in terms of an integrated object model and a functional specification of basic software components. Exactly this is the goal of COSMOS (Common Open Service Market fOr SMEs), a European ESPRIT research project that designs and implements the technical foundations for carrying out business transaction across the Internet [4]. In the following, this project is presented in some detail as a case study to demonstrate the use of some typical functions of distributed systems technology mentioned above.

Compared with existing electronic commerce architectures such as CommerceNet eCo [3], TINA [22] and the OMG Electronic Commerce Reference Architecture [15], COSMOS is tailored around the concept of a *contract*, which is not only used as a metaphor but as a tangible part of the complete process of a commercial transaction. The COSMOS architecture mainly focuses on software design aspects and less on organizational questions. For the latter we refer to the COSMOS white paper [4].

3.1 COSMOS Contract Model

As the most relevant part of the COSMOS object model, we focus on the modeling of contracts since they serve as the common nexus for most of the transaction phases and building blocks of the implementation.

A COSMOS contract could be considered as structured document composed out of text blocks. In this case, the editing process would be simplified, however, the automated processing of a contract will be very limited. On the other hand, one could attempt to cover the full semantics of a contract by building a ‘contracting expert system’. We consider this as a dead end since the expert system overhead is expected as too high – particularly for a Small and Medium Enterprises (SME)/Internet context, characterized by a permanent change of rules, roles, and business subjects.

As a trade-off, the COSMOS contract model aims to identify only those semantically meaningful parts of contract instances which allow for efficient automation and therefore highest increase of the added value. The parts of the contract model can be distinguished by their subject:

- The ‘Who’ part: Parties, Persons, and Signatures are related to the participants of the contract. Parties act under a certain role defined by the contract template. They are instantiated as a legal entity which can be in turn a person or an organization. The first may, the latter must be represented by proxies. “Party“ only indicates that the legal entity is involved in the contract and abstracts away from the actual tasks which are defined for the corresponding role. Finally each legal entity is associated with a signature when the contract has been closed.
- The ‘What’ part is the subject of the contract. It covers all obligation of the involved parties. Each obligation is considered as a transfer of a right which can be either a good, a service, money, or a license. An important feature of the obligation is a list of QoS attributes. It is used for contract templates to specify suitable parties. During contract negotiation, these QoS attributes are subjects of offers and counter-offers. Finally, obligations are to be carried out in the basis of these attributes during contract execution.
- The ‘How’ part defines relationships between obligations: when are which services to be delivered? What is the deadline? Which clause will apply when a party falls behind its obligation? The “How“ part is used to derive a workflow that defines causal relationships, data transfers, delays and deadlines, and the final termination of the execution phase.
- Finally, some common clauses form the fourth part of a contract. These clauses address general terms and conditions at the level of the contract. Also references to applicable external contracts, regulations, and legislation are placed in this part.

Apart from the structural perspective, a contract goes through several steps in line with the transaction phases:

- Initially, a *contract template* is defined, which usually predefines the ‘How’ and the ‘base clauses’ parts. Additionally, roles are defined and for each obligation a requested set of conditions. However, the template does not yet identify the contract parties nor the exact obligations: instead of attribute/value pairs (such as ‘price per acre = $100’ or ‘ground’s humidity = 20%’), constraint expressions are used as QoS specifications (such as ‘price < $150’ and ‘humidity < 30%’).
- *Contract proposal*: By using the broker, the template will be completed if suitable providers can be retrieved from the catalogue. The broker’s task is to replace QoS specifications with the corresponding values offered. For each category of obligations a corresponding offer category is required for the catalog. Accordingly, the party objects of a contract template are replaced by the respective participant description taken from the catalog. If the brokerage step leads to a completed contract that can be signed in principle, a *contract proposal* is given.
- During negotiation, contract proposals are exchanged between the parties. Depending on the semantics of such a contract transfer, it may either be considered as a *proposal* (without legal binding) or as an *offer* (with legal binding if the other parties accept). If all parties accept, the contract is in an *agreed* state and ready for signing.
- After all parties (or their proxies) signed the contract, the electronic contracting Service certifies this. Afterwards, the contract is *executable*, i.e. in technical terms,

it can be transferred to the workflow system.

3.2 COSMOS building blocks

The five functions discussed in Section 2 are covered respectively by five corresponding building blocks in the COSMOS reference architecture underlying the COSMOS prototype implementation (see Fig. 2).

These functions are tightly integrated since they communicate with each other by using contracts as the common representation for the data transferred. All parts of the respective COSMOS prototype may be used optionally by the market participants; It may, for example, happen, that a consortium was already formed before contract negotiations start - such that the catalogue and broker functions are not needed in this case. In other cases, no negotiation is required since the negotiation process itself would be too costly compared with the transaction volume. Finally, one may think of scenarios where no workflow execution is necessary or possible. The configuration of COSMOS components is thus dynamic and depends on the specific business requirements of different kinds of application (i.e. business) transactions.

The COSMOS reference architecture abstracts from implementation technology. This concerns not only its software components but also the contract model. Several current technologies can therefore be used for realizing the respective COSMOS E-Commerce negotiation support prototype implementation, e.g.:

- Current *Web technology* provides the highest performance for online access. This is also the area where new standards emerge at the highest pace.
- *the OMG CORBA standard* promises independence from proprietary hardware and operating systems, here BOCA gains increasing visibility [17]
- Several Frameworks are available for a '*plain Java*' approach, e.g., Voyager [18] for the communication platform or IBM's San Francisco Framework [10] for building component-based applications.
- Finally, also legacy technologies such as EDI has to be supported in the future as an (optionally) integrated part of such an E-Commerce systems infrastructure architecture.

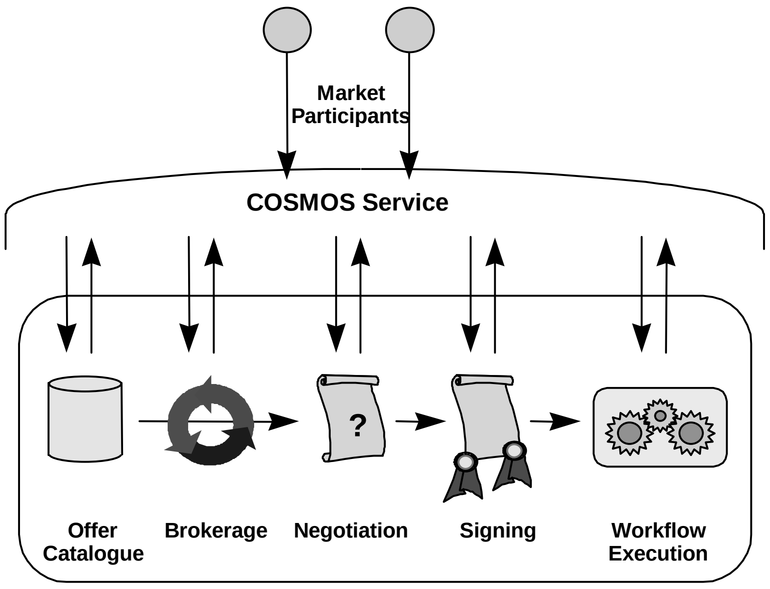

Fig. 2. Building blocks of the COSMOS architecture

4 Conclusions and further developments

System support for electronic commerce is a most practically relevant topic for distributed systems research and technology. In order to meet the important inherent openness and flexibility needs of global electronic markets, it requires ad-hoc software integration both at the system as well at the application level. At the same time, these requirements causes several problems for the introduction of electronic commerce applications: On one hand side, they need to be standardized in order to properly cooperate with one another, but they also need to be dynamically deployable, extensible, and integratable on the other. As a result, we face a ‘balkanized’ separation of electronic commerce tools and technologies today that can only be made interoperable if they adhere to certain standards. However, standardization is a long-term process during which technology development often makes many of the efforts spent there obsolete.

Under this circumstances, the rationale for a generic system infrastructure like the one developed in the COSMOS project is to decompose its functional components into consecutive phases, rather than horizontally into abstraction layers. Then the links between these components can be standardized at the level of the contract model; and, on the other side, any technology decisions for individual component developers are deliberately left open. A most important prerequisite for realizing such a system architecture, however, is a (much more general) *open* and *dynamically* controllable *component based* approach to (also application level) software development.

After having implemented the required functions for the support of commercial transactions, current research developments address additional technological refinements. To give an example, *generic support for auction systems* that is currently being developed at Hamburg University [21] will be integrated to support group negotiation patterns. Another example is the integration of market participants by following the *component-based approach to software integration.* A possible direction has been described in [13].

Finally, *mobile agent technology* is incorporated in the COSMOS project for the transfer of contracts between negotiating parties. Additional rationale for the applicability of this technology is given in [7].

References

1. Adler, W. Lamersdorf, M. Münke, S. Rücker, H. Spahn, U. Berger, A. Brüggemann-Klein, C. Haber "Grey Literature and Multiple Collections in NCSTRL" In: A. Barth, M. Breu, A. Endres, A. de Kemp (Eds.): ‘Digital Libraries in Computer Science: The MeDoc Approach’, Lecture Notes in Computer Science, vol. 1392, Springer-Verlag, Berlin Heidelberg, New York, 1998, pp. 45-170
2. Chalupsky, T. Finin, R. Fritzson, D. McKay, S. Shapiro, G. Wiederhold: "An overview of KQML: A Knowledge Query and Manipulation Language". Technical Report, April 1992
3. Commerce Net Home Page: http://www.commerce.net
4. COSMOS Project Home Page: http://www.ponton-hamburg.de/cosmos
5. DynamiCS Project Home Page: http://vsys-www.informatik.uni-hamburg.de/projects/dynamics/index.phtml
6. Electronic Commerce Homepage of the European Commission: http://www.ispo.cec.be/ecommerce/
7. Griffel, T. Tu, M. Münke, M. Merz, W. Lamersdorf, M. Mira da Silva "Electronic Contract Negotiation as an Application Niche for Mobile Agents" in: Proc. 1st International Workshop on Enterprise Distributed Object Computing, October 1997
8. Griffel: "Componentware", dpunkt-Verlag, Heidelberg, June 1998
9. Griffel, T. Tu, W. Lamersdorf (Eds.): "Electronic Commerce", dpunkt-Verlag, Heidelberg, June 1998
10. http://www.ibm.com/Java/Sanfrancisco/ technical.html, 1998
11. ISO/IEC IS 13235-1, ITU/T Draft Rec X950 - 1, Part 1; ODP Trader Specification, 1997
12. Lamersdorf, M. Merz (Eds.): "Trends in Distributed Systems for Electronic Commerce", Lecture Notes in Computer Science, vol. 1402, Springer-Verlag, Berlin Heidelberg New York, June 1998
13. Merz, F. Griffel, S. Müller-Wilken, W. Lamersdorf: "Electronic Contracting with COSMOS — How to Establish, Negotiate, and Execute Electronic Contracts on the Internet". In: Proc. 2nd International Workshop on Enterprise Distributed Object Computing, San Diego, Nov. 1998 (to appear)
14. Müller, K. Müller-Jones, W. Lamersdorf, T. Tu: "Global Trader Cooperation in Open Service Markets", in: O. Spaniol, C. Linhoff-Popien, B. Meyer (Hrgs.): Proc. Workshop 'Trends in Distributed Systems: CORBA and Beyond', Lecture Notes in Computer Science, vol. 1161, Springer-Verlag, Heidelberg, Oktober 1996, pp.214-227

15. McConnell, M. Merz, L. Maesano, M. Witthaut: "An Open Architecture for Electronic Commerce". OMG/ECDTF/OSM Response, 1997
16. AT&T, DSTC, DEC, HP, ICL, Nortel, and Novell. Trading Object Service, OMG Document No.: orbos/96-05-06, Version 1.0, 1996
17. Object Management Group: "CORBA BOCA - Business Object Component Architecture", Specification, OMG Document Nr. bom/98-01-07, 1998
18. ObjectSpace. Voyager - Core Technology User Guide, 1997, www.objectspace.com/voyager/documentation.html
19. Beat F. Schmid, Markus A. Lindemann (Eds.): Proceedings of the 31st Annual Hawaii International Conference on Systems Science, HICCS'98, Vol. IV, pp. 193-201, Hawaii, January 6-9 1998, 01/1998
20. Tu, F. Griffel, M. Merz, W. Lamersdorf: "Generic Policy Management for Open Service Markets", in: H. Koenig, K. Geihs (Hrsg.): Proc. IFIP International Working Conference on 'Distributed Applications and Interoperable Systemes' (DAIS'97), Chapman & Hall, London/Weinheim/New York, Oktober 1997, pp.211-222
21. Tu, F. Griffel, M. Merz, W. Lamersdorf "A Plug-in Architecture Providing Dynamic Negotiation Capabilities for Mobile Agents" in: K. Rothermel, F. Hohl (Eds.): Proc. 2nd International Workshop on 'Mobile Agents', MA'98, Stuttgart, Lecture Notes in Computer Science, Springer-Verlag, Berlin Heidelberg New York, September 1998
22. Abarca et al.: TINA Service Architecture, Telecommunications Networking Information Architecture Consortium 1997,
    http://www.tinac. com/u/tinac/97/services/docs/sa/sa5.0/final/