# The Transaction Internet Protocol in Practice: Reliability for WWW Applications

Hartmut Vogler, Marie-Luise Moschgath, Thomas Kunkelmann, Jürgen Grünewald

***Abstract*-- One of the most interesting domains of future Internet applications will be located in the area of electronic commerce, where online products and services are offered. Products of value are accounted by monetary transactions, involving a bank or credit card service to handle the electronic cash flow. To achieve a reliable and consistent flow of information in these applications, the concept of transactions has been proven to be the right choice. With the standardization of Transaction Internet Protocol (TIP) there is now a protocol available for the synchronization of distributed Internet applications dealing with transactions.**

**In this paper, we give a detailed overview of TIP and how it can be integrated in electronic commerce applications on the WWW. We have implemented the current version of TIP (3.0) in Java. Based on this TIP implementation we developed an electronic shopping mall application for the usage in the WWW**

***Index terms*-- distributed transactions, TIP, electronic shopping mall, WWW**

## I. Introduction

The future development of the Internet strongly depends on the commercial profit that can be achieved with new applications and services. Currently, most companies using Internet and WWW technologies are restricted to services like advertising and support, but they cannot achieve real profit from Internet business applications. However, the most interesting area of future Internet applications will be located in the area of electronic commerce, where online products and services are offered. Products of value are accounted by monetary transactions, involving a bank or credit card service to handle the electronic cash flow.

To achieve a reliable and consistent flow of information in distributed systems, the concept of transactions has been proven to be the right choice. Transactions offer all of the mechanisms needed for exchanging information of value, such as the data and cash transferred in electronic commerce applications. The drawback of transaction processing is the lack of a common protocol with which distributed nodes synchronize their activities. The protocols used for this purpose are either proprietary, or they are bound to a specific network architecture or middleware platform.

With the standardization of *Transaction Internet Protocol* (TIP) there is now a protocol available for the synchronization of distributed Internet applications dealing with transactions. The advantages of TIP are that it is fully integrated in the Internet protocol suite and that it can easily be implemented, resulting in compact code for the protocol state machine.

These advantages make TIP ideally suited for electronic commerce applications on the WWW. The protocol integration results in easy access to TIP messages from WWW browsers, in the same way as it is possible today for protocols like FTP and telnet. The small footprint paves the way for integration of TIP in Java applets and WWW scripts, where small code sizes are necessary to keep download time acceptable to users.

In this paper, we present TIP and how it can be integrated in electronic commerce applications on the WWW. To achieve this goal, we have implemented the current version of TIP (3.0) in Java. We defined an API which allows other applications the usage of TIP functionalities in a consistent manner. This API is currently available for Java, but it can be ported to other programming languages as well. In Section II we introduce transaction processing in the Internet and the TIP standard. Also, we give an overview of the API specification for TIP.

As an application on top of our TIP implementation, we present in Section III the architecture of a WWW shopping mall which allows the user to buy various components from different vendors in one transaction. We developed an application for buying PC components (e.g. processor, motherboard, RAM) from different vendors. The shopping mall application is implemented as a Java applet, so it can be downloaded by the user. We compare our approach with other shopping mall applications on the WWW. Section IV concludes this paper, giving an outlook on future directions of transaction based electronic commerce applications based on transactions.

## II. Transaction Processing in the Internet

The usage of *transactions* [1] is a very popular concept for the management of large data collections. Transactions guarantee the consistency of data records when multiple users or processes perform concurrent operations on them. In general, the properties of transactions are known as the *ACID* properties (**A**tomicity, an indivisible set of opera-

Darmstadt University of Technology
Information Technology Transfer Office
Wilhelminenstr. 7
D-64283 Darmstadt, Germany
Vogler@ITO.TU-Darmstadt.de

tions; **C**onsistency is guaranteed for the data; **I**solation of parallel data access in different transactions; **D**urability, the results are stored permanently or completely rejected) [9]. The access of distributed resources, e.g. databases on different computers, within a transaction is called a *distributed transaction*. To commit the result, the peers involved in a distributed transaction usually communicate via the *two-phase-commit* protocol (2PC).

### A. Middleware Support for Transaction Processing

In the last few years the *Common Object Request Broker Architecture* (CORBA) [16] of the *Object Management Group* (OMG) has become the most important platform for communication and development of applications in heterogeneous distributed systems. The OMG has specified several object services, such as naming and timing services, for CORBA. One of these services is the *Object Transaction Service* (OTS) [17] for object-oriented distributed transaction processing.
The OMG also defined a language mapping for Java [18] and there already exist several CORBA implementations for Java, e.g. OrbixWeb from IONA [11]. Additionally, Sun Microsystems published a mapping of OTS to Java, the *Java Transaction Service* (JTS) [15]. Unfortunately, there exists only a specification and no full reference implementation.
There are also first experiences and implementations using transaction processing on the Internet [3] from DIGITAL. The TP Internet Server combines DIGITAL's transaction processing software ACMSxp [4] and the AltaVista tunneling products [5].
These approaches have in common that they are bound to a specific transaction processing system of a specific vendor. It is not possible to extend and to integrate these closed systems with other transaction processing systems and products without additional work, such as implementing a bridge or a gateway [12]. For example, different OTS conformant implementations like OrbTP from Bull [2] and M3/OTM from BEA Systems [7] cannot directly interoperate.

### B. Transaction Internet Protocol

The most interesting approach for transaction processing in the Internet is the recently published *Transaction Internet Protocol* (TIP) by Microsoft and Tandem [13]. Currently, TIP version 3.0 has the status of a proposed standard at the *Internet Engineering Task Force* (IETF) and is distributed as the *Request for Comments* (RFC) 2371.
Based on the publicly accessible reference implementation of TIP version 1.0 in Java [20], we implemented the actual version 3.0 including the multiplexing approach but without the *Transport Layer Security* (TLS) [21].
TIP supports the one-phase and two-phase commit protocols and is based on TCP/IP. For every node involved in a transaction, a separate TCP/IP connection will be established. Because TIP makes use of the end-to-end communication mechanisms of TCP/IP like timeouts, the implementation is rather simple and easy to extend for special purposes like agent communication and migration [23]. The main objective of TIP lays in the definition of state machines for the node involved in a transaction in order to achieve a common state of the whole transactional connection.
TIP uses a *two-pipe* model, where transaction synchronization massages and the application messages are separated. The *peer-to-peer* communication for synchronization between the transaction managers using TIP is session-oriented. It is possible to multiplex different transactions over one TIP connection between two transaction managers. On the other hand, the applications can communicate with other protocols or paradigms, e.g. Java-RMI, or CORBA.

#### 1) Pushing and pulling model

TIP distinguishes two models, pushing and pulling a transaction, for propagating a transaction between two nodes. Figure 1 illustrates the pushing mechanism, where the transaction manager (TM) of the initiating node propagates a transaction identifier to a node participating to the transaction, before any resource managers (RM) residing on that node become involved.

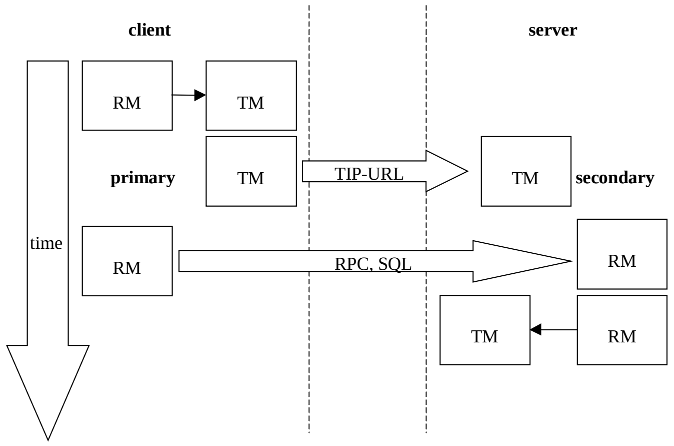

**Figure 1:**Push model

In contrast to this approach in the pulling model illustrated in Figure 2, the client addresses a resource on a remote node, which then demands a new transaction identifier from its transaction manager. This transaction manager now pulls the identifier from the initiating node. In both cases the party that opens the TCP/IP connection for TIP is called the *primary* node, the responding party is the *secondary* node.

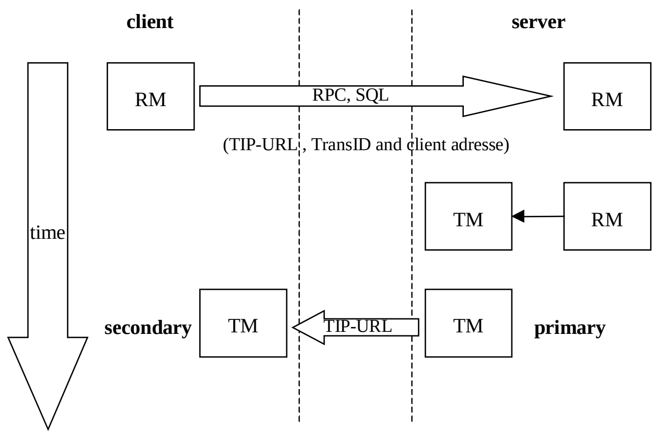

**Figure 2:** Pull model

*2) Commands and responses*

Communication in TIP is based on a command-response protocol. The primary sends a command and the secondary can select a responses related to the command. TIP defines commands either pertaining to a connection (`IDENTIFY, MULTIPLEX, TLS`) or to a transaction (`ABORT, BEGIN, COMMIT, PREPARE, PULL, PUSH, QUERY, RECONNECT`). For example a secondary can response to a `PREPARE` command with `PREPARED`, `ABORTED`, `READONLY` or `ERROR`. As is usual in Internet protocols, messages in TIP are defined as plain ASCII strings with valid ASCII characters from 32 to 127.

*3) States*

The protocol defines several states in which a node can reside; for each state a subset of protocol commands is allowed to perform a transition into another state. Figure 3 illustrates the various states. It is important to understand that these are states of the connection and not states of the transaction.

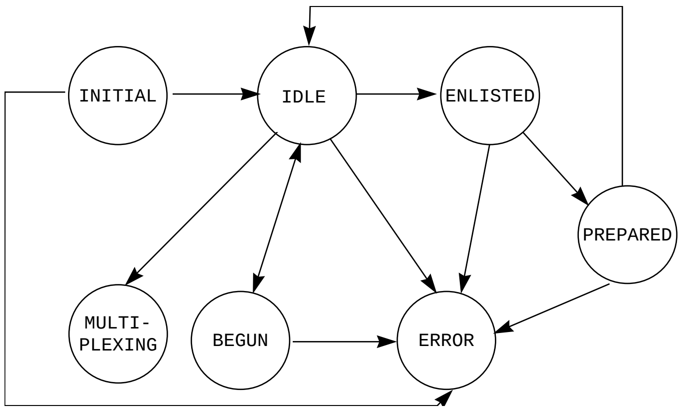

**Figure 3**: Valid states of a TIP connection

Valid states are:

- `INITIAL`: This is the starting point of the TIP state machine after a connection between the transaction managers is established. In this state the nodes agree on a protocol version.
- `IDLE`: The primary and the secondary have agreed on a protocol version, and the primary has supplied an identifier to the secondary for reconnecting after a failure. There is no transaction associated with the connection in this state.
- `ENLISTED`: The connection is now associated with an active transaction, which can be completed by a one-phase or two-phase protocol.
- `PREPARED`: A connection is associated with a transaction that has been prepared.
- `MULTIPLEXING`: The connection is being used by a multiplexing protocol, which provides its own set of connections.
- `ERROR`: An error occurred and the connection with the associated transaction will be aborted. If the connection comes from `PREPARED`, a new connection must be established to complete the commit.
- `BEGUN`: The connection is associated with an active transaction, which can only be completed by a one-phase protocol. Figure 4 shows the state diagram of a successful transaction using the push model. It also shows the commands and responses to synchronize the states at both nodes.

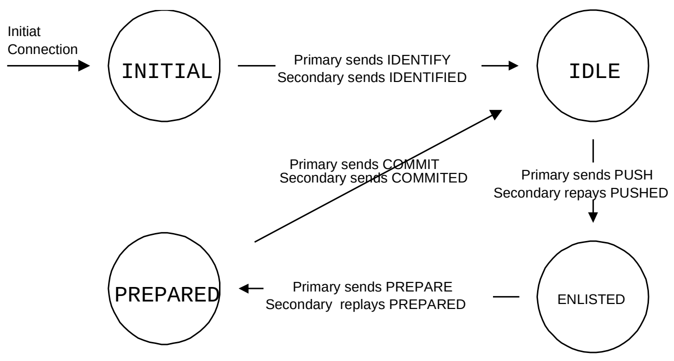

**Figure 4**: State diagram of a successfully committed transaction in TIP

All commands which are invalid in the current state result in a transition into the ERROR state. The error state can also be reached by sending an ERROR response command. Enhancements of TIP to the ordinary 2PC protocol are the possibility of reconnecting a previously disrupted connection and, in version 3.0 of TIP, it is possible to multiplex transactions over a single TCP/IP connection.
Transactions are identified in TIP by transaction identifiers. A transaction identifier can be any valid ASCII string; it is not specified how it has to be constructed. The IDs of other transaction processing systems can be propagated to nodes which are involved in a transaction using TIP if they can be represented in or transformed to ASCII notation.

## C. API Specification for TIP

In [6] an API for the communication between application programs and transaction monitors is specified. The transaction monitors in this model are supposed to be TIP-aware, i.e. they have a built-in TIP protocol machine. The API and its integration into the TIP two-pipe model described in the previous section are shown in Figure 5

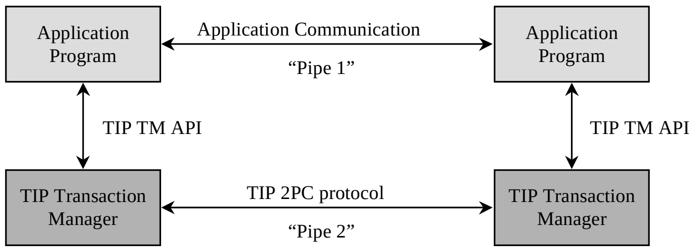

**Figure 5:** API between an application and a TIP-TM and its integration in the two-pipe model

The drawback of this solution is that the TM must be aware on a specific TIP implementation. To be completely independent from the TIP version or implementation, an additional API is necessary between the TM and the TIP state machine (SM), as shown in Figure 6.

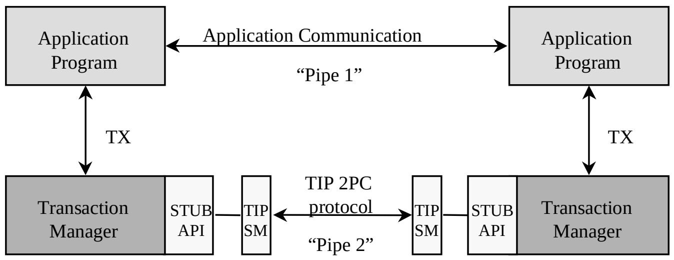

**Figure 6:** Our API between TIP state machine implementation and TM

Our investigations have shown that the best solution for this API is a specification similar to the XA+ interface defined by X/Open [25]. This offers the possibility that TIP can be used by a wide range of transaction managers that can "talk" XA. Our API specification for the TIP state machine is now closely related to the function calls of XA+. Thus, a transaction manager can use different TIP implementations. They need not to be aware which version is supported by the state machine implementation. Additionally, if the application uses the X/Open TX interface [22] to communicate with the transaction manager, only marginal changes are necessary to use existing transaction managers. With this API the transaction managers do not have to be TIP-aware.

## III. TIP-Based Shopping Mall

TIP is well suited as an underlying coordination protocol for many new distributed Web applications. To demonstrate the feasibility of transaction processing with TIP and the possibility of integrating transaction processing in WWW technology, we developed and implemented an architecture for an electronic shopping mall [19] using TIP. As a sample application we chose a scenario where different vendors of hardware and software components are collected in an electronic shopping mall with a common WWW interface to the user. A client of the system can configure her new computer by choosing parts from different vendors, and then purchase the whole computer system in one transaction. Before the purchase, it is necessary to consult the individual vendors to lock the components if they are in stock. If all computer components are available the purchase can be initiated. It is necessary to combine those individual business affairs in one transaction for this example, since it does not make sense to buy a specific motherboard if the corresponding CPU from a different vendor is not available anymore.

### *A. Architecture of the Shopping Mall Implementation*

In our shopping mall we combine various common techniques and protocols, which we extend to a valuable application infrastructure. Figure 7 illustrates the overall architecture and the protocols that are used for communication.

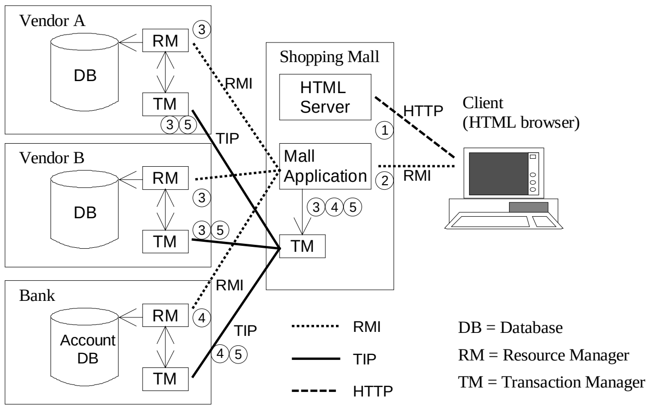

**Figure 7:** Electronic shopping mall architecture, data flow in a purchase transaction

The architecture of our shopping mall represents a typical *three-tier-system*. As a *front-end tier* (that site close to the user, shown on the right side of Figure 7), we use a Web browser with Java capabilities. This thin client downloads the shopping mall Web page with an applet for navigation and presentation to the user. The servers operated by the different vendors presenting their products and the bank server for the payment are located in the *back-end* tier, close to the data resources (the left side of Figure 7). The *middle tier* (center of Figure 7) consists of the shopping mall server containing a WWW server and the mall application. This server is responsible for routing and controlling the work between the front end and the back end.
The numbers in Figure 7 represent the typical steps that are performed during a purchase transaction:

① The WWW server at the shopping mall offers a standard interface, i.e. Web pages with a reference to a Java applet for the clients. With the shopping mall Web page a small applet is downloaded via HTTP to the client and is executed within the client's HTML browser.

② The applet allows the client to walk through the virtual shopping mall. In addition to the flexible presentation of the products, such as a simple picture or video presentation, the applet can contain a shopping agent who assists the client while shopping, e.g. help finding the right combination of motherboard and CPU. After the customer has selected all components of her new computer, she triggers the order and the payment by calling a method at the mall application via RMI.

③ For each order, the mall application creates a new branch object which runs in its own thread. These objects are the originator of a transaction and start this transaction by calling the transaction manager with a `begin_transaction(..)` method. Although the transaction synchronization is always performed using TIP, the call of resources on the servers can be made with any protocol due to the two-pipe model of TIP. For our example, the RMI protocol is used here, since all components of our model are realized in Java.

④ Additional servers can be included as necessary in the transaction by request of a vendor or the mall application, such as the bank server in our example. Depending on the originator of such an extension, the tree of distributed transactional nodes can become a chain of hosts, a tree of depth one as in the example shown, or any intermediate form. All these kinds of transaction trees can be supported with TIP.

⑤ After performing all ordering and purchase actions, the mall application closes the transaction using the 2PC protocol with TIP. Depending on the result of all votes, the servers now can update their database and initiate the shipment of the ordered products.

The shopping mall Web server and the mall application have to be located on the same machine, because a Java applet can only open an RMI connection to the server from which it was downloaded. This fact is determined by the sandbox principle of Java; for security reasons , an RMI connection cannot be opened to an arbitrary server.

### *B. Pushing and Pulling a Transaction*

In the current version of the mall implementation we use the push model for propagating the transaction. The mall application acts as the primary, pushing the transaction to the different vendors, which are the secondaries of the transaction.
In our scenario the mall application is the coordinator of the transaction. In some cases it might be useful to bring the client into the transaction, e.g. a card reader which debits money from an electronic purse. In this case, the client uses the pull mechanism for propagating the transaction. The mall application is again the primary, coordinating the transaction, and can now propagate the transaction using the push model to the various vendors. This example demonstrates how to use the different models for propagating a transaction and how to combine them.

### *C. Advantages of our Architecture*

Web-based shopping malls in practice are realized in various forms. One common aspect is that nearly all of them are on closed systems where transaction processing takes place only at one node, so the need for a distributed transaction control protocol is not necessary [14]. Even if the shopping mall server is physically distributed on several hosts, the transaction synchronization can be carried out with proprietary database protocols, or with standardized solutions like OTS on top of a CORBA platform.
The advantage of using TIP for transaction monitoring lies in the fact that it is independent of a specific base infrastructure and middleware platform. The system becomes open for the integration of heterogeneous server architectures, with no base technology in common except that they are connected via the Internet or a similar network. All other forms of distributed transaction processing rely on a specific architecture. For example, for OTS, a CORBA platform must be installed on every server involved.

## IV. Conclusion and Future Directions

TIP is a very interesting approach to achieve reliability and fault tolerance for Internet applications. Because TIP is a standardized protocol of the IETF, there now exists the ability to synchronize distributed Internet applications dealing with transactions. TIP is not bound to a specific transaction processing system; it can be used as an interoperability protocol. There already exists an approach for interworking OTS and the *Microsoft Transaction Server* (MTS) using TIP [24].
In this paper we gave a detailed overview of TIP and how it can be integrated into a WWW application. Based on our implementation of TIP version 3.0 in Java, we developed an electronic shopping mall. As a sample application for our shopping mall an user can buy components of a computer at different vendors. A purchase take place only if all chosen components are available.
In our opinion TIP has the potential to become the basis for the next generation of transaction processing. The definition of an API to access TIP in a standardized manner from applications will be an important step in this direction.
One future enhancement for electronic commerce applications based on TIP is the integration of the *Transport Layer Security* (TLS) protocol [21]. TLS is an addendum to the TCP/IP stack and protects communication against bugging or modification. TLS evolved from other network layer security mechanisms like *Secure Socket Layer* (SSL) [8] by Netscape, and other proprietary solutions. For the moment, TLS is still in its specification phase. There is no implementation for TLS available, since the protocol is not standardized yet.
Together with TIP, which provides a *reliable* synchronization between distributed nodes in a transaction, TLS offers *confidential* communication between distributed nodes. Both attributes are the two major security demands for electronic commerce applications in open and heterogeneous environments like the Internet. If they can be fulfilled,

electronic commerce applications will have a promising future.

## LITERATURE

[1] P. Bernstein, E. Newcomer: Principles of Transaction Processing, ISBN 1-55860-415-4, Morgan Kaufmann Publisher, 1996

[2] Groupe Bull: *OrbTP: Transaction Processing for Object Request Broker*, White Paper, Version 1.2, http://www-frec.bull.com/dom/orbtp/doc/orbtpwp12.doc, 1997

[3] Digital Equipment Corporation*: TP on the Internet*, http://www.software.digital.com/tpi/, 1998

[4] Digital Equipment Corporation*: ACMSxp*, http://www.software.digital.com/ACMSxp/, 1998

[5] Digital Equipment Corporation*: AltaVista Tunnel Software,* http://www.altavista.software.digital.com/tunnel/, 1998

[6] K. Evans, J. Klein, J. Lyon: *Transaction Internet Protocol - Requirements and Supplemental Information*, IETF RFC 2372, ftp://ftp.isi.edu/in-notes/rfc2372.txt, 1998

[7] J. Edwards: *The BEA M3 Object Management System: Making Objects Mission-Critical*, White Paper, http://www.beasys.com/products/m3/jwp/index.htm, 1998

[8] A.O. Freier, O. Karlton, P.C. Kocher: *The SSL Protocol Version 3.0*, ftp://ietf.org/internet-drafts/draft-ietf-ssl-version3-00.txt, 1996

[9] J. Gray, A. Reuter: *Transaction Processing: Concepts and Techniques*, ISBN 1-55860-190-2, Morgan Kaufmann Publisher, 1993

[10] J. Grünewald: *Transaktionsverarbeitung für verteilte Anwendungen im Internet*, Diploma thesis at the Darmstadt University of Technology, 1998 (in German)

[11] IONA Technologies: *Homepage of OrbixWeb 3.0* http://www.iona.com/products/internet/orbixweb/, 1998

[12] T. Kunkelmann, H. Vogler, S. Thomas: *Interoperability of Distributed Transaction Processing Systems*, Proc. Int'l Workshop on Trends in Distributed Systems (TREDS'96), Aachen, Germany, Springer Verlag LNCS 1161, 1996

[13] J. Lyon, K. Evans, J. Klein: *Transaction Internet Protocol Verison3.0*, IETF RFC 2371, ftp://ftp.isi.edu/in-notes/rfc2371.txt, 1998

[14] D.-M. Lincke, P. Haertsch, Ch. P. Hoffmann, M. A. Lindemann: *Integrierte Electronic Commerce Systeme – Auswahlkriterien und Evaluation aktueller Produktangebote*, University of St. Gallen (Switzerland), Report Nr. BusinessMedia/58/WZMO, 1997, http://www-iwi.unisg.ch/iwi4/cc/em/

[15] V. Matena, R. Cattell: JTS: *A Java Transaction Service API*, http://www.javasoft.com/products/jts, 1996

[16] Object Management Group: *The Object Request Broker: Architecture and Specification*, Updated Revision 2.1, 1997

[17] Object Management Group: *Object Transaction Service*, Updated Revision 1.1, 1997

[18] Object Management Group: *IDL/Java Mapping*, Joint Revised Submission, 1997

[19] B. Schmid, M. Lindemann: *Elemente eines Referenzmodells Elektronischer Märkte*, Tutorial "Elektronische Märkte" (electronic markets), WI'97, 1997

[20] Transaction Internet Protocol, *Reference Implementation*, http://204.203.124.10/pdc/docs/TIP.zip, 1997

[21] IETF Transport Layer Security (TLS) Working group, http://www.consensus.com/ietf-tls/, 1997

[22] X/Open CAE Specification,: *Distributed Transaction Processing: The TX (Transaction Demarcation) Specification*, X/Open Company Ltd., 1995

[23] H. Vogler, T. Kunkelmann, M.-L. Moschgath: *Distributed Transaction Processing as a Reliability Concept for Mobile Agents*, Proc. 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'97), Tunis, Tunisia, IEEE Computer Society, 1997

[24] F. Vogt, S.-H. Oh, P. Baasch, P. Furniss: *OTS/TIP bridges Version 1*, Deliverable D2ac for the ACTS project ACTranS (AC081), 1998

[25] X/Open Snapshot: *Distributed Transaction Processing: The XA+ Specification*, Version 2, X/Open Company Ltd., 1994