Distributed, Interoperable Workflow Support for Electronic Commerce*

Mike P. Papazoglou[1], Manfred A. Jeusfeld[1]
Hans Weigand[1], and Matthias Jarke[2]

[1] INFOLAB, Tilburg University, 5000 LE Tilburg, The Netherlands
{mikep,jeusfeld,weigand}@kub.nl,
WWW home page: http://infolabwww.kub.nl/infolab
[2] RWTH Aachen, Informatik V, 52056 Aachen, Germany
jarke@informatik.rwth-aachen.de,
WWW home page: http://www-i5.informatik.rwth-aachen.de

Abstract. This paper describes a flexible distributed transactional workflow environment based on an extensible object-oriented framework built around class libraries, application programming interfaces, and shared services. The purpose of this environment is to support a range of EC-like business activities including the support of financial transactions and electronic contracts. This environment has as its aim to provide key infrastructure services for mediating and monitoring electronic commerce.

1 Introduction

Present business-to-business EC implementations automate only a small portion of the electronic transaction process. For example, although ordering and distribution of goods can be fast, the supporting accounting and inventory information, payment and actual funds transfer – which require communication with database servers – tends to lag by a substantial amount of time. This time-lag and the decoupling of accounting and payment information from the ordering and delivery of goods and service processes, increases the transactions credit risks and often leads to discrepancies between various information sources requiring expensive and time-consuming reconciliations. Current applications do not yet provide the robust transaction, messaging and data access services typical of contemporary client/server applications. While there is considerable interest in developing robust Internet applications, protection of significant investments in client/server technology and interoperation with mainframe transaction servers and legacy systems is a serious requirement.

Such issues are better addressed by an integration of the organization's business systems and legacy data with the Web and workflow management systems based on distributed object technologies. To be successful with EC applications workflow systems should be able to support an integrated view of all business

* This research is supported in part by the the European Union under the ESPRIT project MEMO (Mediating and Monitoring Electronic Commerce).

elements that cut across departmental boundaries and manage the entire business operational flow. This requires integrating business functions, application program interfaces, and databases across departments and groups. This type of *distributed workflow technology* [10] allows business processes to be shared and passed across the value chain. This encourages networks of highly efficient virtual organizations which will challenge the conventional business paradigm. The most fully evolved and fully functional Internet-based organizations integrate their databases with the Web to offer what is known as *transactional commerce* [7]. The general idea is to use customer specific information to guide the transactions between a company and a customer, while providing adequate security and performance.

The combination of EC transactions and distributed workflows, as advocated in this paper, provides the sequence of business activities, arrangement for the delivery of work to the appropriate inter-organizational resources, tracking of the status of business activities, coordination of the flow of information of (inter and intra-) organizational activities and the possibility to decide among alternative execution paths. This results in the streamlining of business procedures and more efficient communication between business partners. The seamless fusion of distributed workflow and open nested (flexible) transaction technologies [3] enables the development of applications that facilitate the effective integration of routines and business processes across organizations and enable the exploitation of distinctive competencies (from collaborating business partners) without leakage at organizational boundaries.

In this paper we briefly describe an architectural framework that permits the flexibility, interoperability and openness needed for EC applications rather than a collection of independent solutions that may not work in concert. Following this we concentrate on the fusion of distributed workflow management and flexible transaction technology to provide support for EC and we describe EC components such as transactions, workflows and contracts. Issues such as confidentiality, security and authentication are out of the scope of this paper.

2 Related work

This work is related to the following research activities.

The CommerceNet consortium, a leading consortium for Internet commerce, has recently proposed the EcoSystem [2] an object-oriented architectural framework for Internet commerce involving both e-commerce vendors and end users. The Eco system comprises applications and services; a common business language for applications to communicate; an extensible set of interface specifications, class libraries and network services; and a layer of middleware that insulates applications from each other and from platform dependencies.

NetBill is a set of protocols for commerce in information goods and other networked delivered services [1]. This protocol emphasizes atomicity, security and privacy of transactions and certified delivery mechanisms. However, it performs a lot of centralized computations on the NetBill server that checks digital signatures, funds availability and requires customers to have an account with this sever.

Lehmann [8] proposes an ontology-based EDI, which uses a *concept dictionary* that maintains appropriate EDI labels and descriptions of products. The concept dictionary stores the semantic definition of the meaning of each EDI element. This approach is done in two phases. In the first phase, the trading partners negotiate by exchanging product type definitions and synchronize their concept dictionaries. The second phase is the interchange of transactions. This approach prescribes the means by which two trading partners' systems can reach an agreement on common terms and concepts and the set of data needed for such a transaction but does not specify the form of actual transactions.

3 Strawman Reference Architecture for EC Brokering

In this section we describe an open architectural framework for Internet-based EC. Purpose of this architecture is to mediate business-to-business communications and act as a central market-place where enterprises can find authoritative information and use contracting support facilities to effectively conduct their business transactions. This architecture is currently being developed as part of the ESPRIT project MEMO (MEdiating and MOnitoring Electronic Commerce) which aims at designing and developing a core electronic intermediary for electronic commerce. In particular, this project addresses the following issues:

1. Development of a flexible framework for navigating, searching and retrieving pertinent information from a large number of interconnected information repositories containing semi-structured business data.
2. Provision of a secure storage of meta data relevant for managing electronic commerce between business partners while facilitating adaption to specialized markets.
3. Provision of notarial-like services which support the negotiation and contracting phase of electronic commerce by means of a formal language for business communications (FLBC). This language lets application parties communicate by means of meaningful messages and protocols that model business terms and communications based on speech acts and illocutionary logic [16].
4. The coupling of the FLBC with distributed workflows in a way that can result in the execution of EC-like flexible transactions.

We view the EC-Brokering and Notarial Service (ECBNS) as a system that performs mediating tasks in the world of electronic commerce and facilitates the evolution of the Internet into a interconnected marketplace, supporting the exchange of information regarding a wide variety of customers, suppliers, products and services. The ECBNS contains the following modules (see Figure 1.):

1. A Search Engine that allows searching and browsing electronic directories and catalogs containing information about potential business partners. The

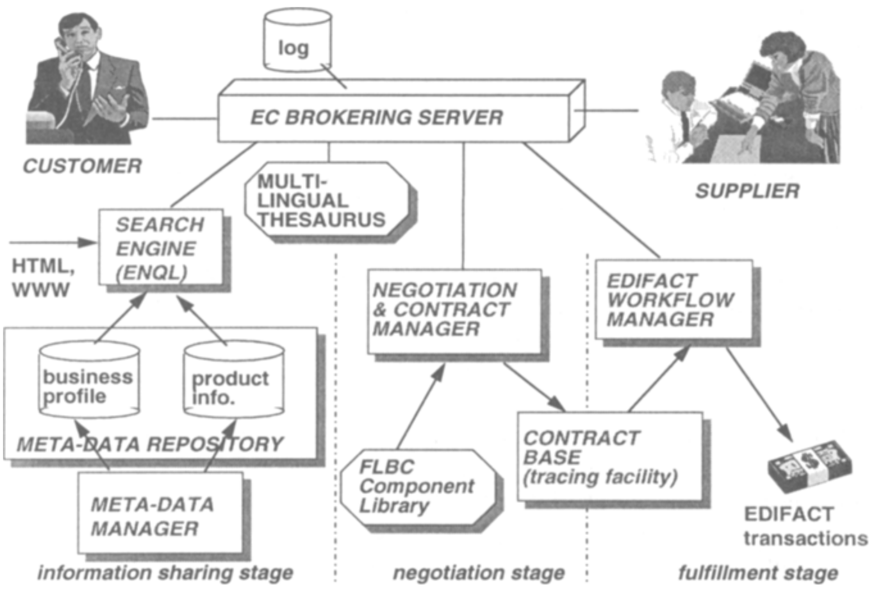

Fig. 1. The EBCNS architecture.

Search Engine makes use of a Meta-Data repository (business profiles, product database) and external databases. The search engine uses a standard terminology in order to describe products in a particular domain and maps between a product an its functionality in order to support queries that ask for products providing a given functionality.

2. A Negotiating and Contracting Manager which supports negotiating and specifying the terms of a transaction – that is, the terms of exchange and payment. These terms may cover delivery, refund policies, arranging for credit, installment payments, distribution rights, etc. These terms are standardized and the negotiation protocol(s) are described by means of FLBC. FLBC message types, as well as higher-level components, are stored in the FLBC component library.
3. The final result of the Negotiating and Contracting is a business contract that is stored in the Contract Base. This serves as input to the EDIFACT workflow manager that supports the execution of the contract with standard EDI software.
4. The Workflow Manager which automates and maps cross-organizational business processes, relating to billing and accounting services, debit/credit, invoicing, etc, to flexible EC transactions.

The log of the ECBNS plays a central role in terms of reliability, authentication and non-repudiation between business partners. The ECBNS must be able to recover from software and hardware failures. Thus the log has to contain all necessary information to re-start it when a crash has occurred. There are two kinds of applications that may be affected:

1. Transactions on the meta-data repository. These can be restarted after recovery from the crash using conventional database techniques.
2. Contract enactments and EC-like transactions. These are more subtle since they are long-running activities and their failure would create immense problems for business deals. A business deal is a collection of interrelated contracts between a known group of business partners. Here recovery techniques for flexible transactions can be used [3].

Additionally, the log records vital information about the partners involved in a business deal. It stores the start and end time end of a contract enactment, communications inside a workflow together with state information of the workflow, and events inside transactions. Partners involved in a contract negotiation and a contract enactment have read permission on the log file, i.e. they can use it to learn about the state of their business relation. Non-fullfillment of an obligation (encoded as a statement in FLBC) is also regarded as an event and recorded in the log. Such data may be used as legal evidence provided that the partners have agreed on the legal relevance of the log file.

In MEMO it is anticipated that there would be several instances of the ECBNS (just like ORBs for CORBA-enabled applications) and each of them will be managed and monitored by a Trusted Third Party (TTP), e.g., a financial institution, a Chamber of Commerce, etc. TTPs act here perform mediating tasks between customers and suppliers. In this way this architecture has the potential to evolve into a interconnected marketplace, facilitating the exchange of a wide variety of products and services.

4 A Layered approach to Electronic Commerce

The architecture presented in section-2 will consist of an extensible object-oriented framework (class libraries, APIs, and shared services) from which developers can assemble applications from existing components. These applications could subsequently be reused in other applications. The EC application framework is a framework for building Internet Vertical-Market (IV-Market) applications based on modeling support for key business processes and services. For this purpose we follow a layered approach and view an EC application framework, based on the architecture depicted in Figure 1, as comprising several layers (see Figure 2). Because these layers are built on each other, the resulting applications are tightly linked through an infrastructure of shared services. The EC application framework layers are organized as follows:

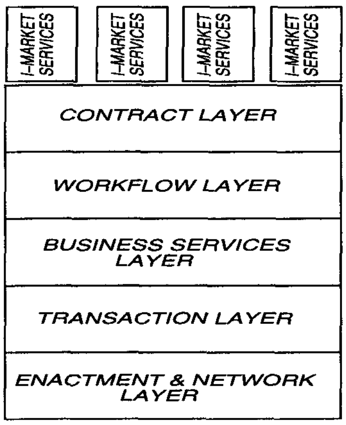

Fig. 2. Electronic Commerce Layers.

IV-Market Services: this layer comprises of services specific to closely aligned vertical markets, such as real estate, securities trading, manufacturing, or any vertical supply chain.

Contract Layer: this layer comprises services that allow buyers and sellers to coordinate their business activities represented in terms of cross-organizational workflow communication activities that describe contractual elements such obligations, violation-conditions, and sanctions.

Workflow Layer: this layer allows the representation of business processes that cut across organizational and geographic boundaries.

Business Services Layer: this layer represents generic business processes and application components common to multiple IV-Markets. These include retail (shopping order fullfilment and shipping) and business-to-business functions (procurement, order entry, inventory, supply chain management, etc).

Transaction Services: the business services and workflow layers are implemented as value-added business and workflow capabilities layered on top of a flexible EC transaction service layer. This layer provides flexible transaction support for such services as funds transfer, payment, billing and accounting services, invoicing, remittance, debit/credit and models contingency, exception and remedial facilities.

Enactment/Network Layer: this layer provides the run-time environment and the reliability/security services to accommodate mission-critical business requirements. It caters for initiating, executing, sequencing and controlling instances of a process definition in conjunction with multi-cast protocols, delivery receipts, authenticated packages and smart firewalls [2].

Figure 3 depicts a meta-model that specifies the basic entities involved in contracts, workflows, transactions and business activities. These are the principal parameters to be specified in designing IV-Market applications. In the following we will describe the key entities involved in this meta-model starting from transactions.

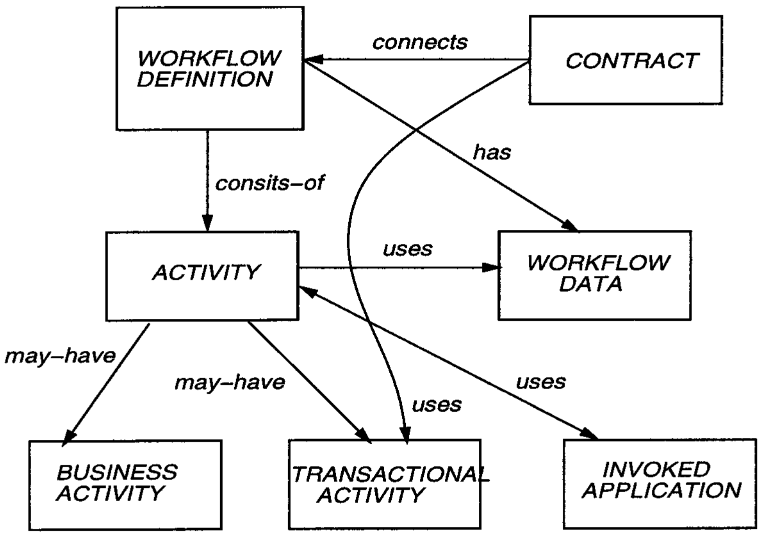

Fig. 3. Meta-model specifying basic entities involved in the EC layers.

4.1 Characteristics of EC Transactions

EC transactions are generally governed by contracts and update accounts may include the exchange of bills and invoices, and exchange of financial information services. EC transactions must provide modeling support and mediate communication, interaction, and coordination among collaborating people and business activities within and between organizations. Transaction support for such cooperative applications demands non-traditional and rather complex mechanisms [6] to support the sharing of uncommitted data between concurrently active (and possibly nested) subtransactions which may have long duration. This concept is known as *open nesting* and has been proposed by the multi-database community to increase transaction concurrency and throughput. However, it requires relaxation of standard properties of the traditional database transaction model such as serializability and isolation [3]. Flexible transaction models address only partly the requirements of workflow applications. Part of the problem is that EC transactions substantially differ from conventional nested database transactions as they may include a variety of unconventional *behavioral* features[1] which include the following:

1. *General purpose information:*
   (a) who is involved in the transaction.
   (b) what is being transacted.
   (c) the destination of payment and delivery.

[1] Some of the characteristics of EC transactions can also be found in the National Information Initiative's (NII) white paper on Electronic Commerce [9].

   (d) the transaction time frame.
   (e) permissible operations.
2. *Special purpose information:*
   (a) links to other transactions.
   (b) receipts and acknowledgments.
   (c) identification of money transferred outside national boundaries.
3. *Advanced functionality:*
   (a) the ability to support reversible (compensatible) and repaired (contingency) transactions.
   (b) the ability to reconcile and link transactions with other transactions.
   (c) the ability to specify contractual agreements, liabilities and dispute resolution policies.
   (d) the ability to support secure EDI transactions that guarantee integrity of information, confidentiality and non-repudiation.
   (e) the ability for transactions to be monitored logged and recovered.

In contrast to flexible transaction models, EC transactions and workflow applications, due to their very nature, are not data-centered. EC characteristics are better addressed by a process-centered approach to transaction management that supports long-lived concurrent, nested, multi-threaded activities [10]. We are currently developing a framework that addresses this situation by providing unified programming and flexible transaction support necessary to program network-centric workflow applications [10] . The properties of EC transactions (occasionally referred to as actions) in this work are summarized in the following:

Specification of compensating actions: these are used to undo, from a semantic point of view, the effects of an action at a particular site. Rules defining compensating transactions are attached to objects and facilities are provided for transactions to distinguish between such tasks and proceed accordingly.

Specification of contingency actions: to execute in case that a given transaction fails. Actions can be *vital* or *non-vital.* If a vital transaction aborts, then its parent must abort. Non-vital actions can be simulated by serial-alternative and parallel alternative schedulers, whereas serial and parallel schedulers can simulate vital transactions.

Unsafe Commitment: actions normally communicate via shared objects and data structures. Once an activity commits, its effects become automatically visible to other activities. To avoid corrupting shared data structures and violate internal consistency – in case tat uncommitted transactions have "observed" each other's intermediate results – the notion of *unsafe commit* is introduced.

Primitive Transaction Class: this type provides low level transaction functionality and supports primitives such as `unsafe commit`, and `cancel`.

Atomic Transaction Class: this class uses primitive classes to materialize atomic units of work such as contingency and compensatable actions.

Scheduling Classes: these implement the scheduling and synchronization processes described in the workflow layer.

In summary, our approach to EC transactions allows transactions to be nested, shared data, reversed, repaired, monitored, logged/recorded, audited, reconciled and linked with other transactions.

4.2 The Business Services Layer

The business service layer holds a collection of predefined workflow components. The components are encoded knowledge about how certain sequences of workflow actions can be performed. For example, a component `shipByAgency` may specify the standard way of sending a product `pr` via a transport agency `ag` from a supplier `sup` to a customer `cust`, irrespectively of the companies being involved. The components of the business services layer are parameterized. The advantage of such components is that certain parts of a workflow must not be defined from scratch.

The approach taken here resembles the reference models found in the ARIS toolset [13]. Reference models in ARIS are diagrams which represent a prototypical model of an aspect of an enterprise. Instead of modeling the enterprise from scratch, one can copy the suitable reference model and fit it to the specific requirements of the application.

4.3 The Workflow Layer

We base the workflow layer on previous work in progress reported in [10] where we provide transactional semantics for distributed workflow programs on the basis of an object library. This library supports specifying the interactions between transactions and workflows which are long-lived activities characterized by a well-defined set of process actions. The workflow layer models processes as complex and possibly nested transactions involving customers, performers, and conditions for satisfaction (including scheduling constraints) for the process as a whole, as well as for each milestone action within the process. Work units that can be found on the leaf-level of such an activity tree are mapped to conventional ACID database transactions. The attributes that apply to each workflow activity (process), which may spawn transactions (actions), can be summarized as follows:

Pre-activity/post-activity conditions: These are the conditions under which a particular activity can be enacted and can be terminated. Actions may be specified to start before or after the activity starts or terminates. Such scheduling actions can be serial actions (executed sequentially), parallel, event-to-start (conditional), serial alternative, and parallel alternative actions. These workflow *types* are described in what follows.

Serial scheduler: this allows actions to be submitted and committed sequentially. The actions within this scheduler establish a *begin-on-commit* dependency with each other, i.e., an action cannot begin unless the previous one commits.

Parallel scheduler: this allows all of its actions to be submitted and executed in parallel as independent actions. These actions also commit independently. It is expected that all actions should commit before their parent commits.

Serial-alternative scheduler: this attempts actions sequentially until one produces the desired outcome. The parent only aborts if all its descendent actions were tried unsuccessfully or if the transaction is timed out.

Parallel-alternative scheduler: where alternative choices are pursued in parallel until one succeeds.

Downloadable scheduler: whereby workflows can be downloaded remotely and executed locally. This is possible in cases where organizations partner very closely to each other and may actually wish to implement homogeneous environments and download workflow scripts to each other for just-in-time execution [12].

Scheduling constraints and dependencies: activity scheduling and timing can be specified as part of workflow activities, thereby allowing more flexibility for transaction scheduling.

Exception handling: pre-defined handlers can be executed depending on the type of exception raised. This allows to trap and distinguish different kinds of aborts and special messages.

Specification of commit dependencies: between actions so that a task waits for a signal from another, i.e., blocks, before it is allowed to commit. Also automatic cancel procedures are provided to semantically undo the effects of unsafely-committed actions if the global activity fails.

In the following we give a high-level view of how a workflow process can be coded. For this purpose we use natural language-like description.

```
class PROCESS-NAME{parameters} is TYPE with
          Local variables
          work is
                    scheduling & synchronization constraints
                    action-execution
                    .........
          end  (status); – –work
          – – blocking & synchronizing statements
          – – produce results
end(status);
```

As shown a workflow process has a type, i.e., scheduler, that can take the values described previously. After instantiation a process object executes its *work* routine which describes the process *script*, i.e., the sequence of actions, e.g., other workflows or transactions, it executes over its lifespan. The *work* statement materializes the behavioral part of a process object which provides the means to create other processes, actions, and objects at remote sites; and to request asynchronous execution of their features and to communicate with them. A process can spawn actions, each with its own independent thread of control, executed in the sequence specified in the *work* statement. These actions may have to be synchronized.

4.4 Contract Layer

Electronic contracts link cash flows to the exchanges of products, goods, and services rendered. Contracts include instructions regarding the handling, routing, scheduling, storing and workflow of the contract itself and of the objects contained or referenced by the contract. Contract instructions can address liabilities, acceptance forms of payment, terms of payment, billing and payment instructions, delivery instructions, return policies, methods of dispute resolution, and so on [9].

In [15] we have extended the workflow constructs described above to model situations that involve contracts and obligations. A contract is modeled as a semantic agreement between two or more collaborating (distributed) workflows in terms of a protocol-oriented specification of obligations. Contracts spell out the conditions under which transactions representing payments are to be made and include payment and other contract related instructions in the form of obligations. Contracts specify also conditions under which a contract can be reviewed, violated and sanctions for potential contract violations. In general, a contract represents a reciprocal relationship between a customer and a supplier(s) and typically consists of two or more interconnected workflow loops. Overall a contract specifies the exchange of a special set of synchronization messages at predefined activities or events mediated between two (or more) workflow enactments. A specification of a contract based on the workflows and transactions, described in the previous subsections, follows.

```
contract class PROCESS-NAME{connected-workflows, parameters} is SERIAL
with
          Local variables
          - -blocking & synchronizing statements
          obligation is
                    obligation-execution
                    changed_by-part
                    violation-part
                    sanction-part
                    .........
          end       ; - -obligation
          obligation is
                    .........
          end       ; - -obligation
                    .........
end-contract;
```

The fact that a contract is a serial action implies that its obligations must be executed in sequence.

5 Implementation Strategy

MEMO is designed as a user-driven project where small and medium enterprises review the quality of the service in three milestones. The user groups are organized by Chambers of Commerce in Germany and the Netherlands, as well as by an Internet service provider in Spain. At the first milestone, the meta data repository will be made available to the user groups. The second milestone adds the search engine and the negotiation support. Finally, the integrated EC broker will be constructed and evaluated within a major trade bank (which also serves as the coordinating partner in MEMO). The integrated EC broker will be implemented on the basis of existing building blocks such as the meta-data repository and the search engine.

The meta data repository is based on the ConceptBase system [5]. ConceptBase has a highly flexible data model for meta data and an expressive deductive query language. It can capture and link information from heterogeneous data sources at different abstraction levels. For example, ConceptBase can record the various product description schemata as well as their instances, the product data.

The search engine is based on an extension of an initial prototype system [11] developed around subject gateways which provide subject related terminology and search facilities. It is anticipated that a subject gateway will be developed for each IV-Market to include product denitiions and descriptions as well as a standard terminology for interaction.

6 Summary

In this paper we have described architectural requirements for electronic commerce relying on core constructs such as automated workflows, transactions and electronic contracts required for transacting parties. In particular we described a process-centric environment based on an extensible object-oriented framework (class libraries, application programming interfaces, and shared services) on which EC application developers can develop and assemble applications from standard components or even specialize existing applications. This EC development environment is realized on a unification of concepts from object-oriented programming with distributed interprocess communication, core open-nested transaction primitives and the notion of workflows and contracts to provide modelling solutions for advanced business applications.

References

1. Cox, B., Tygar, J., Sirbu, M.: NetBill Security and Transaction Protocol. Proceedings 1st Usenix Workshop 1995, htttp://www.nin.cmu.edu/NETBILL/pubs.html.
2. Cross-Industry Working Team: Electronic Commerce in the National Information Initiative. White paper, winter 1995, http://nii.nist.gov/pubs.

3. Elmargarmid, A.: Transaction Models for Advanced Database Applications. *Morgan-Kaufmann*, February, 1992, San Mateo, CA.
4. Georgakopoulos, D., Hornick, M., Sheth, A.: An Overview of Workflow Management: from Process Modeling to Workflow Automation Infrastructure. Distributed and Parallel Databases **3** (1995) 119–153.
5. Jarke, M., Gallersdörfer, R., Jeusfeld, M.A., Staudt, M., Eherer, S.: ConceptBase - a deductive object base for meta data management. Journal of Intelligent Information Systems **4**(2) (1995) 167–192
6. Hsu, M.: Workflow Systems. IEEE Data Engineering Bulletin **18**(1) (1995).
7. Kosiur, D.: Understanding Electronic Commerce. Microsoft Press (1997).
8. Lehmann, F.: Machine-negotiated Ontology-based EDI Electronic Data Interchange. Electronic Commerce: Current Research Issues & Applications, Springer-Verlag (1996).
9. National Information Infrastructure: Electronic Commerce in the NII. http://nii.nist.gov/pubs/pubs_list_and_abstract.html
10. Papazoglou, M.P., Delis, A., Bouguettaya, A., Haghjoo, M.: Class Library Support for Workflow Environments and Applications. IEEE Transactions on Computer Systems **46**(6) 673–686 (1997).
11. Papazoglou, M.P., Weigand, H., Milliner, S.: TopiCA: A Semantic Framework for Landscaping the Information Space in Federated Digital Libraries. DS-7: 7th Int'l Conf. on Data Semantics, Leysin, Switzerland, (1997).
12. Paul, S, et al.: RainMan: A Workflow System for the Internet. Proceedings USENIX Symposium on Internet Technologies and Systems, Monterey, Cal. (1997).
13. Scheer, A.W.: Business Process Reengineering - Reference Models for Industrial Business Processes. Springer-Verlag (1994).
14. Sheth, A. (ed.): NSF Workshop on Workflow and Process Automation in Information Systems, Univ. of Georgia (1996).
15. Verharen, E.M., Papazoglou, M.P.: Introducing Contracting in Distributed Transactional Workflows. Proceedings Hawaii Int'l Conf. on System Sciences (HICSS-31), Hawaii (1998).
16. Weigand, H., Verharen, E.M., Dignum, F.: Dynamic Business Models as a Basis for Interoperable Transaction Design. Information Systems **22**(2) (1997).