Electronic Contracting with COSMOS — How to Establish, Negotiate and Execute Electronic Contracts on the Internet

F. Griffel, M. Boger, H. Weinreich, W. Lamersdorf
VSYS - Computer Science Dept. - University of Hamburg
(griffel | boger | weinreic | lamersd) @ informatik.uni-hamburg.de

M. Merz
Ponton Hamburg
merz @ ponton-hamburg.de

Abstract

Today, the Internet gains more and more attraction even for small companies to contact business partners and to automate cooperation between each other. However, the smaller the company the higher the relative setup costs that are required if the complete process of a commercial transaction is to be supported. We propose COSMOS as an Internet-based electronic contracting service that facilitates commercial partners with offer catalogues, a brokerage service, contract negotiation and signing as well as contract execution. The COSMOS architecture supports these functions in an integrated, unified way. The design and execution of contracts integrates patterns from the CORBA Joint Business Object Facility.

Keywords
Electronic Contracting, Electronic Commerce, Java, CORBA, Business Objects, Componentware

1 Introduction

Every commercial transaction that crosses organizational boundaries is accompanied either implicitly or explicitly by a dedicated *contract*. This contract represents the commitment of each involved party to fulfill the defined obligations and it grants each party a corresponding right to receive a certain service, a good or a payment from the other.

In the economics literature, three main phases are distinguished for a commercial transaction [4, 5] (see also Fig. 1):

- in the *information phase*, market participants gather for possible transaction partners, compare product specifications and prices, and evaluate offers.
- After contacts have been established between participants, offers and counter-offers are exchanged during the *negotiations phase*. This negotiation process may either lead to a situation where agreed terms and conditions have been reached or the negotiation is abandoned.
- After all participants committed their participation in the contract with their signature, assets are exchanged during the *execution phase*. This phase may take from a few seconds up to several years.

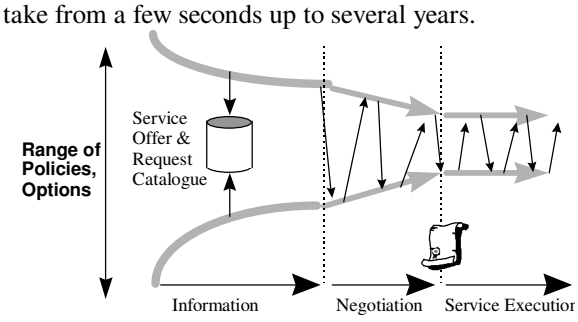


Fig. 1: Business Transactions' Phases

1.1 Electronic Contracting

So far contracts have usually been treated as merely text documents. But the concept of a *contract* is ideally suited as an integration point for a system and application level infrastructure designed to support all phases of a commercial transaction: A contract represents gathered information, agreed terms and conditions, and steps to fulfill mutual commitments in a formal way, combined into one structured document. This well-defined nature of a contract makes it a good candidate not only to be handled by, but also to control an electronically automated system. Since such a system is then driven by a contract as a single central resource a high degree of consistency and ease of use could be achieved.

1.2 High Transaction Costs

Under realistic conditions, the contracting process induces high transaction costs. Usually these costs occur as labour costs within the organization or as costs for third-party assistance. The most common costs are:

- *Information costs*. They occur as market surveillance costs due to lacking market transparency. Rating agencies, publishers and other information providers earn from commercializing their advanced insight as third parties during this phase.
- *Negotiation costs*. Negotiation can be a lengthy process which comes at high labour costs and additional other costs from legal assistance, from possible mistakes in contract structure and handling. Also the signing of a contract is a complex procedure, particularly when it involves a high number of parties.
- *Execution costs*. The management of the execution of a contract raises general coordination and surveillance costs Parties may even fail to satisfy their contractual role during the performance phase which couldcause penalty costs for e.g. not maintaining deadlines.

For the following considerations, we assume a share of transaction costs which could be considerably reduced by using electronic contracts if a proper balance of automation, standardization, business organization and legal regulation is achieved.

1.3 Balance and Complexity

Such a proper balance requires a quite complex service infrastructure with well-defined interfaces to different systems and business processes - most probably already existing.

The most prominent example of an approach to help automating and integrating interorganizational business processes may be the *Electronic Data Interchange* (EDI). EDI message exchange is quite successful, but it is kind of a "passive" data transfer leaving users with making decisions and triggering actions manually. In fact, it does not *integrate* systems and people, but only makes them *interoperate*.

This is achieved through a high degree of standardization and formality, but new and individual business processes and entities may require the design of new and unique EDI message formats. Therefore, establishing EDI may be too expensive or unsuitable for a small company whose product-portfolio is small or quickly changing.

On the other hand, today's Internet technologies offer a widely available infrastructure as a base of interoperation, but lack support of defined and agreed formats specific to different business domains.

The COSMOS project amis ate providing an infrastrucure that allows the integration of all phases of a contract in an electronic form, based on object oriented internet technology. While EDI is mainly a data exchange technology for simple "forms", COSMOS establishes a technology to *create* quite complex "forms" (contracts) in an easy way, supports their semi-automated filling and even the execution of the resulting or implicit business processes for negotiation. This allows even small companies to take advantage of an electronic support of their business transactions, by reducing the overhead during the information phase and giving the opportunity not only to offer new services and goods, but also to use those offered by others in an ad-hoc manner.

Finally, COSMOS aims at supporting the negotiation and execution phases by letting the constructed contract actively influence the processing of itself. Due to the integrated, semi-automated construction of the contract, the COSMOS system is able to consistently include execution definitions (or "flow information"), that can automatically drive the contract's fulfillment.

1.4 Organization of the paper

The rest of this paper is organized as follows: Section 2 illustrates the application background and the Business Model for an Electronic Contracting Service. For the sections following, however, we focus on *architectural issues* of the COSMOS Electronic Contracting Service. Therefore, section 3 presents the technological building blocks of COSMOS and particularly its underlying Business Object model. Then, section 4 outlines the architecture and the component-based modeling of a COSMOS contract in more detail and how the brokerage process contributes to the information included. Also, the contract's control over its own execution is described. The remaining sections discuss some next steps in the COSMOS design and development and conclude the paper. Further literature on the remaining aspects and technical details is available from the COSMOS home page http://www.ponton-hamburg. de/cosmos.

2 Application Background

Contracts may have a number of variation points (or degrees of freedom). To clarify the application field for COSMOS uses, some variations of contractual relationships are given:

- *Number of roles*: Usually two roles are defined, e.g., „buyer" and „seller" or „landlord" and „tenant". However, some contracts may be closed between three parties who organize a circular transfer of goods. In

the case of constructions, contracts are closed between one developer and a set of subcontractors, each providing a separate service.

- *Number of parties*: In simple cases contracts are closed between two parties, where each party plays a single role. However, in complex cases one role may be played by a set of parties, e.g. two tenants, or several buyers who purchase goods from a retailer.
- *Number of related contracts*. Often different types of contracts are related in a „transactional" way, such that either all of these contracts are signed or none of them is considered as legally (and therefore semantically) correct. An example is the combination of a real estate purchase with a mortgage contract.
- *Inter-contract dependencies*. Finally, the contract may be related to already given contracts such as framework contracts, memos of understanding, regulations, legislation, or each party's business terms.
- *Human participants vs. Software Agents*. Usually, a contract party is represented by a human or actually is a human being. In the Electronic Contracting scenario, a company may be represented by a specialized software agent which follows a given negotiation policy and works on or with entities belonging to the particular company. These entities (people, dates, products, departments etc.) are represented as *Business Objects*, i.e. encapsulated data taking care of its own presentation and modification by carrying additional code. Such an BO directly models a concept from the application domain and assures a high degree of consistency. This is especially important for an distributed infrastructure like COSMOS supporting team-work and collaboration of several people and / or systems.

2.1 An Example

Before analyzing the technological Framework for the Electronic Contracting Service, an exemplary situation is given for a compound contract that can be supported by the contracting service.

In the case of purchasing real estate property, usually a mortgage contract is closed between the buyer and a bank. This situation involves two related contracts - one between the buyer and the seller and one between the buyer and the bank. These contracts only make sense when they are closed in a „transactional" way.

Further, certain events may occur during the performance of the contract: buyer and seller need to be notified by the official land registry when the property right has been transferred, the buyer needs to be notified when payments of interest and mortgage installments to the bank, are due. Finally, services from several further parties may be used in order to perform the contract: a notary needs to be involved to avoid repudiation of the contract, several financial and administrative authorities may be involved, etc.

To facilitate the complex procedures required for property purchases, not only the organizational relationships of the parties should be defined in a contract but also the process definition which enables a workflow system to automatically execute the contract after being closed. Further, the contract should be recyclable, i.e., it should exist in a template form that allows for further refinements and different business adaptations. Applied to the example above, the contracting service should comprise a notification mechanism that indicates for each party when a certain action needs to be taken.

2.2 Business Model

From the organizational viewpoint, we model the following roles for contracting participants:

- *COSMOS Provider*: It is assumed that the electronic contracting service is provided by a trusted third party - e.g., a bank, a public authority, a notary, or any other organization that has a good reputation to act as a neutral facilitator.
- *Market participants*. Legal entities that offer and demand services through the COSMOS service.
- *Parties*. Market participants that participate in a contract.

In this setting, a *contract* is an entity that defines roles, parties, obligations, rights and further terms and conditions for its parties. Contract negotiation is moderated and contract execution is monitored by the COSMOS provider. The advantages to business in having this market support in place are tremendous: reduced time to market, support of virtual enterprises, increased efficiency and reduced cost.

3 Technological Foundation

This section presents some technological background that is important to understand some of the architectural design decisions explained in the section hereafter.

3.1 Business Objects as a base for smooth integration

As one of the basic architectural concepts for the COSMOS service, the *CORBA Business Objects Architecture* (BOA) approach has been chosen [6]. Business Objects provide a standardized set of components that can be deployed, inspected, and integrated across organizational boundaries at runtime.

The BOA gives the ability to specify frameworks of independent business application components. Given this, the pieces necessary for a component marketplace are put in place. However, coherence will not be guaranteed without distributed business object standards, the opportunity for application-level, large-grained components that represent business functions (orders, ledgers, customers) or even processes [cmp. 16].

The synergistic effect of BOA standards and the general usability of components is a precondition for the COSMOS market to become a reality. The BOA defines the concept of *business system domains* (BSD) which are distributed object systems that apply to particular business domains, execute under the control of the associated business entity or organization, and maintain a consistent, recoverable representation of that business domain [6].

A BSD could represent a department of a large organization, a small company or even a virtual enterprise which is established just for the execution of single tasks. However, from the electronic contracting point of view, a BSD represents a contracting party and the services provided by the party.

One distinct concept of the Business Object Model is the *Adapter*. It is used by a market participant to provide a logical access point to other BSDs' Business Objects. This preserves a company's autonomy of choosing its internal data modeling and assures a higher level of acceptance for a BO-driven architecture. Technically, the adapter can be compared to a CORBA client stub which may be instantiated separately from the CORBA object. However, in contrast to the stub, the adapter doesn't only map 1:1 the object API to the client side. It also allows to provide meta-information on the related Business Object, local logic, and a set of methods that may differ from those of the stub.

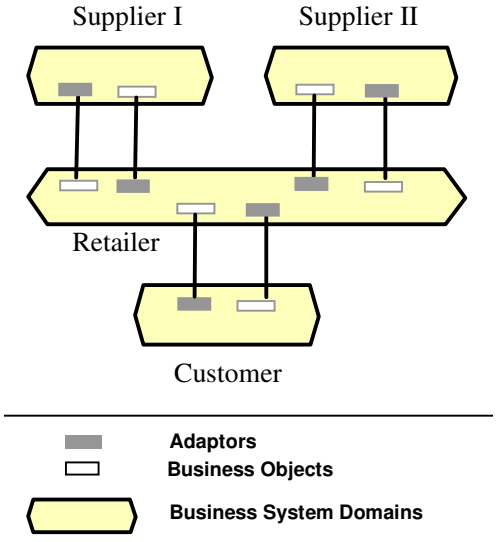


Fig. 2: Business System Domains

An adapter can thus be included as a component in other participants' local software systems. The adapter is exactly that part of an organization's object framework which is intentionally given away to the „outer world" (see Fig. 2).

Accordingly, the Electronic Contracting Service should provide an editing tool that allows users to select market participants by inspecting their respective adapters on an application level. If this participant is to be included as a contract party, the provided information is used to assign events and methods of the business objects and by doing this, single steps of a workflow are defined as a part of the contract.

3.2 Dynamic Data Types using Feature Structures

In object oriented strongly typed programming environments dynamically generated data has to confer to the static type system of the used framework. The usual way to bypass this situation is to express dynamic data information with name-value pairs. These can either be typed using simple types as string and integer, or contain an extra field for type information expressed as string. However, this is considered as a conceptual break with object orientation since this approach only support abstraction, typing, polymorphism and encapsulation in a very poor manner.

In the COSMOS project dynamic data structures play an important role. Service offers, contract templates, legal issues need to be described in a coherent way, but only evolve in a spontaneous manner as response to changing market and juridical conditions and can thus not be described in a static type system and name-value pairs are considered as insufficient.

As a solution to this conflict a technique known as „feature structure" [21, 22] has been chosen to represent dynamic data and types. A feature structure is a typed and structured collection of name-value pairs used to represent attributes of an object. They can recursively be built conforming to type information and structured in an inheritance hierarchy that also allows multiple inheritance.

Powerful unification and matching algorithms for feature structures exist.

We have extended feature structures to not only contain values but also references to real objects so that objects can be stored, retrieved and accessed from them. Thus feature structures allow a smooth incorporation with the rest of the object orientation and component based design. For example, a print service can be de-

scribed as a feature structure that contains some name-value pairs like quality and price, a feature structure describing the paper and a reference to the business object adapter of the service provider.

3.3 Implementation Platform for Distribution

The COSMOS project is designed for a highly distributed environment and based on object oriented internet technology. State of the art for such environments is Client/Server style programming with downloadable clients and platform independent portable implementations of the servers. A combination of CORBA, Java and legacy systems is usually appropriate.

In the case of COSMOS though, object migration is a desirable feature that pure Client/Server style programming fails to provide. Electronic contracts, business objects and adapters should be shippable to the site where they are needed or administrated as appropriate in a changing and developing infrastructure. Therefor Voyager[15], a Java-based ORB that provides mobility of objects and is compatible with CORBA, has been chosen as implementation platform for COSMOS. Also, experiments with extensions to Java that directly incorporate object migration and object grouping into a language called Dejay (Distributed Java) [20] are ongoing.

4 The COSMOS architecture

To create a flexible and adaptable architecture with a couple of distinct but smoothly integrated functionalities, COSMOS has some clearly separated building blocks:.

- *Online catalogues*. Information on market participants can be obtained today either through search engines or through yellow pages databases such as, e.g., the SMARTS online catalogue for business services (www.smarts.org). However, both approaches lack the required selectivity for a sector-specific collection of QoS attributes that help market participants to formulate a precise specification of the requested service or good.
- *Brokers* act on behalf of market participants to form a group of potential contract parties. They require both access to online catalogs and QoS specifications from their customers. Selection policies and navigation interfaces enable customers to use brokers in a flexible way.
- *Contract negotiation support*. Negotiation can be understood as the collaborative editing of a contract as a structured document. Each modification is then considered by the other party as an offer and may be returned as a counter-offer or a rejection. Contract negotiation may also be accompanied by additional measures such as telephone and video conferencing support. The main goal of negotiation support is to ensure, first, an integrated and consistent document processing, second, the possibility to control the negotiation process by a specific negotiation protocol [1, 2], and third, to allow negotiating parties to get attached in different ways to the contracting service.
- *Signing support*. Any document that is edited collaboratively can be signed by the participants by mechanisms known from public key systems [8]. The precondition is the existence of a unique externalized representation of the shared document that can be signed. If human users are involved in contract negotiation, the „what you see is what you sign“ motto should be followed, i.e., a visual representation of the document should be rendered. To prevent fraud, the signature service has to verify that all participants signed the same version of the document.

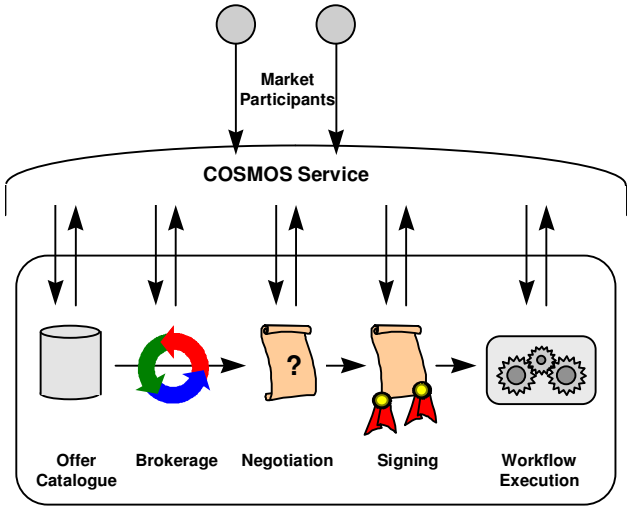


Fig. 3: Functions of an Electronic Commerce System

- *Contract execution support*. Finally, workflow software is usually applied to coordinate the performing of activities by defined roles in a defined order. Such technologies should be applicable for the contract execution phase.

Each of the function mentioned above can be individually deployed and used by market participants. However, their respective added-values vary drastically: while online catalogues are used as an island solution already today, the other components require high set-up efforts if not integrated as a chain of interworking services. The COSMOS architecture thus specifies interfaces and functions of these components such that they act as building blocks of an integrated „one-stop service“ (Fig.

3). Thus the COSMOS approach to tackle transaction costs is both *automation* and *integration* of the required software components:

4.1 Contract Object Model

COSMOS addresses the integrated support of all transaction phases. This is accomplished by using a coherent contract object model.

A contract could be considered as structured document composed out of text blocks. In this case, the editing process may be simple, 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 in an Small and Medium Enterprises (SME)/Internet context, characterized by a permanent change of rules, roles, and business subjects.

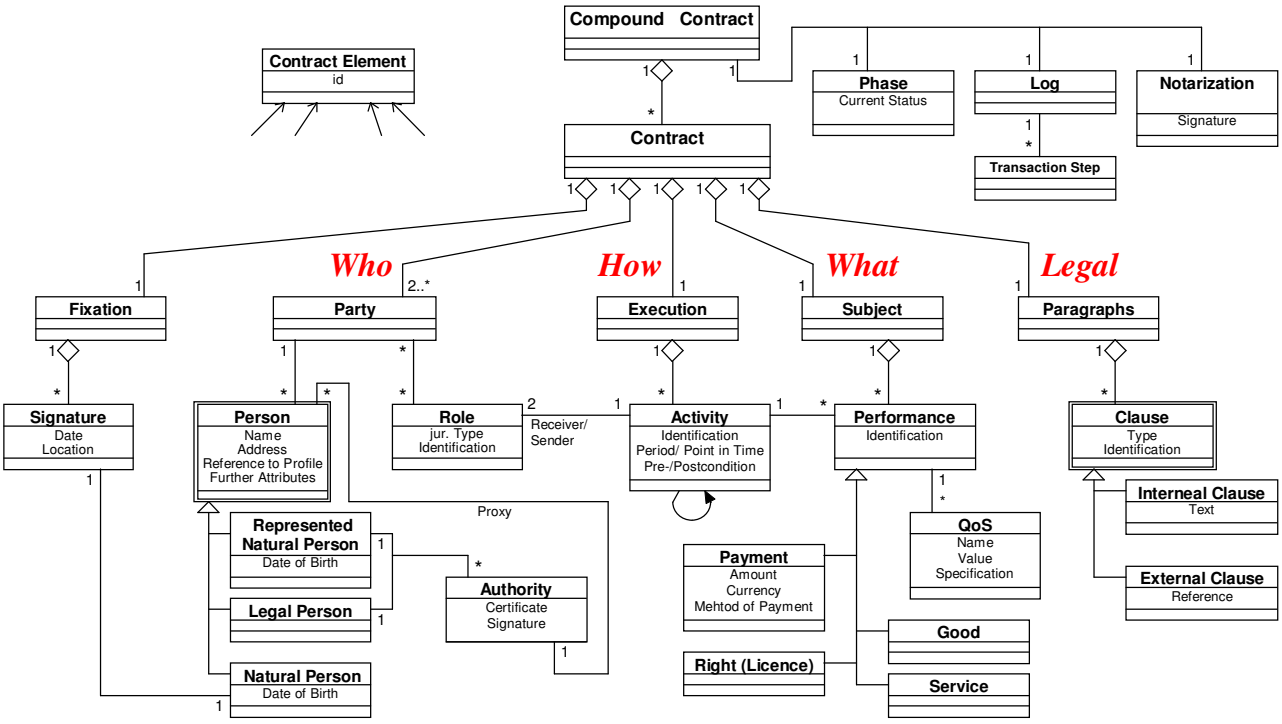
As a trade-off, the COSMOS contract model aims to identify only those semantically meaningful parts of contract instances which allow for efficient automation and therefore highest increase of the added value.

Fig. 4 gives an overview of the main component classes of the COSMOS contract model:

The parts of the contract model can be distinguished by their subject:

Fig. 4: The COSMOS Contract Model

- 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 that can either carry a value or a type information. 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 *Legal* 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 „Legal 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. It can carry type information like „party“ *is-a* „real-estate broker“ and the „good“ *is-a* „arable land“. Instead of attribute/value pairs (such as „price per acre“ = $100, „ground's humidity“ = 20%), constraint expressions are used as QoS specifications (such as „price < $150 and humidity < 30%).
- By using a *broker*, the template will be completed if suitable providers can be retrieved from the catalogue. The broker's task is to assemble type-conforming offers, requests and contract templates and 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 can be 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.

4.2 The brokerage process

The COSMOS approach to brokerage refers back to the ODP/CORBA *trader service* [7, 10]. Here the trader model defines two roles, *exporter* and *importer*:

- The exporter registers a service offer at the trader by specifying its *service type*. This is composed out of an interface type (i.e. the method signatures) and a set of service attribute types. For the referred service type, service attribute values are provided and stored in the trader's offer database.
- On the other hand, the importer specifies the requested service type by the interface type and a constraint expression for the service attributes.

If the requested service matches one or more of the stored offers, references are returned to the importer. For the contracting service, this technology is used in an analog way: market participants register their adapter as well as their specific offer attributes at the catalogue. Since one participants may register several services, different adapters may be required. In turn, for each service type, different offers may be registered [3]. A 1:N relationship between participants and adapters as well as between adapters and offers will thus be maintained.

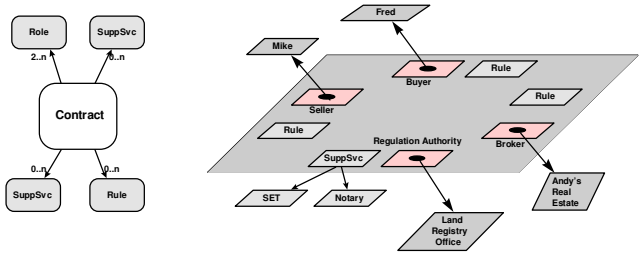


Fig. 5: Relations within a contract

For standardized services (such as the land registry office), equally standardized adapters can be deployed. I.e., the service type corresponds with the adapter class. However, if non-standard services are sought the offer database of the catalogue needs to be browsed by the contract creator.

4.3 Component-based Contracts

Since market participants are represented through Business Objects, it is assumed that each participant has registered his adapter in the catalogue. A set of offers is associated with each adapter. If a certain offer has been detected as a query result, the corresponding adapter will be provided to the client.

Regarding the contract model, Adapters as well as offer descriptions are integrated in the following way:

- Every *Person* class contains the type information and an adapter object in the contract model and
- Every *Performance* class contains a set of QoS attributes that has been taken from the offer attribute set, the type information describing it and an adapter to its business object..

In the subsequent phase, *negotiation* means to modify values of QoS attributes, types and references to business objects and to transfer the contract among the parties involved. To properly access contract clauses, the COSMOS *contract editor* (CCE) is used. This editor

allows to visualize and manipulate clauses in a comfortable way: meta-information provided by adapters can be introspected and used to map an event of one party to a method of another one.

Similar to, for example, the JavaBeans event model [11], this approach facilitates the definition of providers and receivers of services and allows a „(re-) wiring“ of the concrete business object components filled into the original contract-template.

COSMOS' approach to guarantee consistency and provide smooth integration is using business objects (BOs). BOs are stored in a company's database and may represent its people, goods etc. By directly registering these BOs as mediatable services or contracting parties with the COSMOS contract broker (CCB) any change or mistake in the representation of a BO's data is avoided. Using adapters for the registration may protect (not making it visible or accessible) parts of a BO and allows it to stay within the database originally hosting it.

Furthermore, the BOs' adapters are included into the contract assuring consistency again. Also, we gain the object-oriented paradigm's advantage of encapsulation and combination of data and code: A BO contained in a contract can represent itself in an adequate way within the CCE. It can control the correctness of its wiring during the negotiation phase as well.

The main problem in choosing such a BO approach is the mapping of business objects to databases. On the one hand, there is a model of an active, complex entity comprising data and code which should be retained during the whole transaction to gain a high degree of consistency and self-automation. On the other hand, existing database systems that are powerful enough (regarding speed and scalability) to support real-world COSMOS' applications employ relational data models and stored procedures. Thus, to retain the advantages of the BO-view to the overall architecture but being applicable in practice, COSMOS defines an abstraction layer for the database access. This layer allows to use relational as well as object-oriented databases and is able to wrap a company's legacy DBMS.

Storing contracts (and even their templates) suffers from a similar representation problem. COSMOS' contracts are based upon an object oriented model that is able to describe a contract's structure as well as its building blocks (cmp. Fig. 4). This model can be mapped to an XML (Extensible Markup Language) [www.w3.org/XML] model. Both the object model as well as the XML model can directly be used to transmit and display the contract during the negotiation phase and to store the contract. However, for signing a contract the XML representation is preferred since it is in a linear and man-readable form which is important for legal issues.

But the storage representation of this model is non-trivial, since the relationships of the described components (BO-adapters) and the components themselves have to be maintained within the database while being extractable as efficiently as possible.

Since the subject of a contract has been defined and stored including these relationships, a workflow engine will be capable to derive relevant information for the execution phase of the transaction.

4.4 Contract Execution

The information gathered within the contract during earlier phases directly allows to derive a workflow to execute a contract. The a signed contract contains much information that can be used to lead the contract to its fulfillment. In a usual text based contract this information is implicitly given through the components relations and the „legal clauses“ and sometimes explicitly as a graph representation like with a Gant diagram.

Using an electronic and object oriented approach this information together with extra information that is included in a contract template or provide by or service provider. This information is kept in the contract object model and can due to the object encapsulation be guarded consistent or can present itself with corresponding viewing objects. A graphical representation of a workflow based on petri-nets can then be generated from the contract. The reason to make the flow-model explicit firstly is to allow its manual editing by a user during the negotiation phase supplementing it with steps or correlations that cannot be expressed implicitly. Secondly, the explicit graph-model can be utilized more easily to drive the COSMOS workflow engine (CWE).

This workflow representation can be used in two ways to execute the workflow. In the first approach, the CWE is an abstraction from real-world workflow systems which has adapters to different workflow environments, automatic ones as well as ones based on notifications of humans for example using email. Basically, the CWE itself is an interpreter for colored petri-nets [13]. The petri-net model has been chosen since its expressive power comprises the wide variety of more or less powerful workflow descriptions used in nowadays workflow and groupware environments [14].

The contract's explicit flow-model mentioned above is represented as a PAMELA program interpreted by the CWE. The Petri-net based Activity Management Execution Language (PAMELA) in fact is an textual description of extended colored petri-nets [9].

Single activities triggered by the COSMOS workflow engine are mapped to method calls of the participating business objects. Since these BO originally were issued by companies' offering themselves or their services and goods to the COSMOS contract broker, each BO „knows" its interface to an individual company's environment. Thus, the contract's components are able to directly contact and drive a company's workflow system in a consistent and correct manner [cmp. 18].

For companies not having established their own internal workflow system COSMOS additionally includes a second self-installing workflow environment based on Object-Space's Voyager technology [15]. In this environment each activity within a flow is modeled as an active object (derived from the corresponding BO-adapter) that is migrated to a Voyager server-process installed at each participating contracting party. Again, this can be achieved automatically by analyzing the contract's information regarding mutual relationships of its components. Each activity corresponds to a transition in the underlying petri-net model.

The places needed by such a model are represented as JavaSpaces that can be distributed between the contracting parties or be hold centrally within the

CWE. Altogether, this results in a data-driven workflow implemented as a distributed petri-net. The neat advantage of this environment is the direct inclusion of all participants and the self-driven flow triggered by the BO-adapters contained in the signed contract.

This system will be described in full detail in another paper. Fig. 6 depicts the resulting overall architecture of COSMOS.

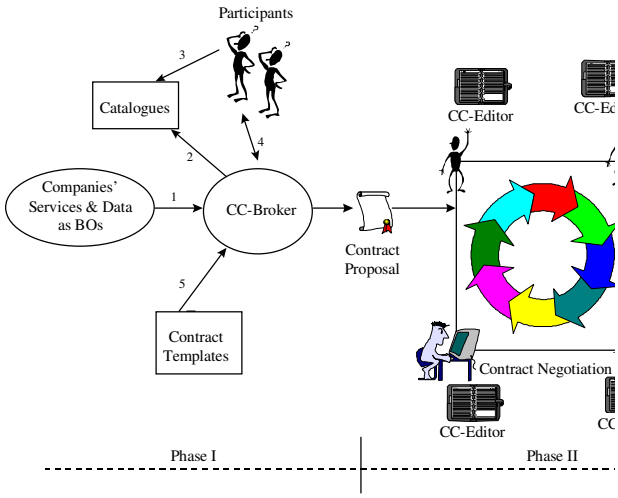


Fig. 6: COSMOS Electronic Contracting Reference Architecture

5 Implementation status and ongoing work

The early COSMOS development dealt with carefully designing the contract model and its XML representation. Also, some contract templates have been built to allow to experiment with the contracting process. A first prototype of the broker is ready to use utilizing a single simple database adapter to Oracle DBs, but lacking the full expressive power for representing complex BOs yet. Until having fully specified and implemented the COSMOS catalogue service, for experiments, the OSM catalogue [www.osm.org] is used. The contract editor's prototype can already be used to edit QoS specifications and offers within the contract. The graphical representation of the resulting workflow is in an early development stage (just presenting not modifiable yet). Security and signing [cmp. 19] employs the current JDK 1.2 betas and will be supplemented by COSMOS' own security-policy manager in near future. Regarding the workflow execution some early experiments have been conducted utilizing a Lotus Domino system. But the analyzing and surveying phase of existing workflow environments has not been finished.

On the other hand, the automatic deduction of PAMELA descriptions and their execution by the CWE already work and COSMOS' own Voyager-based workflow environment has been implemented prototypically.

Right now, most work is done elaborating on the database mapping of business objects as well as the contracts themselves. Relational and object-oriented models are under investigation and the possibilities of the direct inclusion of XML constructs are explored. Also, the use of PJama [17] as a fully persistent OO-system and its interaction with the contract model are examined.

6 Conclusion

Electronic Contracting is a promising application of today's open Internet-technology based environments with their relatively small set-up costs allowing even small companies to utilize them. On the other hand, contracting comprises some well defined phases and actions and a supporting architecture should consider all of them in an integrated and easy to handle manner. In particular, the autonomy of companies' data models and legacy systems should be retained while supporting consistent and standardized methods for interoperation and collaboration. COSMOS' approach to do so is the consequent utilization the *business object* concept throughout its infrastructure combined with corresponding adapters minimizing the reciprocal action on companies' internal data and processes. In COSMOS contracts are viewed as a kind of active documents that are able to control their

own modification and may influence at least part of their environment. This point of view is supported by highly structured document meta-models expressed with XML allowing to capture even some of the semantics of the described documents, at least assuring their consistency. The advantages of the BO-approach have to be proven in real-world scenarios yet, particularly regarding performance, handling and computational overhead. Nonetheless, COSMOS provides an infrastructure well-suited to elaborate upon such scenarios and doing further research in this quite promising direction.

7 References

[1] J. S. Rosenschein and G. Zlotkin. Rules of Encounter - Designing Conventions for Automated Negotiation among Computers. MIT Press, 1994.

[2] T. Tu, F. Griffel, M. Merz, and W. Lamersdorf. Generic Policy Management for Open Service Markets. In: H. König, K. Geihs, and T. Preuß, eds., Distributed Applications and Interoperable Systems, DAIS'97 Cottbus, Germany, pp. 211-222, Chapman & Hall, 1997.

[3] M. Merz, T. Tu, W. Lamersdorf: „Dynamic Support Service Selection for Business Transactions in Electronic Service Markets". In: *Proc. Intl. Workhop on Trends in Distributed Systems*, Aachen 1996, Springer, Berlin, Heidelberg New York 1996, pp. 183-195

[4] B. Schmid: "Electronic Markets". In: *Wirtschaftsinformatik*, 35 (1993) 5, S. 465-480.

[5] Z. Milosevic: „Enterprise Aspects Of Open Distributed Systems", PhD. Thesis, Department of Computer Science, University of Queensland, 1995.

[6] Business Object Domain Task Force: „Combined Business Object Facility", BODTF-RFP 1 Submission, OMG Document No.: bom/97-11-09.

[7] K. Müller-Jones, M. Merz, and W. Lamersdorf. The TRADEr: Integrating Trading Into DCE. In K. Raymond and L. Armstrong, eds., Open Distributed Processing: Experiences with Distributed Environments, Proceedings of the 3rd IFIP TC 6/WG 6.1 International Conference on Open Distributed Processing pp. 476-487, Chapman & Hall 1995.

[8] B. Schneier, „Applied Cryptography", John Wiley & Sons, 1996.

[9] K. Müller-Jones, M. Merz, and W. Lamersdorf, Kooperationsanwendungen: Integrierte Vorgangskontrolle und Dienstvermittlung in offenen verteilten Systemen. In F. Huber-Wäschle, H. Schauer, and P. Widmayer, eds., GISI 95 - Herausforderungen eines globalen Informationsverbundes für die Informatik, Zurich, pp. 518-525, Springer 1995.

[10] AT\&T, DSTC, DEC, HP, ICL, Nortel, and Novell. Trading Object Service, OMG Document No.: orbos/96-05-06, Version 1.0, 1996.

[11] Sun Microsystems, Java Beans 1.01 API Specification, www.javasoft.com/beans/spec.html 1997.

[12] G. Booch, J. Rumbaugh, and Ivar Jacobson. Unified Modeling Language User Guide, Addison-Wesley, 1998.

[13] K. Jensen. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Springer 1992.

[14] OVUM Workflow Computing. OVUM Report, London 1995.

[15] ObjectSpace. Voyager - Core Technology User Guide, 1997, www.objectspace.com/voyager/documentation.html.

[16] http://www.ibm.com/Java/SanFrancisco/technical.html, 1998.

[17] http://www.sunlabs.com/research/forest, 1998.

[18] I. Claßen, H. Weber, and Y. Han. Towards Evolutionary and Adaptive Workflow Systems - Infrastructure Support Based on Higher-Order Object Nets and CORBA. In Proceedings of the First International Workshop on Enterprise Distributed Object Computing, EDOC' 97, pp. 300-308, Australia, IEEE 1997.

[19] Z. Milosevic, D. Arnold, L. O' Connor, Inter-enterprise Contract Architecture For Open Distributed Systems: Security Requirements, WET ICE' 96 Workshop on Enterprise Security, Stanford, USA, June 1996.

[20] M. Boger. Migrating Object in Electronic Commerce Applications,Working Conference on Trends in Electronic Commerce, TREC'98, Hamburg, Springer, 1998.

[21] A. Zeller, G. Snelting. Unified Versioning through Feature Logic. Informatik-Bericht No. 96-01, revised version, TU Braunschweig, Feb. 1997.

[22] B. Carpenter. The Logic of Typed Feature Structures. Cambridge University Press, 1992.