A Three-Phase Model of Electronic Marketplaces for Software Components in Chemical Engineering

Mareike Schoop, Jörg Köller, Thomas List, Christoph Quix
Informatik V (Information Systems), RWTH Aachen, 52056 Aachen, Germany
{schoop, koeller, list, quix} @ informatik.rwth-aachen.de

Abstract: Electronic marketplaces in the business-to-business operate in different branches. Abstracting from these realisations of the concept of an electronic marketplace, we can derive a general model of a business transaction with the following three phases. Starting with a search for new business partners, successful negotiations lead to a contract which then needs to be fulfilled. In this paper, these three phases will be discussed in detail, emphasising the problems with current practices in electronic marketplaces. An extended model that overcomes these problems will be presented and applied to the context of trading of software components for chemicaI engineering.

1. INTRODUCTION

Electronic marketplaces in the business-to-business area have been the subject of research for a number of years. These types of marketplaces provide a forum for bringing together buyers and sellers with the aim of enabling and supporting trade. In recent years we have seen different implementations of the concept of an e-marketplace. For example, some approaches (such as [www.baunetz.de]) concentrate on providing facilities for finding new partners. Interactions leading to a business deal and fulfilling the related contract are not supported and thus need to take place outside the marketplace. Other approaches (such as [www.chemunity.com]) automate the interactions. No search is possible but a request is directly sent to approved suppliers in an auction-like manner.

In general, we can abstract from the different implementations onto a general model of a business transaction. Starting with a search for new business partners, successful negotiations lead to a contract which needs to be fulfilled. Such a three-phase model (search, negotiate, fulfil) has been used in many facets [SS01] and sometimes integrated with other views on business processes [SL98].

In this paper we will present the model in detail and discuss the current practices for each of the three phases. Problems will be pointed out and extensions to the current practices will be proposed to overcome the problems. Thus a new comprehensive model will be presented (section 2). The application context of trading software components for chemical engineering, that requires a sophisticated electronic marketplace will be introduced (sections 3 and 4). Our model will then be applied to the context, showing the specific requirements of such a marketplace (section 5). This paper concludes with a discussion of our approach.

2. A MODEL FOR A B2B E-COMMERCE PROCESS

During a commerce process, the involved participants usually go through three phases. Firstly, a party looks for potential business partners. A buyer wants to find relevant suppliers of the product (s)he is looking for; a seller might want to find potential customers for the products (s)he can supply. After locating potential (new) partners, the second step is to come to an agreement that is acceptable to all partners. Partners might bargain about the price, might find a compromise about the delivery dates, might negotiate about quality aspects of the products. The aim is to finalise a contract that specifies the business deal. Therefore, this second phase concerns negotiation about details of the agreement. If the negotiation is successful then a business deal is struck and the outcome is a contract which will then have to be processed by the partners in the third phase, e.g. concerning logistics, payment etc. The general model that can be extracted from the above observations is one of three phases.

The search phase is about finding business partners; the negotiation phase is about finding agreements leading to a contract; the fulfilment phase concerns the execution of the contract. The three-phase model is independent of any technological means, i.e. it is valid for traditional commerce processes as well as for electronic commerce interactions (see its application in the MEMO – Mediating and Monitoring Electronic Commerce – Project[57]). For example, a buyer might look for potential suppliers in the yellow pages, in the catalogues of chambers of commerce or on the internet.

In this paper we will concentrate on *electronic marketplaces for business-to-business electronic commerce.* The current practices in such marketplaces can best be discussed using an example of an existing business-to-business marketplace of the chemical industry called *chemUnity*[58]. A buyer's request containing information about the product (s)he wants to purchase, its type and concentration, the delivery address and time is transferred via the marketplace to all potential suppliers as specified by the buyer. Suppliers have a fixed amount of time (usually 25 hours) to react. Those who choose to send an offer will be taken into account. The best offer is determined by the marketplace based on the buyer's selection criteria. If the best

[57] http://www.abnamro.com/ memo/

[58] http://www.chemunity.com

offer is within the price range indicated by the buyer, then the transaction is completed and the following obligations exist: The seller must supply the product(s) indicated in the original request whereas the buyer must provide the payment according to the offer received.

Abstracting from the example, we can state general observations concerning the three phases in electronic marketplaces as follows.

The search phase consists of (extended) keyword search based on some classification, e.g. a product catalogue, a list of companies in a certain branch etc. using these kinds of search mechanisms presupposes good knowledge of the search items by the search party and an appropriately structured search domain. For example, if a company would like to find new business contacts or would like to find suppliers of certain products that have different names in different companies, then keyword-based search is clearly insufficient.

The protocols of electronic negotiations that are usually supported in elctronic marketplaces are auctions or electronic catalogues. In the latter case, the option is one of "take it or leave it" – either to order at the price specified in the catalogue or not to enter into the business transaction at all. Auctions can be useful for settings as described above. However, even in the example of *chemUnity* certain problems are obvious. Complex negotiations cannot be supported by such a model. For example, the cheapest supplier might not be the one offering the best quality, the cheapest supplier might not be trustworthy, the third cheapest supplier might be able to deliver much quicker than the cheapest one etc. Furthermore, if negotiations concern frame contracts, then a different negotiation protocol is required. Highly interactive exchanges that occur in traditional commerce can be transferred to electronic commerce where, on the one hand, the potential of information technology can be exploited to offer new functionalities and to support effective interactions and, on the other hand, information technology cannot (and indeed should not) replace the human negotiator by an automated software agent but rather support human negotiators in their tasks [SQ01].

The fulfilment phase is the one that is usually covered best in any electronic marketplace. Payment models are supported (usually payment by credit card) and an integration with the companies' logistic systems is achieved. If all goes well after the contract has been finalised then such a model is sufficient. However, if disagreements occur between the parties as to which obligations need to be fulfilled, whether certain duties have been carried out according to the agreements made during the negotiation etc., there is hardly any support to help solving such problems. No history behind an agreement is usually provided that could help the parties themselves or an independent third party to understand why certain agreements have been reached and where the specific problem lies.

To summarise, there are potential problems with respect to current practises for all three phases. Nowadays there exist a number of electronic marketplaces for different branches. Therefore, a (new) marketplace requires additional functionalities for all phases to make it attractive to participants and to distinguish it from its competitors. For example, to capture different relations between concepts, semantic search mechanisms need to be provided so that similar and related information can be found; a new negotiation protocol is required that is interaction-based and supports the communication-intensive exchanges in complex negotiations; different payment models should be provided to capture the different needs of various application contexts. Furthermore, a monitoring component could help to observe the interactions and trace them back in case of conflicts.

The three-phase model will be used throughout the present paper. Section 3 and 4 will present the area of trading software components for the chemical industry as one example application area of our work. The three phases will be reconsidered in section 5 in relation to the application area. In particular, we will discuss the realisation of some of the additional functionalities discussed above for the specific context.

3. CONTEXT: SOFTWARE COMPONENTS FOR THE CHEMICAL INDUSTRY

This section presents an application domain, namely the usage and trading of software components in computer aided process engineering (CAPE).

3.1 Software components

Modern software systems, especially large enterprise systems, tend to grow more and more complex but require at the same time increased flexibility. This flexibility facilitates easy integration of new subsystems or the extraction of functionality of parts of the system to be used elsewhere. Additionally, managing interdependencies between different subsystems in a complex enterprise environment has become a challenging task for software engineers. Therefore, the component-based approach for design and implementation has become popular and has been proven to be useful [Ho00].

Software components can be considered as the next step beyond objects. Based on existing definitions the term “software component” is used in this paper as follows: “A software component is an executable, stand-alone piece of software with a clearly defined interface and behaviour.” The component’s interface allows other pieces of software (e.g. other components) to access its functionality. There are different middleware approaches facilitating the implementation and deployment of software components by providing low level communication infrastructure, component lifecycle management, transaction services, and similar services such as (D)COM and COM+, CORBA, and Enterprise Java. In addition, several proprietary middleware systems exist.

The fact that software components are stand-alone pieces of software which can be delivered and deployed in a given environment makes them a good candidate for being traded on the web in a component marketplace. Several concepts and technical architectures for web-based component marketplaces have been developed [JGR99, Be98, BKR96, WRMT95]. However, all of these marketplaces follow a horizontal approach, i.e. the type of components that can be sold is not limited to specific domains. The horizontal approach can become a problem for various reasons (see sections 3.3 and 4)

3.2 CAPE-OPEN components

The challenges for software engineering concerning integration and flexibility of complex software systems depicted above are also relevant to the CAPE domain, especially to process simulation. Process simulators are tools designed to create mathematical models of manufacturing facilities for processing and/or transforming materials. Chemical manufacturing through continuous or batch processing, polymer processing, and oil refining are examples of such processes. Process simulators are central for designing new processes; they are also used extensively to predict behaviour of existing or proposed processes. Designing and simulating such processes is a very complex task which typically involves many different tools with strong interrelations. Most of these tools are highly specialised, expensive, and require much expertise to be run and maintained. Therefore, process simulation can be seen as a perfect candidate for applying component techniques for mutual integration and data exchange.

However, two years ago no component-based architectures were used because most existing systems had a FORTRAN core which is not even object-oriented. The tools used for process simulation were closed, monolithic applications which made it almost impossible to include new components from other vendors or to combine these tools [CO96]. However, such integration and combination is desirable, as the manual exchange of data between those application is tedious and error prone. Additionally, these tools were (and still are) highly heterogeneous because they may run on simple PCs using Windows or mainframes using Unix. To combine these tools each of them must be divided up into standardised components with defined interfaces.

This problem was addressed by the European CAPE-OPEN initiative in which the chemical industries and major vendors of process simulation software have accomplished a standard of open interfaces for process simulation software [BJ99a,BJ99b]. The overall outcome of CAPE-OPEN was the definition of a conceptual design and interface specifications for simulators which consist of an assembly of relatively large software components. Standard components of a process simulator have been identified [CO98] and the semantics of these components and their interdependencies have been clearly defined in terms of UML diagrams, COM and CORBA interface definitions, and textual descriptions [http://www.global-cape-open.org].

3.3 Requirements for a CAPE-OPEN marketplace

In the previous section we have discussed that software components are good candidates for being sold in a web-based marketplace. We have also pointed out that for making a marketplace attractive, additional services should be offered which can be effectively designed for a vertical, i.e. domain-specific, marketplace only. We will now present a set of services for trading CAPE-OPEN components via the web. In contrast to a horizontal component marketplace the CAPE-OPEN components have the following properties.

In the CAPE domain, there are only a few component types with approximately 15 different interfaces. The behaviour of a component, i.e. the way it can be integrated and used, is precisely defined. This allows us to integrate an automated standard-compliance test in the marketplace assuring a certain level of quality of the components offered. This means that if a user owns a CAPE-OPEN simulator executive and buys a CAPE-OPEN unit (s)he can be sure that it will work. Although the components' behaviour is standardised they differ widely in their internal functionality. In case of units this means that there are components for various purposes (e.g. reactors, distillation columns, etc.) which may differ only in very subtle details that can be crucial for the success of a simulation. These differences can be described exactly only because there is a underlying common terminology. Therefore, we can include a powerful semantic product search in the marketplace which would be difficult to implement in the horizontal case. Since the behaviour of the components is highly standardised, the interrelations of components available in the marketplace can be made explicit. Similar to the semantic specification of the component itself it can also be described which other components are necessary to perform a calculation. For example, it can be specified that a specific reactor unit implementation needs a thermodynamics package supporting certain properties and calculation methods. This can be used for linking components in the marketplace that will interoperate smoothly. All CAPE-OPEN interfaces are designed to run in a distributed environment. Therefore, the marketplace can support not only classical licensing models such as buying or renting components. Rather it is also possible to take licence models into account which are based on application service provider (ASP) approaches (see [KLJ01]). Additionally, due to the domain focus of the marketplace, specific user support can be offered in terms of FAQ's discussion groups, or expert forums.

Defining and offering the above services for unspecified software products is difficult as will be discussed in the next section. Another problem is that describing the exact functionality of a component (not only in terms of their interface definition) is a non trivial task. In our case this problem is solved by offering only a very limited range of components with a functionality that is clearly defined by the software standard.

4. APPLYING THE MODEL TO THE CONTEXT

In section 2, we have presented a model for a business-to-business electronic commerce transaction. It consists of three phases: search for products and business partners, negotiation about the contract, and fulfilment of the contract. Based on the requirements for a marketplace for software components, we will now apply the three-phase model to our application context.

The marketplace provides support for the three main phases search, negotiation and fulfilment, based on extended services which are required for this context. These services are certification and classification of components and management of licenses for components as described in the previous section. The architecture of the marketplace itself is component-based so that new services and functionalities can easily be integrated. In the following, each of the three main phases and the extended services will be presented in more detail.

4.1 The Search Phase

In our context of software components for the chemical industry, we will use a combination of different searching technologies. This idea has also been used in the MEMO Project [Le00, Je00]. The user can start to explore the information on the marketplace by browsing through an ontology of concepts. While browsing, the user can compose a query for a normal keyword-based search engine. The basic data model for the ontology consists of concepts which might be related to several other concepts and might have a number of attributes or properties (see upper part of figure 1).

As mentioned in section 4, finding the right component for a specific context of a chemical simulation can be difficult because of the complex requirements which have to be fulfilled. The static properties of a component (given through its interface) are well defined by the CAPE-OPEN standard. The standard describes basically three relevant interface types; the components implementing these interfaces will then be used in a CAPE-OPEN compliant process simulator. The three interfaces describe material (thermodynamics) properties, (chemical engineering) units and numerical solvers. To simulate a specific plant, a user has to choose a unit for each apparatus of the plant, specify a numerical solver for the units and the materials for the feed of the plant.

The realisation of the search engine for components faced two main problems. On the one hand there is a large variety of components for each interface type. The hierarchical structure of the ontology is used so that the engineer can easily navigate through the units. This structure is based on the model Clip [BSM01] which itself is based on pdXi, an application protocol of the STEP initiative [ISO 10303]. The extended data model for the ontology is shown in figure 3 (only the first hierarchy level is displayed).

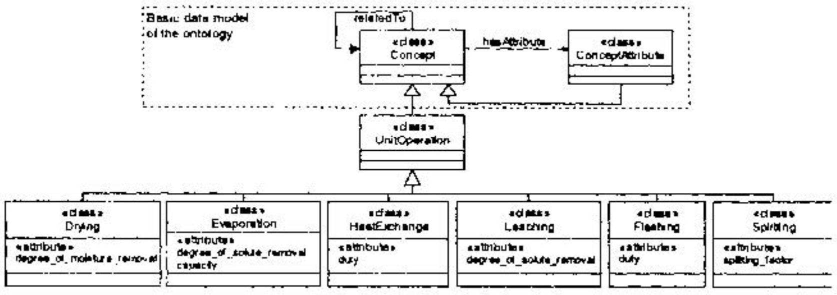

Figure 1. Data model of the ontology including the CliP hierarchy

The ontology now enables the engineer to find the apparatus (s)he wants to simulate. Units on the other hand only specify a mathematical model for an apparatus. There are many different mathematical models (and therefore many different units) that can be used to simulate one and the same apparatus. To find the right one, the chemical engineer has to take into account many constraints. For example, not every unit can be used with every numerical solver, certain numerical solvers cannot be combined in the same plant, and different mathematical models (that are coded into the units) might be better in certain situations which depend on the pressure used in the plant, the temperature and other properties. Some of these conditions, such as “which combinations of solvers can be used together”, are known and can be modelled into our ontology, see figure 2 which contains the extensions to the data model of the ontology that is used to formulate relations between units and solvers. Some of these conditions have the form of a simple rule (such as “the pressure that is used in a reactor has to be supported by the solver”). These conditions are expressed using the terms of the data model of the ontology and can be used to narrow a search explicitly through specifying the properties used by the conditions. Other conditions are based on relations between certain solvers and units (such as “solver A is known to work well with unit B”). These conditions are formulated by explicitly instantiating attributes of the data model on the instance level of the ontology. To formulate a relationship between two units that is not contained in the data model, the general `relatedTo`-Attribute can be instantiated. These relationships cannot be used to formulate a search query on the ontology, but can be used to navigate through the products. For example there can be a link between the units A and B that states that unit B can be used for free if unit A is bought.

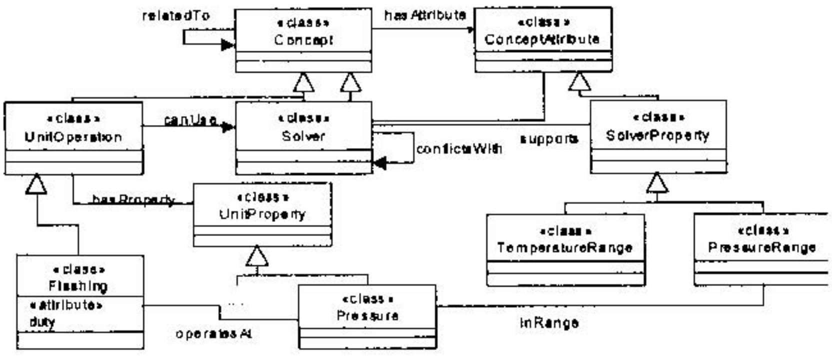

Figure 2. Relationship between units and solvers

In addition to the product-oriented ontology, our ontology contains first approaches to describe services related to the components. We describe vendors that offer components and consulting services related to these components. During a search an engineer can switch between the product view and the corresponding vendor view or navigate a "consulting" link to find vendors that offer consulting services for a specific component. In the future this basic system will be extended to enable the specification of properties of a potential component to find vendors that can develop a very specialised component.

To summarise, the main advantage of our approach for searching is based on a semantically rich ontology which provides a clear understanding of the domain. Based on this information, the user is able to specify precise search criteria that go far beyond the possibilities of normal keyword-based search. This is especially useful in an application context where the products differ only in a few properties. The ontology is (indirectly) based on a domain standard so that an engineer can use the search without special knowledge.

4.2 The Negotiation Phase

The result of the search phase is a list of components the developer is interested in and the suppliers of these components. Alternatively, the search phase can end with a list of companies the developer wants to negotiate with about a more complex product (e.g. a component which is not yet completely specified or consulting and technical support that has to be included). In addition, the user can negotiate with a company about a frame contract which applies especially to the case of a company offering several components that a developer is interested in. By doing so, the developer does not have to negotiate about every small component (s)he is going to purchase in the future.

One of the key problems in the negotiation phase is that there is no negotiation model which is appropriate for all kinds of transactions. As mentioned before, this phase can range from a simple catalogue-like ordering process to complex peer-to-peer negotiations about individual (frame) contracts. These more complicated contracts are not only about the licensing of a software component, but might also include information about consulting or technical support, the requirements for the component etc. Therefore, we envision a flexible negotiation module which supports multiple negotiation models.

In the simplest negotiation model, the products are presented with their properties and prices in an electronic catalogue. This model applies especially to "off-the-shelf" components, i.e. components that are ready to deploy with the CAPE-OPEN process simulator. The off-the-shelf components can be bought just as they are – the different variants, license models and support models with the corresponding prices are displayed. The buyer of the software component can only choose between these fixed components. For example, an engineer is able to choose the models which fits his needs best. There is no bargaining between the customer and the supplier about the price or the functionality of the component.

Another popular negotiation model in electronic commerce are auctions. Although in our opinion auctions are in general less applicable in this context, auctions should be supported by the marketplace. For example, a company might be looking for a component that fulfils certain requirements. Instead of searching for a supplier, the company can publish a request for quotations and start a reverse auction. As price is certainly not the only criterion for a customer, multi-attribute auctions should also be taken into account.

To support the more difficult case of complex negotiations that can, for example, occur when complex components, consulting services, or technical support are concerned a novel negotiation protocol is offered. This protocol emphasises the exchange of communication acts involved in negotiations and is interaction-based. We will present a brief introduction of our model; the interested reader is referred to [Sch99, SQ01] for more detailed discussions.

The approach is based on speech act elements [Ha81, Se69] and formal logics [DW95, Sch98] and supports structured electronic message exchange. Each message has a certain type (e.g. offer, request, counter-offer, discussion, quotation) that specifies how the message content is meant. The negotiation steps follow a characteristic pattern. For example, a supplier's offer can only be answered by a customer's counter-offer, acceptance, rejection, or discussion request and not by another offer of the supplier. The content of messages is represented in a semi-formal way, linked to the ontology. Taken together, the structured approach helps to prevent most misunderstandings about what exactly is meant in this type of written communication, thereby making the obligations of each partner during a negotiation explicit. The formalisation of our approach enables reasoning about the obligations. The content of negotiations can be accessed in various ways, e.g. by content (all messages concerning numerical solvers), type (all offers by company A), obligations (all open obligations, all unfulfilled obligations of company B), people involved (all messages sent by X), time (all request sent between 5/5/00 and 8/2/01) and various combinations (all offers by company X sent in 2000 that have been accepted and not yet fulfilled).

Since the message exchange is done in a structured way and the exchanges are stored, they can be used later in case of conflicts to find out what was agreed and which obligations were accepted during a negotiation.

A similar model can be used for the negotiation of frame contracts. A frame contract is a "parameterised" contract which specifies the general conditions of business relations. In the context of software components, a frame contract might specify consulting and technical support, pricing and licensing, etc. Frame contracts are necessary because chemical engineering companies are often major companies that usually need more than one component of a supplier. Both parties, customers and suppliers, may be interested in long-term business relationships. The marketplace allows negotiations about frame contracts. The conditions of these contracts are taken into account when prices are calculated. If a frame contract exists between the two companies, the customer can later order components easily by referring to the frame contract. Some conditions do not have to be discussed again as they are specified in the frame contract.

To summarise, the marketplace offers several negotiation models. Depending on the situation, the customer or the supplier can choose a model for a transaction. Independent of the negotiation model, all negotiation data can be recorded by the marketplace (if both parties agree). This enables the traceability of the negotiation. The trace of the negotiation can be used in legal conflicts. Making the obligations of the business partners explicit enables the monitoring of the execution of the contract during the fulfilment phase.

4.3 The Fulfilment Phase

Support during the fulfilment phase is given in many different ways. First, the component has to be "delivered" to the customer. Therefore, a secure download mechanism has to established that allows only authorised users access to the software. We do not plan to develop a new mechanism, rather we will rely on existing technologies.

After the component has been transferred to the customer's site, it has to be ensured that the software is not used more often as allowed in the contract. Therefore, license servers are required to control the use of the component. Different types of licenses or payment models can be handled by the marketplace. In this context, a trusted third party (TTP) can be involved to handle payment and to monitor the contract execution. The TTP controls whether both parties act according to the contract.

The payment model for off-the-shelf components can be based on a license server system. A license server has to be installed at the local network of the engineering company, called the local license server. Depending on the license model in use for a specific component, the local license server needs an online connection to the central license server of the marketplace or the TTP. The implementation supports local user control to ensure that only qualified and authorised users have access to the components. A "pay-per-use" model can also be supported by a license server.

The way a simulation component is actually bought depends on how the customer is registered at the marketplace and whether the customer has a frame contract with the supplier. Ideally, the engineer can select the component, download it and plug it into the simulator. The local license server is notified about the new license situation via the Internet, billing is done according to the agreements in the frame contract. The TTP can be involved to collect the billing information.

If no frame contract is applicable, the user can still choose to "buy" the simulation component according to the conditions given in the contract. License models that do not work on a pre-paid basis are not available. The user can download the component, license information will only be sent to the user when the payment is verified. The business transaction here is done via the marketplace, the TTP acts as a monitor and keeps logs of each transaction.

As the marketplace or the TTP can provide a license server, the license server has to provide several functionalities. The billing information for the use of components is recorded and the licenses issued by the software vendors are managed. On the one hand, a library or an interface with functions to retrieve license information from the server or to update license information on the server will be provided. These functions can be built into the components by the component developer. On the other hand, the software company must be able to maintain the licenses, e.g. issue new licenses, upgrade licenses, remove licenses. These functions will be provided by a special interface for software companies to the license server of the marketplace or the TTP.

Finally, consulting and support activities can be included in a contract about software components. Especially for consulting activities (be it for chemical engineering simulations or for general business or IT aspects) it is problematic to formulate a contract that covers all detailed specifications that were discussed in the negotiation. In the case of a misunderstanding between customer and consultant about the extent of consulting, the contract and its links to the negotiation trace can be used to resolve a potential conflict.

4.4 Data Management

Data management is one of the core services of the marketplace. Although the user has no direct interface to the data management module of the marketplace, the quality of this module may have an impact on the other modules. We will briefly discuss the functionalities which have to be delivered by the data management component. A more detailed discussion about the requirements for a business data management component can be found in [QS00].

Firstly, all the data of the marketplace has to be managed in an integrated way. As many different modules form the marketplace software, many different data models will be in use. Since data needs to be exchanged between the modules, the data management module has to manage the data in a common model. Different views on this data model will be provided as a mapping from the common data model to the specific data model of the module. Furthermore, data has to be integrated from external sources such as product catalogues, company profiles or ontologies.

A common data model of component data is especially important for searching as the information about software components should be comparable. On the other hand, a supplier wants to present his/her product in a unique look-and-feel to be distinct from other components. Therefore, data presentation can be customised based on the needs of the users of the marketplace.

The ontologies (cf. sec. 5.1) are also managed by the data management module. If a new component is registered at the marketplace, it has to be classified into the ontology. The search module is then able to access this information. Facilities for ontology maintenance will also be provided as the ontology for software components will certainly evolve over time. The data model for ontologies is currently being implemented in an extensible way, so that new classes and relationships can be added later on.

Finally, access rights and user profiles have to be managed. Not every user of the marketplace is allowed to see everything. In particular, data of negotiations and contracts should only be accessible by the parties involved in the negotiation. A company may provide additional information to its customers via the marketplace.

4.5 Implementation Issues

The proposed extension for the different phases of the electronic commerce process as described above are embedded in an overall architecture of the marketplace. Each functionality is encapsulated in one or more modules. Furthermore modules capture the transaction data. Our approach is to have a middle layer between the data modules and the ones implementing functionality to achieve the possibility to adapt the functions to different marketplace structures. It is not our goal to implement a general framework for electronic marketplaces but to offer functionality that can be combined with existing software solutions. The modules are implemented as Enterprise Java Beans (EJBs) according to the Java 2 Enterprise Edition. We use the Inprise Application Server as EJB container. We have implemented the data layer using entity beans. These beans can be replaced by an existing software solution for electronic marketplaces. The middle layer between functionality and data is realised as a set of session beans that implement a well-defined set of interfaces. The modules implementing the advanced functionality rely on these interfaces. To replace the data layer with another existing marketplace solution the session beans have to be rewritten according to the interfaces.

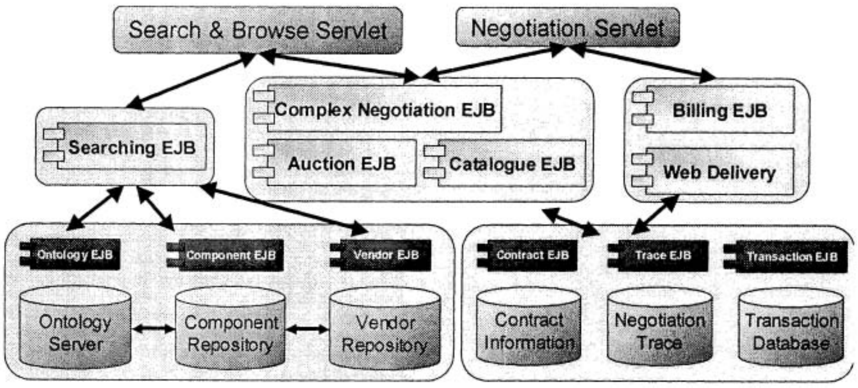

Figure 3. Overall architecture

The overall architecture is depictured in figure 3. There are several (logical) separate databases for the ontology, the components and vendor information and databases for contract information, a store for the business transactions and a trace database for the negotiation and fulfilment trace. Some databases have strong interrelations (for example ontology and component database). The coordination of the databases is realised by the data management module using a meta database (not depicted in figure 3). The middle layer consists of entity beans that encapsulate the database access and have to be tailored according to the underlying system. The search and negotiation modules are implemented as session beans that access the databases through the middle layer entity beans. The user interface is currently realised by web-based servlets. In the case of the negotiation we have three different modules in use, each representing a different business model: Auctions, complex negotiations and catalogue-based buying. These modules have to use the transaction and trace beans to create new data (business transactions and negotiation logs). Billing and delivery is then done using the related beans.

The overall architecture allows a flexible integration of services such as the ontology-based search engine or the negotiation modules with a marketplace. New alternative or additional services can be added easily. The data abstraction based on the entity bean-middle layer and the data management enables the use of the services on different marketplaces.

5. CONCLUSION

Electronic marketplaces exist in many facets. However, in order to be acceptable to users, the three phases of an electronic commerce process, namely search, negotiate, fulfil, need to be supported efficiently. We have discussed the problems with the current practices for all of the phases. For example, keyword-based search mechanisms presuppose a good knowledge of the search item and a search space that is constructed accordingly. A number of horizontal and vertical marketplaces are in operation. In order to be competitive and distinct, a marketplace nowadays thus needs to offer additional functionalities. The useful additions depend on the branch the marketplace is designed for. Therefore, most of the functionalities make only sense when developing vertical marketplaces. For example, we discussed the idea of ontology-based semantic search mechanisms.

Our extended three-phase model has been applied to the context of software components for the chemical industry. In order to show its generalisability, the model has also been used in other contexts such as the construction industry (cf. the MEMO project [www.abnamro.com/memo]). The approach presented in this paper can be combined with other frameworks. For example, the marketplace *chemUnity* could be extended by the functionalities we presented in this paper. Therefore, our model is not an implementation model but a conceptual basis for an implementation.

To summarise, we have presented a sophisticated model for a holistic support of the three phases extended by innovative services that provides an efficient framework for electronic marketplaces in the context of software components for chemical engineering. The model enables efficient support of business transactions ranging from a search module consisting of sophisticated semantic search and keyword-based search, over a negotiation module offering simple electronic catalogues, auction models, and support of complex highly interactive negotiations, to different payment models and licence models in the fulfilment phase.

6. REFERENCES

[Be98] A. Behle, An Internet-based Information System for Cooperative Software Reuse, In *Proc. of 5th IEEE International Conference on Software Reuse,* pp 236-245, 1998.

[BJ99a] B. Braunschweig , M. Jarke, A. Becks, J. Köller and C. Tresp: Designing Standards for Open Simulation Environments in the Chemical Industries: A Computer-Supported Use-Case Approach. In *Proc. of the 9th Annual Int. Symposium of the Int. Council on Systems Engineering,* Brighton, England, June, 1999.

[BJ99b] B. Braunschweig, M. Jarke, J. Köller, W. Marquardt and L .v. Wedel: CAPE-OPEN - experiences from a standardization effort in chemical industries. In *Proc. Intl. Conf. Standardization and Innovation in Information Technology (SIIT99),* Aachen 1999.

[BKR96] P. Buxmann, W. König and F. Rose: The Java Repository – An Electronic Intermediary for Java Resources, In *Proc. of the 7th Int'l Conference of the International Information Management Assoc.,* 1996.

[BSM01] B. Bayer, R. Schneider and W. Marquardt: Integration of Data Models for Process Design - First Steps and Experiences. In *Computers & Chemical Engineering* **24** (2000), 599-605.

[CO96] CAPE-OPEN, Project Programme, Annex I, Brite/EuRam Be 3512, 1996.

[CO98] CAPE-OPEN, Conceptual Design Document 2. http://www.global-cape-open.org/CAPE-OPEN_standard.html,1998.

[CO00] Global CAPE-OPEN Master Plan for CAPE-OPEN Laboratories Network, Global CAPE-OPEN project deliverable, 2000.

[DW95] F. Dignum and H. Weigand: Communication and Deontic Logic. In R.J. Wieringa and R. Feenstra (eds), Information *Systems, Correctness and Reusability, Proc. of ISCORE-94 Workshop,* Singapore, pp. 242-260, 1995.

[Ha81] J. Habermas: Theorie des kommunikativen Handelns, volume 1: Handlungsrationalität und gesellschaftliche Rationalisierung. Suhrkamp, Frankfurt am Main, 1981.

[Ho00] J. Hopkins: Component Primer, Comm. ACM, 43(10), October 2000.

[ISO 10303] ISO 10303: Part 231, Process Engineering Data: Process Design and Process Specifications of Major Equipment. ISO TC 184/SC4/WG3 N740 (1998).

[Je00] M.A. Jeusfeld: Business data structures for B2B commerce. In Proc. Conf. EMISA – Methods for Developing Information Systems and their Application, Information Systems for E-Commerce (EMISA-2000), Linz, Austria, 2000.

[JGR99] H.-A. Jacobsen, O. Günther and G. Riessen: Component Leasing on the World Wide Web, In *Proc. of the 1st ACM Conf. Electronic Commerce,* ACM Press, 1999.

[KLJ01] J. Köller, T. List and M. Jarke: Designing a Component Store for Chemical Engineering Software Solutions, In *Proc. 34 th Hawaiian Intl. Conf. On System Sciences (HICSS-34),* Maui, 2001.

[Le00] C.J. Leune: Final document on searching, querying and discovering mechanisms. *MEMO Deliverable 1.4,* July 2000 (http://www.abnamro.com/memo/).

[QS00] C. Quix and M. Schoop: Facilitating Business-to-Business Electronic Commerce for Small and Medium-Sized Enterprises. In *Proc. of the First Intl. Conf. on Electronic*

Commerce and Web Technologies (EC-Web 2000), Greenwich, UK, Springer-Verlag, pp. 442-451, September 2000.

[Sch98] M. Schoop: *Towards Effective Multidisciplinary Communication: A Language-Action Approach to* Cooperative Documentation Systems. PhD Thesis, The University of Manchester, UK, 1998.

[Sch99] M. Schoop: A Theoretical Framework for Speech Act Based Negotiation in Electronic Commerce. In S. Klein, B. Schneider (eds), *Negotiations and Interactions in Electronic Markets, Proc. 6th Research Symposium on Emerging Electronic Markets (RSEEM),* Münster, Germany, pp. 79-89, 1999.

[Se69] J. Searle: Speech Acts: An Essay in the Philosophy of Language, Cambridge University Press, Cambridge, 1969.

[SL98] B. Schmid and M. Lindemann: Elements of a Reference Model Electronic Markets. *Proc. 31st Hawaiian Intl. Conf. On System Sciences (HICSS-31),* Hawaii, 1998.

[SQ01] M. Schoop and C. Quix: DOC.COM: Combining document and communication management for negotiation support in business-to-business electronic commerce. In *Proc. 34th Hawaiian Intl. Conf. On System Sciences (HICSS-34),* Maui, 2001.

[SS01] S. Schmitt and B. Schneider: Einsatzpotentiale der KI im Electronic Commerce. *Künstliche Intelligenz,* 1/01:5-11, 2001.

[WRMT95] E. Whitehead, J. Robbins, N. Medvidovic and N. Taylor: Software Architecture: Foundation of a Software Component Marketplace, In *Proc. 1st International Workshop on Architectures for Software Systems,* Seattle WA, April 24-25, 1995, 276-282.