FRAMEWORK ARCHITECTURE ENABLING AN AGENT-BASED INTER-COMPANY INTEGRATION WITH XML

Klement J. Fellner and Klaus Turowski
Otto-von-Guericke-University Magdeburg, Institute for Technical and Business Information Systems, Business Information Systems, P.O. Box 41 20, 39016 Magdeburg, Germany, phone: +49 (391) 67 1 - 83 86, fax: - 12 16, E-mail: {fellner|turowski}@iti.cs.uni-magdeburg.de, URL: http://www-wi.cs.uni-magdeburg.de

ABSTRACT.

More and more cooperating companies utilize the World Wide Web (WWW) to federate and further integrate their heterogeneous business application systems. At the same time, innovative business strategies, like virtual organizations, supply chain management or one-to-one marketing as well as trend-setting competitive strategies, like mass customisation are realisable. Both, the necessary integration and the innovative concepts are demanding software supporting automation of communication as well as coordination across system boundaries. In this paper, we describe a framework architecture for inter-company integration of business processes based on commonly accepted and (partially) standardized concepts and techniques. Further on, it is shown how the framework architecture helps to automate procurement processes and how a cost-saving black-box re-use is achieved following a component-oriented implementation paradigm.

KEYWORDS: Software agents; XML/EDI; UN/EDIFACT; component-orientation

INNOVATIVE BUSINESS STRATEGIES MOTIVATING INTER-COMPANY INTEGRATION

The Internet enjoys great popularity as a universally available communication channel. The World Wide Web (WWW) as its most popular part is changing from a media for information gathering to a platform for distributed (business) application systems. The number of companies offering goods and services via the WWW rises dramatically. Furthermore, companies use the WWW to federate their mostly heterogeneous business application systems (Gaedke & Turowski, 1999).

Under these conditions innovative concepts, like virtual enterprises (Arnold, Faisst, Härtling, & Sieber, 1995), supply chain management (Houlihan, 1992) or one-to-one-marketing (Piller & Schoder, 1999) as well as trend-setting business strategies, like mass customisation (Pine II, 1993) are realizable. Mass customisation, as the combination of the other mentioned concepts, aims at the efficient mass-production of customer-individual products and services (Turowski, 1999a).

Competitive strategies, like mass customisation, imply the existence of state-of-the-art information structures enabling and improving inter-company process integration. This holds especially for small and medium enterprises (SME) participating in a cooperation (or virtual enterprise) following a mass customisation strategy.

Guaranteeing economic and effective communication between enterprises therefore constitutes an essential success factor when following one of the mentioned strategies. One way to improve communication between cooperating enterprises is the utilization of techniques for *electronic data interchange* (EDI). In this paper, we propose an approach how this can be done using standardized and well-known protocols and techniques. Generally, the advantage of EDI lies in the organizational information-based surplus values (Kuhlen, 1996, esp. p. 90), represented by, i.e., improved organization structure, improved processes, or time and cost savings.

There is a rising demand on software that (partially) automates the business processes and that supports exchange of information between potentially incompatible business application systems across the borders of different enterprises. To allow a broad range of application scenarios, such software must feature ease of installation and use, easy integration in an existing application environment, as well as a reasonable price.

In this paper we present a framework architecture enabling inter-company integration based on commonly accepted (and partially standardized) techniques and concepts including preparatory work (Fellner & Turowski, 1999), (Rautenstrauch & Turowski, 1999), (Turowski, 1999a), (Fellner & Bünger, 2000).

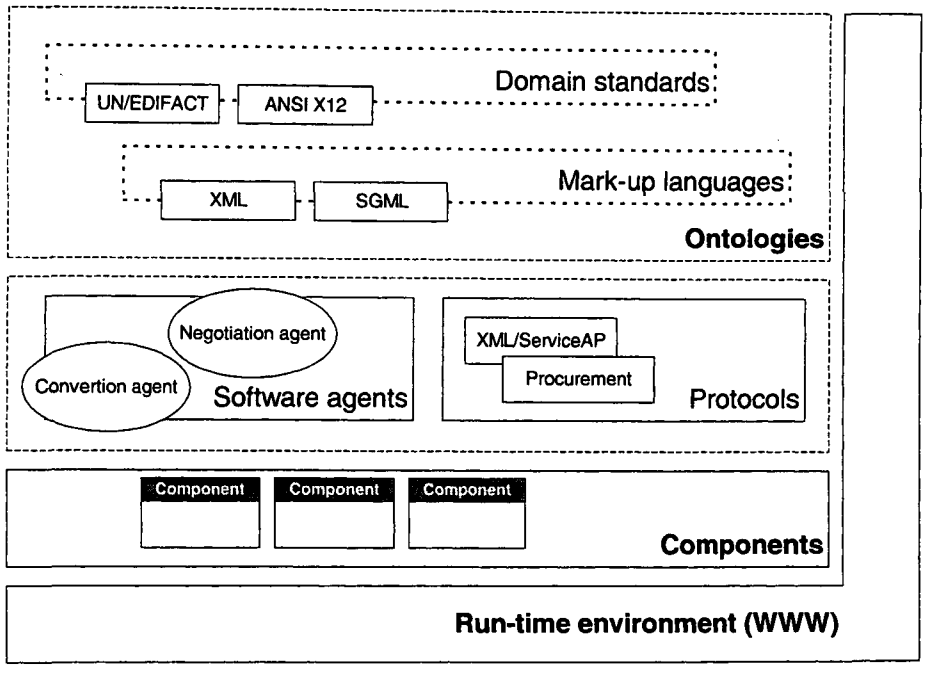

Figure 1: Framework architecture for inter-company integration

The use of a platform independent mark-up language in combination with commonly accepted communication standards breaks down the borders caused by heterogeneous business application systems. (We propose to use a combination of the *Extensible Mark-up Language* (XML) and UN/EDIFACT (*Electronic Data Interchange for Administration, Commerce and Transport*).) In this context, we introduce a (domain-specific) communication protocol for procurement processes, as well as a domain-independent protocol for negotiating communication channels and security demand, e.g. to ensure secure transactions. Software agents use these protocols to fulfil negotiation tasks as well as the conversion of incompatible outputs of business application systems. The software agents are composed from software components. This allows for cost-saving re-use and easy adaptation to company-specific demands. A java-based *component application framework* (Turowski, 1999b, pp. 5-7) for inter-organisational communication and coordination works as agent integration platform and creates agents to support queued (business) tasks. For each procurement task, e.g., the component application framework generates a negotiation agent and a conversion agent. Both agents are removed after the respective procurement task is completed.

ONTOLOGIES AND INTEGRATION WITH MARK-UP LANGUAGES

With UN/EDIFACT (UN, 1995) the United Nations established the best-known standard for inter-organisational electronic data exchange. Due to fundamental drawbacks, the standard did not get the expected recognition and implementation extent, (Zbornik, 1996, pp. 92-93), (Goldfarb & Prescod, 1998, pp. 106-110). Some major drawbacks of UN/EDIFACT are:

- Absence of semantic rules, e.g. for quantity or packaging units.
- Implicit assumption that every organization uses the same business processes and scenarios.
- Economic shortcomings, e.g. high implementation costs, especially for SME.
- Organizational shortcomings, e.g. slow adoption to changing business processes, complicate adjustment of established business processes and rules.

The projects *Open-EDI/object oriented-EDI* (TMWG, 1998), *Universal Data Element Framework* (UDEF) (Harvey et al., 1998), *Basic Semantic Repository* (BSR) and its successor *BEACON* (Steel, 1997) address the first-mentioned problems. Consequently, the focus of these projects lies in the definition of common business scenarios and semantic rules.

In contrast, the *XML/EDI Group* (Peat & Webber, 1997) emphasizes on the possible surplus value induced by the use of XML (Bray, Paoli, & Sperberg-McQueen, 1997). The XML/EDI Group, as well as related efforts (for an overview cf. (Steffen, 2000)), aim at reducing implementation costs and improving flexibility.

The mark-up language XML, a subset of the *Standardized Generalized Mark-up Language* (SGML) (Cover, 2000), offers ways to define individual mark-up elements (tags) as well as document structures that are kept available in so-called *Document Type Definitions* (DTD). A DTD allows the specification of grammar for the syntax on which a data exchange is based. If the identifiers of such a DTD are taken from a standardized format, or a term set upon which cooperating partners agreed on, messages based on this DTD are meaningful for the receiver, because every mark-up element (XML tag) refers directly to underlying semantics of the data.
Figure 2 shows an example message encoded in XML for an enquiry for a bicycle-saddle. The used XML tags, i.e. MESSAGE, PRODUCT, PARTNR, as well as the allowed syntax could be defined in an attached DTD. The enquiry itself contains information regarding the message itself (TYPE, ENQUIRY-DATE) and the description of the demanded product (PART-NR, DESCRIPTION, MATERIAL).

```
<MESSAGE>
  <TYPE>Enquiry</TYPE>
  <ENQUIRY-DATE>20.02.2000</ENQUIRY-DATE>

  <PRODUCT>
    <PART-NR>230-239844-BEZ-531</PART-NR>
    <DESCRIPTION>Bicycle saddle</DESCRIPTION>
    <MATERIAL>Leather</MATERIAL>
    ...
  </PRODUCT>
</MESSAGE>
```

Figure 2: Enquiry encoded in XML

The basic idea behind XML-based EDI is to tag the data with (standardized) XML tags. This is utilized in this paper in combination with a multi-agent system improving the inter-organisational communication and subsequently the coordination of inter-organisational business processes.
We use segment names from the UN/EDIFACT-standard as field descriptors to ensure communication between individual agents of the multi-agent system. The XML tags in figure 2, which were chosen to be readable by humans, will therefore be replaced by UN/EDIFACT segment names.
Figure 4 shows how to replace the ENQUIRY-DATE from figure 2 to be compliant to the proposed combination of XML and UN/EDIFACT. The now standardized message means "This enquiry is from February 20, 2000.". One may say that this can also be seen in the message from figure 2. This holds, if both partners agree on the term ENQUIRY-DATE, but normally one has to communicate with systems or partners not known at the time of definition.
The names used for the XML tags correspond to the UN/EDIFACT definitions. Figure 3 shows an excerpt from the corresponding document. The segment names are depicted in bold letters. High-level segment names are directly used as XML tag-names (DTM). Dependent segment names (2005, 2380 and 2379) are preceded by the corresponding high-level segment name (DTM2005, DTM2380 and DTM2379). Necessary attributes (2005) are presented as parameters of higher-level XML tags. Optional attributes (2380 und 2379) are inserted as stand-alone, subordinate XML tags.

```
DTM DATE/TIME/PERIOD
    To specify date, time, or period.
 ...
 2005 Date/time/period qualifier,              M, an..3
  ...
  137 Document/message date/time, Date/time when a
      document/message is issued.
 ...
 2380 Date/time/period,                        C, an..35
 ...
 2379 Date/time/period format qualifier,       C, an..3
 ...
  102 CCYYMMDD, Calendar date: C = Century; Y = Year;
                               M = Month;   D = Day.
```

Figure 3: UN/EDIFACT segment names

As various projects are on their way to establish repositories for domain-specific XML tags based on UN/EDIFACT, backward compatibility for the chosen naming conventions is guaranteed. The relation

between new terms and the UN/EDIFACT categories can be preserved (Harvey et al., 1998, esp. pp. 10-12). Beyond this, the vocabulary of the agents may be adapted using mapping repositories. For example, an enterprise in the U.S. that uses the ANSI X12 standard may use a EDIFACT/X12-mapper for its agents to be able to understand UN/EDIFACT based XML messages.

```
<DTM DTM2005="137">
  <DTM2380>20000220</DTM2380>
  <DTM2379>102</DTM2379>
</DTM>
```

Figure 4: XML-message based on UN/EDIFACT segment names

SOFTWARE AGENTS SUPPORTING INTER-COMPANY INTEGRATION

Based on the data flow in figure 5 we explain the role of software agents in the framework architecture presented in section 1. The procurement process of a manufacturer will serve as explanatory example. Subsequently we describe the implementation. To keep the presentation as readable as possible, we restrict the example to one supplier.

Figure 5 shows the software agents and application systems on manufacturer's and supplier's side as well as message exchange between them. In common, a (software) agent is defined as an autonomous problem-solving unit that collaborates with other agents to achieve optimised results. For an in-depth discussion of software agents we suggest (Bradshaw, 1997, pp. 4-12). It is assumed that both actors, the manufacturer as well as the supplier, utilize an application system for *production planning and control* (PPC), which, at least, provides an proprietary interface for exporting, respectively importing data in a non-standardized format. In a mass customisation setting, e.g., the manufacturer is responsible to procure the (customer-individual) parts necessary for producing the product configured by the customer. Corresponding enquiries are generated by the PPC system.

The generated enquiry must be transferred automatically to possible suppliers using EDI. Therefore, the output of the PPC system (the enquiry) is transferred to the conversion agent. The conversion agent translates the (proprietary) output of the PPC system into an XML format agreed by all actors. Based on the information regarding the receiver of the message (the supplier), the conversion agent also filters the needed information to meet requirements of the addressee. This part is important to assure that critical information, like participating suppliers, are kept secret. The conversion agent then forwards the translated and filtered information to the addressee – the supplier's software agent. The transfer is carried out using standardized Internet protocols (cf. section 4) enabling the use of existing Internet connections, thereby reducing transfer costs.

On the supplier side, a *listener* (a system part of the *framework component* (cf. figure 7) that also generates agents) waits for incoming requests, and transfers the request (the enquiry) to the corresponding conversion agent. The conversion agent translates the enquiry to the format that is necessary for the local PPC system. Here, only information relevant for the PPC system to generate an offer is translated. This is possible, since each transmitted data item is marked with a standardized or formerly agreed and therefore understandable XML tag categorizing the content (cf. Figure 2 and Figure 4).

The supplier's PPC system (automatically) generates an *offer* based on the terms of delivery given in the enquiry. Then, it transfers the offer to the manufacturer using the same procedure and agents as stated above. This constitutes the general communication between the manufacturer and the supplier in a simple one-to-one-case where all terms of delivery are fixed in advance.

In real world processes, terms of delivery, like date of shipment, quantity, deadline, or price, are most likely subject of negotiation. To include and automate these negotiation processes we propose to use *negotiation agents*. Negotiation agents are especially useful, when more than one supplier is able to deliver the required parts (cf. figure 5). Objects of negotiation are parts that have to be produced in order to assemble a (customer-individual) product. Negotiation agents do not need to know much about these objects. Identification and certain constraints, i.e. due dates or quantities, are sufficient. After negotiation took place, a negotiation agent returns the terms of delivery on which one ore more suppliers agree to deliver. The negotiation process may be carried out in several iterations with alternating offers and counteroffers. A negotiation agent at manufacturer-side, as well as at supplier-side may decide to generate a counteroffer, accept an offer, or abort the negotiation process. In case of a successful negotiation, the results are passed over to the corresponding conversion agents, and subsequently, to the respective PPC system. Last, the PPC system of the manufacturer generates an order for the winning supplier.

The whole negotiation process is carried out utilizing existing conversion agents, thereby allowing extensive re-use of definitions (i.e. XML to PPC mappings) and implementations.

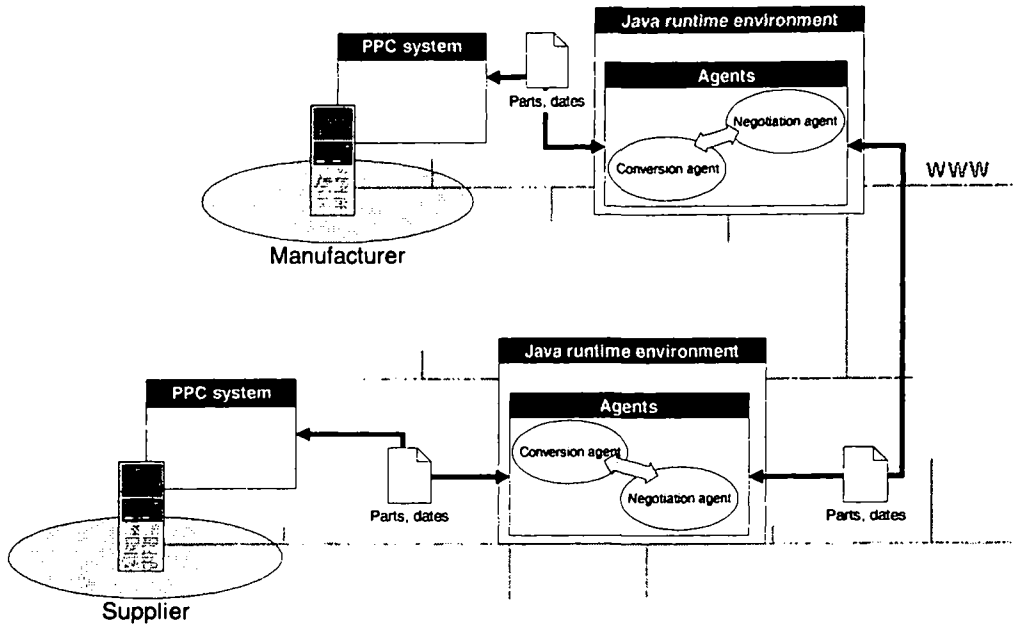

Figure 5: Message exchange for procurement

COMMUNICATION AND SECURITY

Until now we supposed that cooperating enterprises and their software agents are at least technically able to exchange the defined messages. So, those cooperating systems must communicate in a certain way with each other. When looking at real world application systems, this may not hold. Especially business application systems provide mainly proprietary interfaces that must be addressed explicitly. Utilizing our approach of a multi-agent system with an explicit conversion agent may also help to solve this problem.

A publicly known or mutually agreed communication channel must be used by the agent system to establish the first-time communication. No critical business data is transmitted within this communication session. It is used only to negotiate on a *secure* communication channel for further transactions. We use the XML-based *Service Advertising Protocol* (XML/ServiceAP) (Fellner & Bünger, 2000) for negotiation. The XML tags are based on the UN/EDIFACT QUOTE and REQUOTE (REquest for QUOTE) messages. For instance, The enquiry for communication channels is transmitted as XML/ServiceAP-REQUOTE and the corresponding answer is realized as XML/ServiceAP-QUOTE. To determine all possible communication channels between two co-operating PPC systems the initiating PPC system sends an empty XML/ServiceAP request (i.e. `<XMLServiceAP CLIENT='BY'></XMLServiceAP>`). The receiver of this message subsequently delivers the information on all supported channels (see figure 6 for an example of this message). In this scenario the software agents are considered as part of the process initiators, the PPC systems.

Figure 6 shows an example of a PPC system that supports communication via encrypted HTTP (*Hypertext Transfer Protocol*) connections (i.e. the *Secure Socket Layer Protocol* (SSL) (Nusser, 1998, pp. 124-127), the *Java Cryptography Extension* (JCE) (Sun Microsystems, 2000), and the (not encrypted) *Remote Method Invocation* (RMI) (Sun Microsystems, 1998)). At this stage, the XML messages contain information about possible connections only, e.g. the public key for RSA (popular public key technique developed by Rivest, Shamir, and Adelman) encryption (Smith, 1997, pp. 203-218). The connection itself is afterwards established using a suitable *communication component* for the chosen protocol, which will be loaded at run-time (cf. Section 5). The described procedure helps to protect critical data from improper use.

To add additional security to the submission, the channel may be randomly selected for each new connection. In conjunction with a secure first-time connection this helps to prevent potential Internet protocol attacks. The secure first-time connection is important in this context, because all relevant information, like the port, or the used encryption is exchanged. As we now must know something about

the communicating actor's communication protocols, this restricts the utilization of our approach within which we claim the possible communication with unknown suppliers (partners). Possible attacks on the internal information systems may be prevented by allowing only checked (i.e. through certificates) or known partners to connect directly. For further information on general security issues when implementing EDI with Internet protocols we refer to (Stein, 1998) or (Stallings, 1995).

```
<XMLServiceAP CLIENT='BY'>

  <PROTOCOL CAPTION='JCE'>
    JCE://192.168.0.2:7002
    ?PUBLICKEY=b0011cb08d8689aa0...
    +ALGO=RSA
    +AMP=RSA/ECB/PKCS1Padding
  </PROTOCOL>

  <PROTOCOL CAPTION='TCP'>
    TCP://192.168.0.2:7001
  </PROTOCOL>

  <PROTOCOL CAPTION='RMI'>
    rmi://marcelpc:1100/rmiserver0
  </PROTOCOL>

</XMLServiceAP>
```

Figure 6: Example of a PPC system offering different communication channels

Communicating PPC systems have to support communication via at least one open interface (i.e. the *Common Object Request Broker Architecture* (CORBA) (OMG, 1998)). To enable a specific connection, all participating systems must know the specific information for the chosen protocol. Table 1 shows the protocols actually supported by the prototype with the mandatory information.

Protocol / technology	Mandatory information
TCP/IP	• Host-name resp. IP-Address • Port, at which the server process runs
TCP/IP with CE (encryption)	• Hostname resp. IP-Address • Port, at which the server process runs • Public key, encryption algorithm, -methods und –padding
Remote Method Invocation (RMI)	• URL Hostname resp. IP-Address • Port and service provided
Object Request Broker (ORB) without Name-Service	• Host-name resp. IP-Address and • Service provided
ORB without Name-Service	• Reference number for the provided service (Object)

Table 1: Possible Protocols with necessary connection information

The software agents for inter-organisational coordination as well as the agents for negotiating the communication protocols use the *same* information infrastructure. They are implemented as *multi-agent system*. The multi-agent system is based on the *contract net* paradigm (Smith, 1980, p. 1104) and is realized as a *manager/contractor* contract-net, cf. e.g. (Zelewski, 1993, p. 20). An agent represents each actor involved in a contract net. The manager-agent addresses the contractor-agent directly. The contractor-agent in turn answers with an offer that is collected by the manager-agent. The manager-agent subsequently chooses the best among all offers handed in and informs each contractor-agent upon the decision.

In our application the negotiation agent at the manufacturer-side acts as a manager-agent, the negotiation agents at supplier-side act as contractor-agents. As the conversion agents execute the initial distribution and receipt of enquiries and the final order placement, they are part of the manager/contractor-net in a narrower sense. Together with the negotiation agents they fulfil the task of so-called *contract processors* (Kirn, 1996, p. 23).

With the use of XML the contract-net protocol (Albayrak & Bussmann, 1993, p. 61) is easier to set up compared to the standard manager/contract net. It is still necessary that the sequence of messages must be defined, but their content and its arrangement may be a superset of information mandatory for negotiation. This holds, since the negotiation agent only extracts relevant information. Additionally, the use of XML allows extensive, cross-organizational re-use of negotiation agents, as the negotiation protocol may be exchanged or adapted easily. In particular, different negotiation protocols may be served by a superset of transmitted information. On the negative side, this information overload means higher network traffic, because each negotiating partner gets all the information independent from the need of the implemented negotiation algorithm.
If the orders are placed and the business partners agreed on a integration of their business processes the same multi-agent system may be used for the cross-organizational coordination of the production process, i.e. in case of machine breakdown or failure. If a failure occurs in the production process of one participating actor, a negotiation agent may be instantiated with the actual, changed, delivery terms. The negotiation process stays the same as stated above, apart from the fact, that only the affected supplier is contacted. Other suppliers are only contacted, if the negotiation with the actual supplier fails.
To coordinate these inter-organizational production processes the well-known procedures for in-house coordination (Corsten & Gössinger, 1998) are of restricted use. Especially, because prerequisites of these procedures, like the information allocation generally do not hold. Suppliers may, i.e., only provide detailed information on their production plans (i.e. shift plans), when they are very closely connected to the supplier. Furthermore, it may not hold that a solution convenient for the manufacturer causes a positive contribution to the target function of the supplier. In a narrower sense, the multi-agent system therefore does not carry out any planning procedures. Rather, it offers additional coordination means for production plans of PPC systems of cooperating parties.
Another possibility to extend the described approach may be the utilization of alternative variants of the contract-net approach. Following (Zelewski, 1997), the combination of networks for coordination purposes and market mechanisms appears to be promising.

Implementation of the framework component

All different agents are based on the same generalized kind of software agent. With this, the same information infrastructure can be *re-used* for different tasks. The underlying system architecture at the manufacturer-side is depicted in figure 7.
The implemented prototype is written entirely in Java and comprises the individual software agents as well as a rudimentary PPC system for demonstration purpose. The software agents are instantiated and coordinated through an additional software component - the *framework component*. The initialisation of the software agents is done as required by the business process. For instance, the framework component is responsible for instantiating a negotiation agent for each incoming enquiry.
The different software agents are executed within a Java application and need a *Java Runtime Environment* (JRE). As a *Java virtual machine* is available for all popular operating systems used in business environments, this prerequisite does not restrict the potential implementation environments of the proposed approach. Another advantage of Java is that the necessary functionality for inter-agent-communication is already integrated in the Java application programming interface (API) (Sun Microsystems, 1998).
In the following we describe some implementation aspects for each software agent of the presented approach. The agents that convert the output of PPC systems are specific to a respective PPC system. Supporting an additional PPC system implies the adaptation of existing conversion agent, or the implementation of a new conversion agent. Besides this, the only prerequisite for PPC systems to take part in the given scenario is an accessible interface supporting a known output and input format for exchanging data. In a simple case, a conversion agent has to read in a text file created by a PPC system, to extract necessary data, and to create a corresponding XML messages.
On the other hand, it is possible that the conversion agent gathers necessary information directly from the PPC system using remote function calls. For instance, one conversion agent in the prototype uses the *Business Application Programming Interface* (BAPI) to gather data directly out of the business application system SAP R/3 (SAP, 1997). Generally, the needed communication protocols are implemented as stand-alone components and will be loaded at run-time using the Java class-loader facility. This allows the re-use of existing communication components in all agents as well as an easy utilization of the XML/ServiceAP described in section 4.
Like the agents converting the PPC output into XML (PPC-to-XML), the agents responsible for converting XML messages in a PPC system specific format (XML-to-PPC) are specific too. But the

effort for implementing this agents is relatively modest. This is, because the parsing of XML messages can be implemented using freely available components, e.g. the XML-Parsers from IBM (*XML4Java*) or Sun (*XMLParser*), and the mapping of the output can be done based on the implementation of the PPC-to-XML conversion agent.

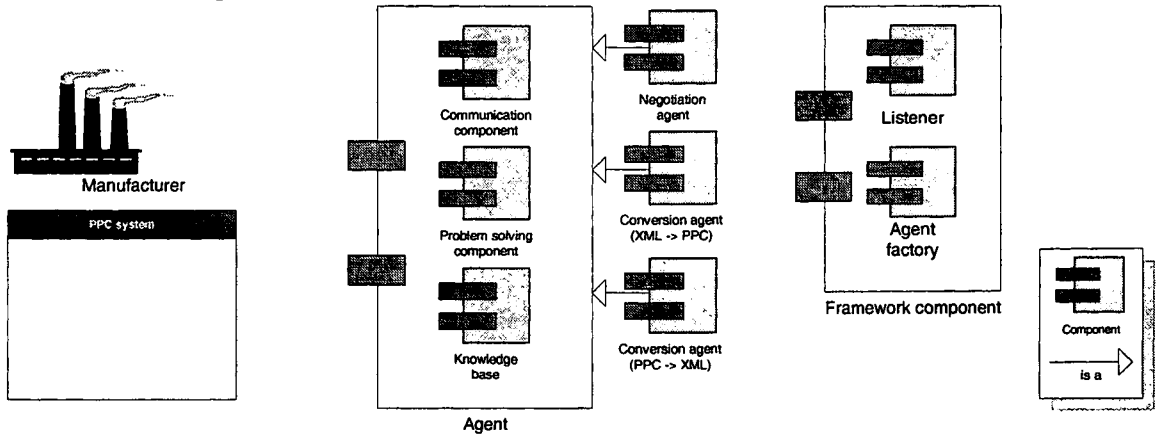

Figure 7: The manufacturer-side system architecture

Besides the conversion agents, negotiation agents are needed in our approach. The implemented negotiation agents are made up of a *knowledge base,* a *problem solver* and a *communication component.* This architectural pattern is used for all software agents in our approach. The conversion agents described above, for example, consist of a knowledge base containing a simple one-to-one mapping of XML tags to PPC system specific entries (i.e. positions in a comma-separated file), a problem solver implementing the mapping algorithm, and the communication component containing a XML parser and the protocols for the communication with the other agents as well as the PPC system (i.e. the BAPI calls of SAP R/3). The main difference between the same components of different agents (i.e. the problem solver) lies in the complexity of the supported task (i.e. mapping data vs. complex preference functions).

The knowledge base of a negotiation agent contains all incoming offers for one enquiry, the enquiry itself and results of a preceding negotiation within a multi-step negotiation process. The data itself is encoded and stored in XML and can be extracted using the *XML Query Language* (XQL) (Robie, Lapp, & Schach, 1998).

The problem solver (on manufacturer-side) is responsible for finding the best offer among all offers that are handed in. When a new offer arrives, relevant information is passed over to the problem solver utilizing the communication component. Subsequently, the problem solver determines the best offer using a preference function. This process stops, when for each enquiry a recline or an offer has been received. Additionally, criteria can be formulated on which the process stops immediately (i.e. time consumption). Afterwards, the implemented preference function is used to choose the best available offer for a given enquiry. The selected offer may also serve as the starting point for a new round of negotiation. For instance, a new enquiry (based on the former on) is send to the supplier with the best offer containing a new (maybe shorter) delivery time. It is important to mention that this procedure can lead to a situation where a formerly positive offer may be withdrawn resulting in the need for a complete new procurement process. After an offer is accepted for an enquiry, the negotiation process stops and the other participants (negotiation agents, suppliers) are informed by the negotiation agent to cancel their offers.

Through strict separation of the different components a negotiation agent may be re-used in similar scenarios. For example, a company with different preference functions for products or group of products will re-use the same negotiation agent by adapting or exchanging the problem solver. As the negotiation agent is initiated with enterprise-specific information the same kind of negotiation agent also may be used at the manufacturer-side as well as the supplier-side.

OUTLOOK AND CONCLUSIONS

Utilising an efficient and effective information infrastructure for inter-organizational integration is a critical success factor for enterprises following state-of-the-art business strategies. The presented framework architecture helps to improve necessary inter-organizational communication and

coordination. The framework architecture is based on commonly accepted, and, where available, standardized techniques and concepts. The combination of XML with established communication standards helps to bridge the gap between heterogeneous business application systems.
The use of open standards, the re-usability of essential software components, and their platform independence allows for an easy adaptation of the framework architecture to enterprise-specific circumstances. Especially small and medium enterprises may profit through lower initial costs.
Above the support of inter-organizational communication, the proposed approach allows a generic re-use of multi-agent systems enabling extensive automation of procurement, coordination of production across organizations, and negotiation on applicable communication protocols and security standards using an identical information infrastructure.
The exploitation of the mentioned techniques may be carried out step by step. For example, after an initial use of software agents to improve communication, a multi-agent system can be set up to automate procurement. Last, a multi-agent system to coordinate the manufacturing may be set on top re-using the existing components.

REFERENCES

Albayrak, S., & Bussmann, S. (1993). Kommunikation und Verhandlungen in Mehragenten-Systemen. In H. J. Müller (Ed.), **Verteilte Künstliche Intelligenz: Methoden und Anwendungen** (pp. 55-81). Mannheim.

Arnold, O., Faisst, W., Härtling, M., & Sieber, P. (1995). Virtuelle Unternehmen als Unternehmenstyp der Zukunft? **HMD**, *32*(185), 8-23.

Bradshaw, J. M. (1997). An Introduction to Software Agents. In J. M. Bradshaw (Ed.), **Software Agents** (pp. 3-46). Menlo Park: AAAI Press.

Bray, T., Paoli, J., & Sperberg-McQueen, C. M. (1997). **Extensible Markup Language (XML).** Available: http://www.w3.org/TR/PR-xml.html [1998, 06-12].

Corsten, H., & Gössinger, R. (1998). Produktionsplanung und -steuerung auf Grundlage von Multiagentensystemen. In H. Corsten & R. Gössinger (Eds.), **Dezentrale Produktionsplanungs- und -steuerungs-Systeme: Eine Einführung in zehn Lektionen** (pp. 174-207). Stuttgart: Kohlhammer.

Cover, R. (2000). **SGML: General Introductions and Overviews.** Oasis-Group. Available: http://www.oasis-open.org/cover/general.html [2000, 02-30].

Fellner, K., & Turowski, K. (1999). Component Framework Supporting Inter-company Cooperation. Paper presented at the Proceedings 1999 **Third International Enterprise Distributed Object Computing Conference (EDOC'99)**, Mannheim.

Fellner, K. J., & Bünger, M. (2000). Basistechnologie zum betriebsübergreifenden Austausch von Umweltinformationen – Fachliche und technische Abstimmung. Paper presented at **the BUIS 2000 - Betriebliche Umweltinformationssysteme**, Olten (CH).

Gaedke, M., & Turowski, K. (1999). Generic Web-Based Federation of Business Application Systems for E-Commerce Applications. Paper presented at the **Second International Workshop on Engineering Federated Information Systems (EFIS'99)**, Kühlungsborn.

Goldfarb, C. F., & Prescod, P. (1998). **The XML Handbook.** Upper Saddle River: Prentice-Hall.

Harvey, B., Hill, D., Schuldt, R., Bryan, M., Thayer, W., Raman, D., & Webber, D. (1998). **Position Statement on Global Repositories for XML.** Available: ftp://www.eccnet.com/pub/xmledi/repos710.zip [1998, 12-01].

Houlihan, J. B. (1992). International Supply Chain Management. In M. Christopher (Ed.), **Logistics - The Strategic Issues** (pp. 140-159). London.

Kirn, S. (1996). Kooperativ - Intelligente Software. **Information Management**, *11*(1), 18-28.

Kuhlen, R. (1996). **Informationsmarkt: Chancen und Risiken der Kommerzialisierung von Wissen.** (2 ed.). Konstanz: Universitätsverlag Konstanz.

Nusser, S. (1998). **Sicherheitskonzepte im WWW.** Berlin: Springer.

OMG (Ed.). (1998). **The Common Object Request Broker: Architecture and Specification (Revision 2.2)**: OMG.

Peat, B., & Webber, D. (1997). **Introducing XML/EDI: "The E-business Framework".** Available: http://www.geocities.com/WallStreet/Floor/5815/start.htm [1998, 12-01].

Piller, F., & Schoder, D. (1999). Mass Customization und Electronic Commerce: Eine empirische Einschätzung zur Umsetzung in deutschen Unternehmen. **ZfB**, *69*(10), 1111-1136.

Pine II, J. B. (1993). **Mass Customization: The New Frontier in Business Competition.** Boston: Harvard Business School Press.

Rautenstrauch, C., & Turowski, K. (1999). A Virtual Enterprise Model for Mass Customization. Paper presented at the **Second World Manufacturing Congress (WMC'99), International Symposium on Manufacturing Systems (ISMS'99)**, Durham.

Robie, J., Lapp, J., & Schach, D. (1998). **XML Query Language (XQL)**. Available: http://www.w3.org/TandS/QL/QL98/pp/xql.html [1999, 01-01].

SAP (Ed.). (1997). **BAPIs - Einführung und Überblick**. Walldorf: SAP.

Smith, R. E. (1997). **Internet Cryptography**. Reading, Massachusetts: Addison Wesley Longman.

Smith, R. G. (1980). The Contract Net Protocol: High Level Communication and Controll in a Distributed Problem Solver. **IEEE Transactions an Computers,** *29*, 1104-1113.

Stallings, W. (1995). **Network and Internetwork Security**. Upper Saddle River, New Jersey: Prentice Hall.

Steel, K. (1997). **The Beacon User's Guide: Open Standards for Business Systems.** Available: http://www.cs.mu.oz.au/research/icaris/beaug1.doc [1998, 12-01].

Steffen, T. (2000). Internet-Quellen zu XML/EDI. **Wirtschaftsinformatik,** *42*(1), 78-86.

Stein, L. D. (1998). *Web Security*. Reading, Massachusetts: Addison Wesley Longman.

Sun Microsystems (Ed.). (1998). **JDK 1.1.6 Documentation - Java Development Kit**. Mountain View: Sun Microsystems.

Sun Microsystems (Ed.). (2000). **Java Cryptography Extension**. Mountain View: Sun Microsystems.

TMWG. (1998). *Reference Guide:* **"The Next Generation of UN/EDIFACT": An Open- EDI Approach Using UML Models & OOT (Revision 12).** Available: http://www.harbinger.com/resource/klaus/tmwg/TM010R1.PDF [1998, 12-01].

Turowski, K. (1999a). Agenten-gestützte Informationslogistik für Mass Customization. In H. Kopfer & C. Bierwirth (Eds.), **Logistik Management - Intelligente I+K Technologien** (pp. 199-209). Berlin: Springer.

Turowski, K. (1999b). Ordnungsrahmen für komponentenbasierte betriebliche Anwendungssysteme. Paper presented at the **Tagungsband des 1. Workshops Komponentenorientierte betriebliche Anwendungssysteme (WKBA 1)**, Magdeburg.

UN. (1995). **United Nations Directiories for Electronic Data Interchange for Administration, Commerce and Transport**. Available: http://www.unece.org/trade/untdid/Welcome.html [1998, 12-01].

Zbornik, S. (1996). **Elektronische Märkte, elektronische Hierarchien und elektronische Netzwerke: Koordination des wirtschaftlichen Leistungsaustausches durch Mehrwertdienste auf der Basis von EDI und offenen Kommunikationssystemen, diskutiert am Beispiel der Elektronikindustrie.** Konstanz: Universitätsverlag Konstanz.

Zelewski, S. (1993). **Multi-Agenten-Systeme für Prozeßkoordinierung in komplexen Produktionssystemen. Ein verteiltes Problemlösungskonzept auf der Basis von Kontraktnetzen** (Arbeitsberichte des Seminars für Allgemeine Betriebswirtschaftslehre, Industriebetriebslehre und Produktionswirtschafts, Arbeitsbericht 46). Köln: Universität zu Köln.

Zelewski, S. (1997). Elektronische Märkte zur Prozeßkoordinierung in Produktionsnetzwerken. **Wirtschaftsinformatik**, *39*(3), 231-243.