IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITB12619

This paper appears in the book, *Emerging Trends and Challenges in Information Technology Management, Volume 1 and Volume 2* edited by Mehdi Khosrow-Pour

Distribution of ERP System Components and Security Considerations

Nico Brehm & Jorge Marx Gómez,
Carl von Ossietzky University Oldenburg, Ammerländer Heerstrasse 114-118, 26129 Oldenburg, Germany,
{brehm, marx-gomez}@wi-ol.de

ABSTRACT

As enterprise resource planning (ERP) systems become more complex the financial expenditures that are connected to the application of such systems dramatically increase. ERP systems consist of many software components which provide specific functionality. However, these ERP systems are designed as an all-in-one solution, often implementing functionality not needed. Furthermore, such ERP systems depend on very large-scale infrastructures like servers and networking technology, which are very expensive to install and to maintain. The new idea is to develop a novel ERP system architecture which facilitates an overall reusability of individual business components (BC) through a shared and NON-monolithic architecture based on a peer-to-peer (P2P) network. In this paper we present an architecture approach which uses secure Web Services to wrap ERP components that are provided by a distributed system which appears as an ERP community. This system architecture opens up new vistas for small- and medium sized enterprises that commonly cannot afford the application of conventional ERP systems.

INTRODUCTION

Modern ERP systems consist of many software components which are related to each other. Currently these components are administered on a central application server. In connection to the ERP system complexity several problems appear:

- Not all installed components are needed.
- High-end computer hardware is required.
- Customizing is expensive.

Due to the expensive proceedings of installation and maintenance only large enterprises can afford such complex ERP systems. One solution to face these problems is to develop a distributed ERP system where the system components are reachable over a network (e.g. internet). This component ensemble (federated system) still appears as single ERP system to the user, however it consists of different independent elements which exist on different computers. Based on this construction it is possible for an enterprise to access on-demand functionality (components) as services of other network members over a P2P network. This approach solves the mentioned problems as follows:

- Due to the separation of local and remote functions, no local resources are wasted for unnecessary components.
- Single components are executable on small computers.
- Due to decreasing complexity of the local system also installation and maintenance costs subside.

As a result of these (cost) advantages ERP systems of the specified kind would open up new vistas to small- and medium-sized enterprises, which require the same functionality and scalability as large enterprises [1, p. 477]. Figure 1 shows our estimation of how the financial expenditures will subside when federated system is used whereas different enterprises share ERP components that are comparable to the components of conventional ERP systems regarding their functionality.

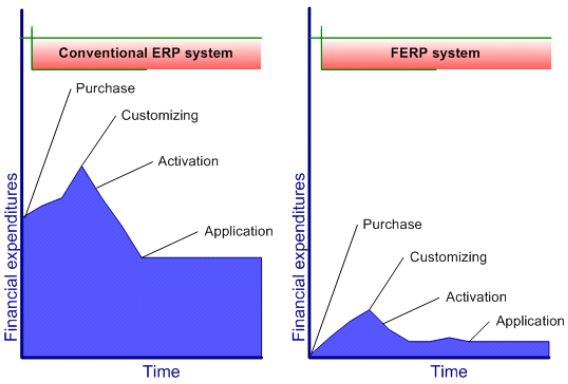
A conventional ERP system causes following costs:

- *Purchase*: A complex standard software system has to be bought which contains more components than the enterprise will need. Furthermore high-end hardware is required.
- *Customizing*: The ERP system must be customized which means that a lot of parameters of a complex software system have to be configured by specialists.
- *Activation*: After quality assurance all configurations must be transferred to the production system. This process must be supervised by specialists. Besides this, members of staff have to be trained.
- *Application*: During the application of the production system, administrators are needed to supervise and reconfigure the running system. Costs thereby incurred depend on network- and hardware resources to be kept available as well as the staff needed to administer the system. The more complex the ERP system is the more time employees will need to accustom themselves to it.

Whereas an FERP system causes following costs:

- *Purchase*: A minimal standard ERP system has to be bought which contains only these basic functions every enterprise needs e.g. a database system, functions for master data administration and a graphical user interface. Compared to a conventional ERP system the minimal installation is less complex whereby less hardware is required for the subsequent application.
- *Customizing*: Assumed that all required ERP components will be available within the ERP-P2P network the customizing process is restricted to only those ERP components the enterprise is supposed to need. Compared to the complexity of conventional ERP systems and their amount of components the FERP system is less complex from the point of view of the utilizing enterprise. Due to reduced complexity requirements of customizing specialists subside and costs are lowered.

***Figure 1**. Expected cost differences between conventional ERP systems and FERP systems*



- *Activation*: The customized ERP system must be connected to the ERP-P2P network which provides all ERP components as services. In this phase the enterprise policies regarding security, quality of service and accounting must be configured as to subsequently meet the enterprises requirements when a service is utilised during the application phase. Although conventional ERP systems do not necessitate this step the FERP system costs will be less because less system elements have to be supervised when the FERP system goes productive.
- *Application*: From the point of view of the enterprise all further costs are connected to the administration of a comparatively small hardware system, the utilization of services which must be paid and the training of employees concerning a less complex system.

Enhancing this approach a lot of research and technique level problems accrue. If the goal is to connect different enterprises to one single ERP system, a characteristic issue is standardizing and disclosure of the underlying ERP system architecture. The application of eXtensible Markup Language (XML) standards offers an appropriate background to start up in this area, because XML, XML namespaces and XML schemas provide useful mechanisms to deal with structured extensibility in a distributed environment [22]. Besides, in connection with the common use of distributed applications, several *security problems* exist. The most important security objectives in the case of distributed ERP systems are:

- *Resource protection*
- *Confidentiality* of transmitted data
- *Integrity* of transmitted data
- *Authenticity* of communication partners
- *Anonymity* of communication partners against unauthorized parties
- *Non-repudiation* of transactions
- *Reliability* (trustability) of communication partners

RESOURCE SHARING IN AN ERP NETWORK

Distributing the software components of an ERP system on different servers some advantages arise for the operators of each of these servers, because hardware demands are made to only their part of the total ERP system. This reduction of complexity facilitates administration and availability securing because all measures are confined to only one ERP peer whereas conventional ERP system operation is geared to the provision of all components at the same time. Figure 2 shows these two approaches in comparison to each other. The left hand side represents the architecture of a conventional ERP system where a closed amount of ERP components (C1, C2, …, C8) are installed on the same application server. The right hand side shows an open ERP network where each node is assigned to one ERP component which is provided as service (S1, S2, …, Sn). This P2P network consists of allied network nodes that all together represent a federated ERP system. New components are added as new ERP peers that provide corresponding services.

As explained above, the distribution of the ERP system is based on a P2P architecture. Each peer can communicate with all the rest of the participating network nodes. Among other forms of P2P structuring, the illustrations below use a pure P2P architecture whereas the integration of a centralized control is abandoned. The assets and drawbacks of this method shall receive no further consideration in this first instance. The duties and responsibilities of every network node are divided into two sections. On one hand the service providing peers and on the other hand the ones which utilize theses services establish the basis for exchanging software components, whereas the over-all system-functionality will be available to the whole ERP network.

Figure 3 shows how the ERP components are distributed within the P2P network. In this example it is also possible that one peer can encapsulate more than one ERP component. Likewise the possibility exists, that a peer does not offer components on its own but even so it can access all provided services of the whole network and by this the whole ERP functionality is available to it. The total charges and costs of the ERP system are averaged to all ERP network nodes, which means that each node saves expenses.

Assumed that the amount of data (D) of the code of each component (C) of the whole ERP system (C1, C2, …, Cn) is equal and each component has the size of 100 kByte, then a storage capacity of D*n (100 kByte*n) is necessary to store the code of the whole ERP system. Besides persistent memory requirements, e.g. as hard disk space, all of the components and their functions must be available as running processes within the application server. This leads to the conclusion that the proposed component distribution to several servers has three main advantages:

- Every ERP server stores only a part of the total system.
- The open ERP network architecture allows the provision of more ERP components than only one server does.
- New ERP system versions arise by adding new components and peers to the network. New versions are available at once.

Perspectively the proposed distributed architecture paves the way to an *overall ERP system* that includes all imaginable functionality whereas the functions are provided as Web services. Although there are differences between Web services and business components a lot of similarities exist that argue for an equalisation of both technologies [2, p.1 ff]. Solutions and models for

- standardising,
- development,

Figure 2: *Conventional ERP system with ERP components (C) versus a federated ERP system that provides its ERP components as services (S)*

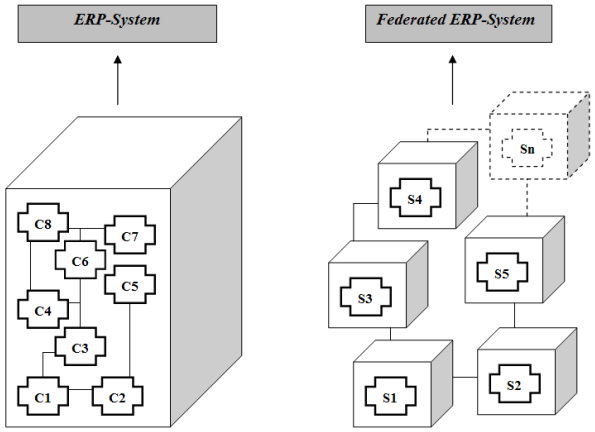
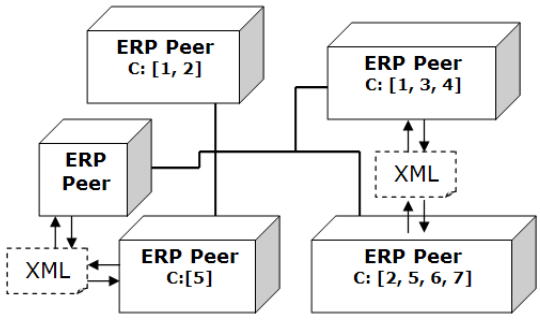


Figure 3: *ERP peers provide various ERP components (C) in a P2P network based on Web Services*



- customisation
- and composition

of business components [3] cacan be adapted if we assume that the encapsulation of ERP functions as business components equals the provision of these functions as superior Web Services.

A SERVICE-ORIENTED ERP SYSTEM

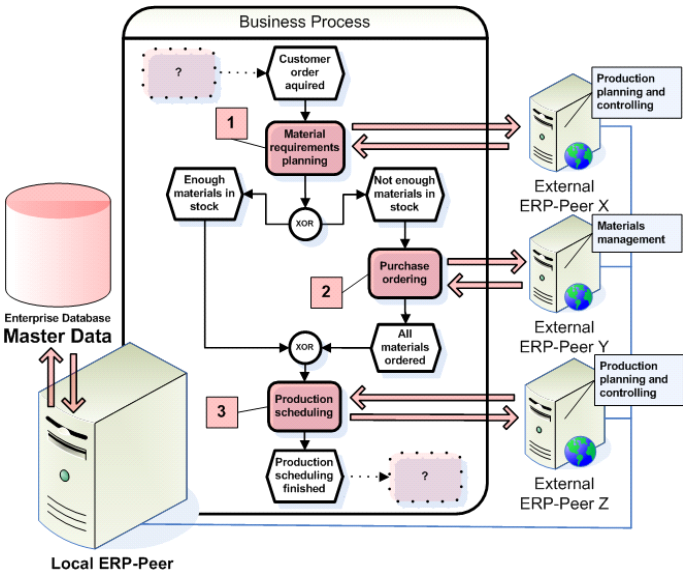
This proposal aims at outsourcing of system functions to other network nodes. The distributed ERP system is based on a service-oriented architecture whereby all functions are available as services and unambiguously addressable. This system works as follows:

1. The network consists of service consuming and service providing network nodes.
2. Each client, which provides an interface to an enterprise is called ***mandator*** and is connected to the enterprise database
3. The processing steps of a business process are stored in the local database of a mandator as *workflow*. A workflow in this context is a plan of sequentially or in parallel chained functions as working steps in the meaning of activities which lead to the creation or utilization of business benefits.
4. Finding a function within the P2P-network means that a request which contains the function type must be send to all service providing peers.
5. After receiving the responses to a function type request the mandator must elect a network node to be accessed.
6. A function call contains parameters as business objects and other (primitive) values that are delivered to the service providing network node. A business object in this context is a snapshot of the enterprise database at a particular time in a standardized format. Function calls can contain other function calls.
7. A function returns a list of either directly modified business objects or independent values that are necessary for subsequent business object updates (e.g. intermediate data)
8. Returned business objects must be synchronized with the local database of the mandator

EXAMPLE

Below we chose a manufacturing company to exemplify the distribution of ERP system functions. Figure 4 shows a typical example of a manufacturing business process as Event Process Cain (EPC) and in connection to this an opportunity of how internal functions can be processed by other ERP peers.

Figure 4*: Example manufacturing business process and outsourcing of ERP-internal functions*



The example in figure 4 shows a highly simplified work process of a manufacturing company starting with the acceptance of the customer order until the completion of the production scheduling. The process contains three functions which are each supposed to be executed by external nodes of the P2P network.

- *Function 1*: After a customer placed a production order, it hast to be examined whether there is sufficient material in stock. The necessary input data which must be sent to the external network node includes all Bills of Material (BOM) of the product to be produced and information about the stock of inventory as regards all individual parts. On this basis the function calculates whether production scheduling can start directly or whether anymore material has to be ordered at first. Finally the external function returns a respective information message back to the local business process.
- *Function 2*: If there is not enough material in stock the second function determines the missing material and sends material orders to the corresponding suppliers. The input values are BOM, stocks of inventory and supplier information. After the demand of material is calculated material orders are sent to the suppliers automatically. For simplification purposes we assume that the suppliers will deliver all materials in time without considering possible problems in this context. The return value of this external function is complex object which contains all executed material orders and the related delivery times.
- *Function 3*: When there is enough material in stock production is supposed to start. Therefore the third function creates all needed data records, e.g. optimized time schedules and work plans, and wraps them as a complex object which will be returned back to the requesting ERP peer. Among other things BOM, work places, capacities and production calendar data are required as input values of this function.

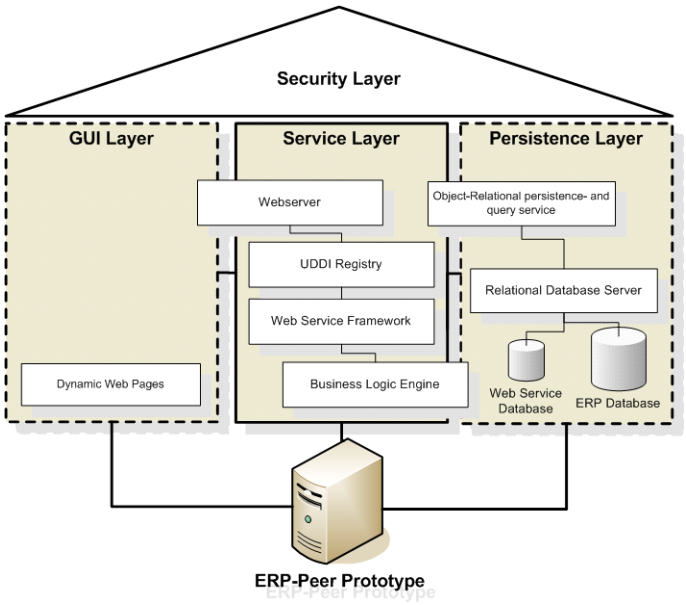
The local peer in this example wraps the required data as business objects which are included in the remote function calls. Complex return objects contain secondary master data records that are inserted into the local enterprise database. The example shows how single functions of a business process are executed on external ERP peers and how a local ERP system benefits from using external business logic, e.g. by using optimisation algorithms.

PROTOTYPE

Following the ideas of our motivation a first prototype was developed which is based on open source software components and open standards. As visible in figure 5 the architecture of this prototype consists of four layers:

- *Persistence Layer:* This layer contains the central enterprise database server which includes two databases. On one hand the enterprise database stores the enterprise master data whereas on the other hand the Web Service database builds a basis for the UDDI registry. For this purposes, our prototype uses a MySQL server [8]. Above this database server we introduced an object-relational persistence and query service which undertakes the task of mapping data of the relational database to objects of the applied programming language. The mapping framework we use is the open source product Hibernate [9].
- *Service Layer:* This layer processes the functionality of the ERP system. Workflows are controlled by the Business Logic Engine which is connected to the enterprise database. This first prototype version uses Java as programming language so that a *Java Virtual Machine* executes business logic functions as Java classes. A further version is supposed to have a Workflow Management System included. Above this system part a Web Service framework exists, which is the basis for providing ERP functions to

Figure 5: ERP-Peer Prototype with four different layers

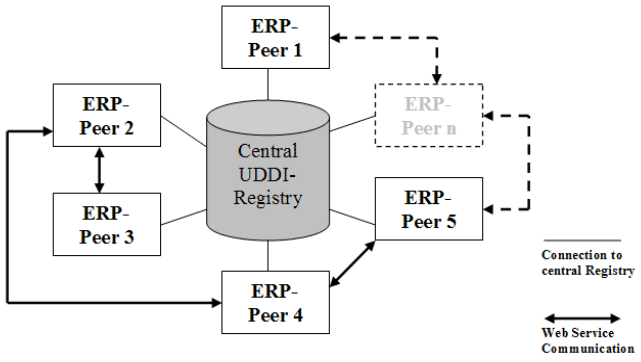


external peers and at the same time permits access to functions running on external peers. Answering this purpose the prototype uses the open source product *Apache AXIS* [10]. Those Web Services which are available for external network nodes are referenced by the peer own UDDI registry. For this we integrated the open source product *jUDDI* [11] of the Apache Software Foundation. Finally a Webserver provides an interface to external ERP peers and to the user (GUI). The whole communication with the outside network is based on HTTP. The used Webserver and Java-Servlet-engine is *Apache Tomcat* [12] which is available as open source product, too.

- *GUI Layer:* The interaction with the end user is controlled by a Web browser. The graphical user interface is based on dynamic HTML pages, which are generated by utilizing Java Servlet Technology [13]. When the interaction with an end user becomes necessary a web site will be generated and sent to the browser of this user.
- *Security Layer:* As to realize individual security objectives of ERP peers which are assigned to a certain enterprises, necessary security mechanisms stretch across all already mentioned architectural layers. The peer-internal communication protection applies the Secure Sockets Layer (SSL) protocol, which means that both, the communication between Webbrowser and Webserver as well as interaction between application server and database server are secured by using this standard. Whereas regarding authentication within the ERP-P2P network the Username Token Profile 1.0 [14] of the Web Service Security specification [15] is applied. For this the prototype includes Apache WSS4J [24] as open source product. Measures regarding confidentiality utilize methods of Public Key Infrastructures (PKI) whereas XML-Encryption [16] is used as standard to apply asymmetric encryption functions like DSA or RSA. The open source project Apache XML Security [17] of the Apache Software Foundation provides a free Java library which was included in our first version of the prototype.

The GUI-Layer and the Persistence Layer can be seen as optional because these layers are not necessary when a network node only has to provide ERP functions for external peers and no peer-own master data has to be stored.

Figure 6: Central management of all Web services in a P2P network where the ERP peers communicate independently with each other



CENTRALIZED AND DECENTRALIZED UDDI REGISTRIES

One of the basic considerations of an ERP-P2P network is that the network has to feature Web service search functions in order to enable the ERP peers to localize the needed functionality. The conventional approach is based on the integration of a central UDDI-registry1, which includes all information about which servers provide which Web services [4]. Figure 6 applies this model to a distributed ERP system.

The approach of figure 6 is also called brokered P2P, where peers connect to a server in order to discover other peers, but then manage the communication themselves [5, p. 29]. ERP peers that offer a Web service have to send this information to the central UDDI-registry for publishing purposes. In order to search for a Web service, ERP peers submit a UDDI-request that includes an unambiguous identification of the needed service. However this solution has two disadvantages:

1. Management and organisation are centralized whereas the central UDDI-registry is a single point of failure.
2. Scalability is limited to the maximum capacity of the central instance.

The second approach abandons the integration of a central registry and is based on a completely decentralized solution where the peers run independently without the need of centralized services. This approach is also called pure P2P. The discovery is decentralized and the communication takes place directly between the peers [5, p. 29]. Figure 7 shows, how this decentralized P2P system can be applied to a distributed ERP system.

The decentralized Web service management is based on the idea that every ERP peer publishes its own Web services through its local UDDI-

Figure 7: Decentralized management of Web services in an open network and independent communication between ERP peers

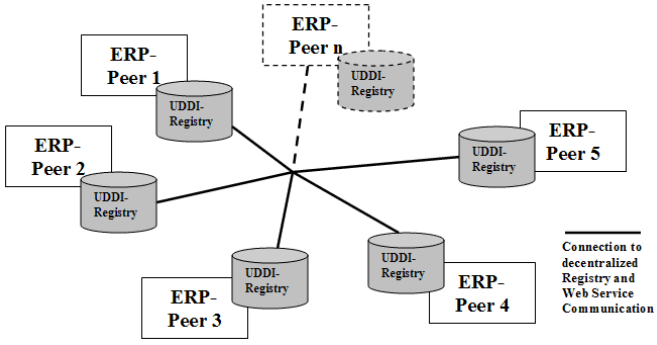
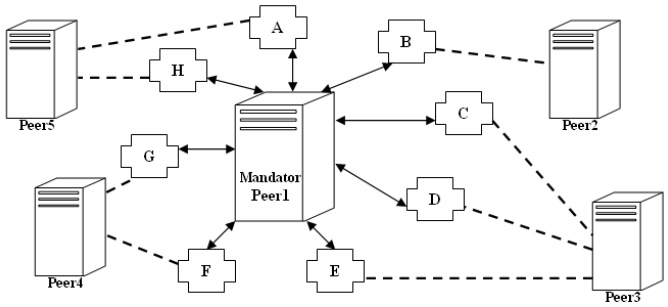


Figure 8: *Outsourcing of functions (services) and central call-controlling by the mandatory*



registry and acts as service provider and as service broker at the same time [6]. The distributed search for a Web service is based on the following procedure (simplified version without considering a break condition):

1. Receive a UDDI search request
2. If the requested Web service is locally available then send a search response to the requester
3. Forward the search request to all known neighbours

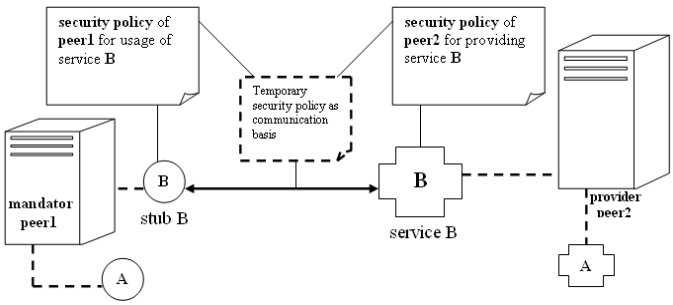
In order use Web services in a completely decentralized network or to offer Web services, it is enough to know only one neighbouring ERP peer because all search requests are forwarded through this neighbour peer.

SECURITY AND POLICIES

During the automation of all steps to integrate Web Services as ERP components it is difficult to automate the decision making process. The decision to use a Web Service is not only affected by the existence or non-existence of the Web Service. Strategic enterprise objectives flow into the decision making process. Qualitative Web Service features must be transparent for service occupants. Particularly interesting knowledge in the context of ERP distribution are security characteristics and their provableness. The proposed approach uses *policies* to express the requirements of the participating communication partners (service occupant and service provider). Policies build up the basis to shortlist providers and help to decide which Web Services are supposed to be used.

As explained above a business process is controlled through a workflow which is managed by the mandator. Figure 8 shows how the mandator controls all function calls. Executing the business process means to sequentially or in parallel call services (functions) provided by other ERP peers. Services deliver return values the mandator can process before a subsequent service is used. A function call is completely executed on only one peer so that there are no direct relationships between different functions on different peers.

Figure 9: *Negotiation of a common security policy*



As explained above it is necessary to negotiate policies which are based on quality properties, before the usage of services (function calls) of other ERP peers is possible. Security characteristics are one of the most important issues in this connection. The WS-Security specification [15] provides a way to protect the integrity and confidentiality of messages and to implement authentication and authorization models in Web Services [18, p. 123]. Furthermore the reassurance of response times in the scope of service level agreements (SLA) could be another subject of such a policy. In this context the main focus is set on security policies. Figure 9 clarifies how the negotiation of a security policy is processed.

Example 1: Combination of two security policies of two communication partners to one common (temporarily) security policy

Security policy of the mandator:

```
<wsp: Policy wsu:Id=”Service_B_Requester_Policy_Peer1”>
        <wsp:ExacltyOne>
                <wsp:All>
<nsSecurityAssertion_1 level=”1” algorithm=”X”/>
                        <nsSecurityAssertion_2 level=”2”/>
                        <nsSecurityAssertion_3 level=”4”/>
                </wsp:All>
        </wsp:ExacltyOne>
</wsp: Policy>
```

Security policy of the Provider:

```
<wsp: Policy wsu:Id=”Service_B_Provider_Policy_Peer2”>
        <wsp:ExacltyOne>
                <wsp:All>
<nsSecurityAssertion_1 level=”1” algorithm=”X”/>
                        <nsSecurityAssertion_2 level=”1”/>
                </wsp:All>
                <wsp:All>
<nsSecurityAssertion_8 level=”9” algorithm=”Y”/>
                        <nsSecurityAssertion_2 level=”2”/>
                </wsp:All>
                <wsp:All>
                        <nsSecurityAssertion_3 level=”3”/>
                </wsp:All>
        </wsp:ExacltyOne>
</wsp: Policy>
```

Temporary common security policy as basis for communication:

```
<wsp: Policy>
        <wsp:ExacltyOne>
                <wsp:All>
<nsSecurityAssertion_1 level=”1” algorithm=”X”/>
                        <nsSecurityAssertion_2 level=”1”/>
                        <nsSecurityAssertion_2 level=”2”/>
                        <nsSecurityAssertion_3 level=”4”/>
                </wsp:All>
                <wsp:All>
<nsSecurityAssertion_1 level=”1” algorithm=”X”/>
```

```
                    <nsSecurityAssertion_2 level="2"/>
                    <nsSecurityAssertion_3 level="4"/>
                    <nsSecurityAssertion_3 level="3"/>
                </wsp:All>
            </wsp:ExacltyOne>
</wsp: Policy>
```

Example 1 shows a negotiated policy which provides the communication basis of the two participating ERP peers. It is a precondition that both sides can interpret and realize the meaning of the policy. WS-Policy provides a method (intersection) which can be utilized for the negotiation process. Intersection is the process of comparing two Web Services for common alternatives. An interaction can take place when both sides agree on at least one alternative [19]. A policy alternative is a combination of elements (e.g. rules or instructions) of the superior policy that must not be separated which means that for example all rules have to be followed. The intersection method delivers zero or more policy alternatives.

To implement and realize the subjects of security policies an additional tier is integrated into the explained ERP peer architecture. Outgoing messages are converted into policy-conform messages. Incoming messages are inspected and checked up on conformance and either forwarded or rejected. Similar to a Web Service description (in WSDL) which is referenced by the "domestic" UDDI-registry it is possible to retrieve the associated Web Service security profile (security policy). However, not only the respective security policy must be part of a WS-policy profile. A list of all supported security mechanisms and standards and important properties that bear upon system security are imaginable. For instance a trustworthy system environment according to the Trusted Computing Group Software Stack Specification (TSS) [20] could be nominated by including respective property entries into a security profile in order to increase the degree of trust on the side of the service consumer.

***Figure 10**: Selection of an adequate provider by comparing the different common security policies*

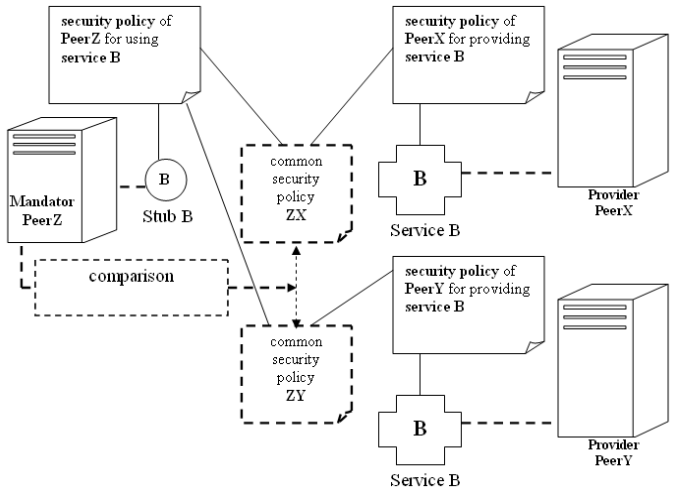
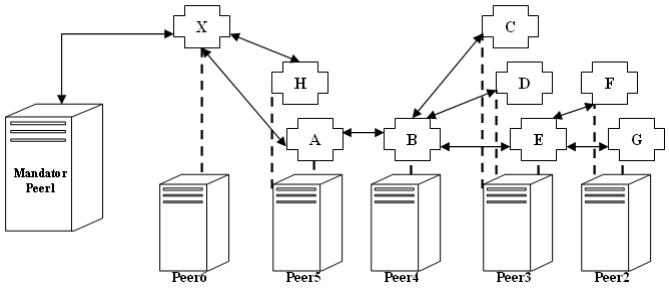


***Figure 11:** Nested function calls without a central control instance*



Policy-based provider selection

Besides the service existence and the possibility to discover a service the security policy of a service-providing ERP peer establishes a further basis to decide whether the service will be used or not. Figure 10 shows how the mandator compares the security policies of two providers. After applying the intersection method the common security policy contains either none or more alternatives which act as selection criteria. When there are no alternatives encountered, which means that the security policy is empty, the interaction cannot take place. In case of multiple alternatives being available an additional internal cost calculation can flow into the decision making process.

So far we assumed that the security policy to be negotiated between service consumer and service provider correspond to the commitment that each service must not be related to services on other peers. In case of intermediate result calculation by third parties all security policies of all computation participants have to be considered. Figure 11 shows a scenario of nested function calls without a central control instance.

As illustrated in figure 11 it is not enough to build up secure connections to the provider, when a hierarchic processing is executed, because the provider arbitrarily sources computation parts out and consequently forwards confidential data. Considering reutilization of the original mandator policy as a way to safeguard the interests and rules of the mandator within the overall network implicates additional problems. Granted, that the provider takes the original mandator policy as basis to negotiate temporary security policies in order to communicate with other network peers for outsourcing purposes, all applied cryptographic functions like encryption- und sign functions are the same. However the question is whether the application of these functions still makes sense.

The degree of trust must be considered when (temporary) common security policies are negotiated because the importance of a policy subject depends on the counterpart's identity. Even if a trust model (e.g. to quantify and consider the network wide reputation of a provider) is adopted, the problem still remains when the provider arbitrarily forwards function calls. Our proposal is that a client (e.g. mandator) must classify a ***provider type*** and correlate it with the explained security policy so that the provider can differentiate between allowed and not allowed sub-providers. The WS-Trust specification [21] defines mechanisms to validate provided tokens and to assess their level of trust. The specification defines extensions that build on WS-Security [15] to provide a framework for requesting and issuing security tokens and to broker trust relationships.

CONCLUSIONS AND OUTLOOK

Comparing distributed ERP systems and ERP systems running on only one computer, the distributed systems offer a lot of advantages. Particularly small- and medium sized Enterprises (SMB) benefit from using shared resources. However, the design and development of a system architecture is subject to a number of risks. Because ERP systems process confidential enterprise data, security considerations play an important role when an open network (e.g. the internet) is used. This approach proposes to distribute ERP systems by sourcing out functions as services in a peer-to-peer network in order to use the resources of allied ERP nodes. Standardized Web Services are used as basic technology. Referring to Web Service Security (WSS) and WS-Policy it is possible to negotiate common policies as body of rules and regulations for the communication between service consumer and service provider in order to meet the

security requirements of both parties. The proposed approach highlights the implication of trust relationships as a common problem. Mainly when hierarchic processing of service requests is allowed the application of the WS-Policy standard is not satisfactory which means that appropriate enhancements are necessary. Third party certification of service providers, e.g. by trusted domains or trusted realms as specified in the WS-Trust and WS-Federation [23] standards, can contribute to solve the trust-related problems, however it makes the integration of new services more complicated. Arbitrating between theoretical trust models, their technical practicability and the potential acceptance must build up the basis for further research work.

REFERENCES

Kurbel, K.; Rautenstrauch, C. (1989): Ein verteiltes PPS-System auf Arbeitsplatzbasis, München, GI-19.Jahrestagung II Computer-gestützter Arbeitsplatz, Springer-Verlag

Krammer, A., Turowski, K. (2001): Spezifikationabedarf von Web-Services. In: Ortner, E., Overhage, S. (Hrsg.): 1. Workshop „Entwicklung von Anwendungen auf der Basis der XML Web-Service Techologie. Darmstadt

Turowski K. (2003): Fachkomponenten: Komponentenbasierte betriebliche Anwendungssysteme, Aachen

Papazoglou, M.P.; Krämer B., Yang J. (2003): Leveraging Web-Services and Peer-to-Peer Networks, INFOLAB – Tilburg University, Fernuniversität Hagen

Taylor I.J. (2005): From P2P to Web Services and Grids: Peers in a Client/Server World, London et al.

Dustdar S.; Treiber M. (2004): A View Based Survey on Web service Registries, Technical University of Vienna

OASIS (2005): UDDI Version 3.0.2: UDDI Spec Technical Committee Draft, URL: http://uddi.org/pubs/uddi-v3.0.2-20041019.htm

MySQL.com (2005): http://www.mysql.com/

Hibernate.org (2005): http://www.hibernate.org/

Apache Web Services Project, AXIS (2005): http://ws.apache.org/axis/

Apache Web Services Project, jUDDI (2004): http://ws.apache.org/juddi/

Apache Jakarta Project, Tomcat (2004): http://jakarta.apache.org/tomcat/

Sun Microsystems (2004): Java Servlet Technology J2EE, http://java.sun.com/products/servlet/

Organization for the Advancement of Structured Information Standards (OASIS) (2004): Web Services Security: X509 Certificate Token Profile, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

Atkinson B. et al. (2002): Web Services Security (WS-Security) Specification, http://www-106.ibm.com/developerworks/webservices/library/ws-secure/

W3C (2002): XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/

Apache (2005): XML Project, XML Security, http://xml.apache.org/security/

Hasan J. (2004): Service-Oriented Architecture in C#, Apress

Siddharth Bajaj et al. (2004): Web Services Policy Framework (WS-Policy)

Trusted Computing Group (2003): TCG Software Stack Specification (TSS), https://www.trustedcomputinggroup.org/

Anderson S. et al. (2005): Web Services Trust Language (WS-Trust)

World Wide Web Consortium (W3C) (2002): Web Services Activity Statement, Introduction, http://www.w3.org/2002/ws/Activity

Siddharth Bajaj et al. (2003): Web Services Federation Language (WS-Federation), http://www-106.ibm.com/developerworks/webservices/library/ws-fed/

Apache Web Services Project, WSS4J (2005): http://ws.apache.org/ws-fx/wss4j/