Service-oriented computing (SOC) and the relationship to EBTs

Sakari Itkonen

Helsinki November 2, 2009

Advanced eBusiness Transactions for B2B-Collaborations -seminar
UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta/Osasto – Fakultet/Sektion –Faculty/Section	Laitos – Institution – Department
Faculty of Science	Department of Computer Science

Tekijä – Författare – Author

Sakari Itkonen

Työn nimi – Arbetets titel – Title

Service-oriented computing (SOC) and the relationship to EBTs

Oppiaine – Läroämne – Subject

Computer Science

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
	November 2, 2009	12 pages

Tiivistelmä – Referat – Abstract

The purpose of this seminar paper is to give the reader a picture on how service-oriented computing can be used for business-process engineering and business-to-business automation. The work presents the common techniques and protocols used in Web service related transactions. Also the challenges that inter-organizational eBusiness transactions pose for the current techniques are discussed. Finally three proposed solutions to these problems are presented.

Avainsanat – Nyckelord – Keywords

Service-oriented computing, transactions, eBusiness

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Contents

1 Introduction

The purpose of this seminar paper is to give the reader a picture on how service-oriented computing can be used for business-process engineering and business-to-business automation. While it is possible to use collaborative business processes with service-oriented computing the safeguarding of business-to-business automation is still a challenge. If the automation isn’t safeguarded, enterprises will not see all the potential benefits and advantages service-oriented computing has to offer. Safeguarding issue can be addressed with injecting business semantics into eBusiness transactions. Transactions should also be able to cope with exceptions and know how to handle different sorts of compensations.

Even though the problem are is quite new there are several proposed methods or protocols on how to deal with the issues raised by inter-organizational transactions. This paper introduces three of them. The methods are presented in on a higher conceptual level and the actual algorithms are not examined.

To actually implement inter-organizational e-Business transactions in a Web service or Services-oriented architecture context some changes in techniques need to be done. ACID based transactions do not work well in loose-coupled environments but nevertheless they shouldn't be forgotten. Loose-coupled environments like the Web need different kinds of transaction principles but the back-end systems behind Web services still often remain to be strongly-coupled and therefore work really well with ACID transactions. Transaction protocol for Web services and eBT should be specified so that it takes ACID into consideration but still manages to satisfy all the needs of eBusiness participants.

The remainder of the work is organized as follows. Section 2 introduces and explains terms and techniques related to Service-oriented Computing and eBusiness transactions. In section 3 the challenges that inter-organizational transactions pose are discussed. Section 4 presents the proposed methods to deal with the challenges. Finally, section 5 presents the summary and conclusions.

2 Basic concepts and techniques of Service-oriented computing

2.1 Service-oriented computing

Service-oriented computing is a design paradigm that specifies the creation of automation logic in the form of services. The main concept behind SOC is to construct applications from independent components or services that have standard interfaces. Tsai et al. tell in their paper [Tsai06] that this is achieved by separating software development into three parties. The parties are service providers, service brokers and application builders. Service providers write the components using programming languages. Service brokers register and publish the components. Application builders then use the published components to construct applications with high-level specification language. SOC promotes loose-coupling and enables recomposition of applications at runtime.

2.2 SOA stack

MacVittie notes in her article [Mac07] that service-oriented architecture stack is still evolving and there really isn't a definition what a SOA stack should contain. Basically the SOA stack comprises of the core layers which are transport, messaging, description, and delivery layers. In addition to these you could have coordination, orchestration and transaction management layers just to name a few. The reason for the vagueness of the concept is that no single vendor controls the contents and techniques used for service-oriented solutions. Vagueness may cause some problems but it is also very much suitable for service-orientation because it provides the much needed flexibility and interoperability. This also enables each organization to define their own SOA.

Figure 1 presents one interpretation of the current SOA stack next to the representational state transfer (REST) software architecture stack. The common protocols used with the layers are HTTP and FTP for transport, SOAP for messaging, WSDL for description and UDDI for discovery.

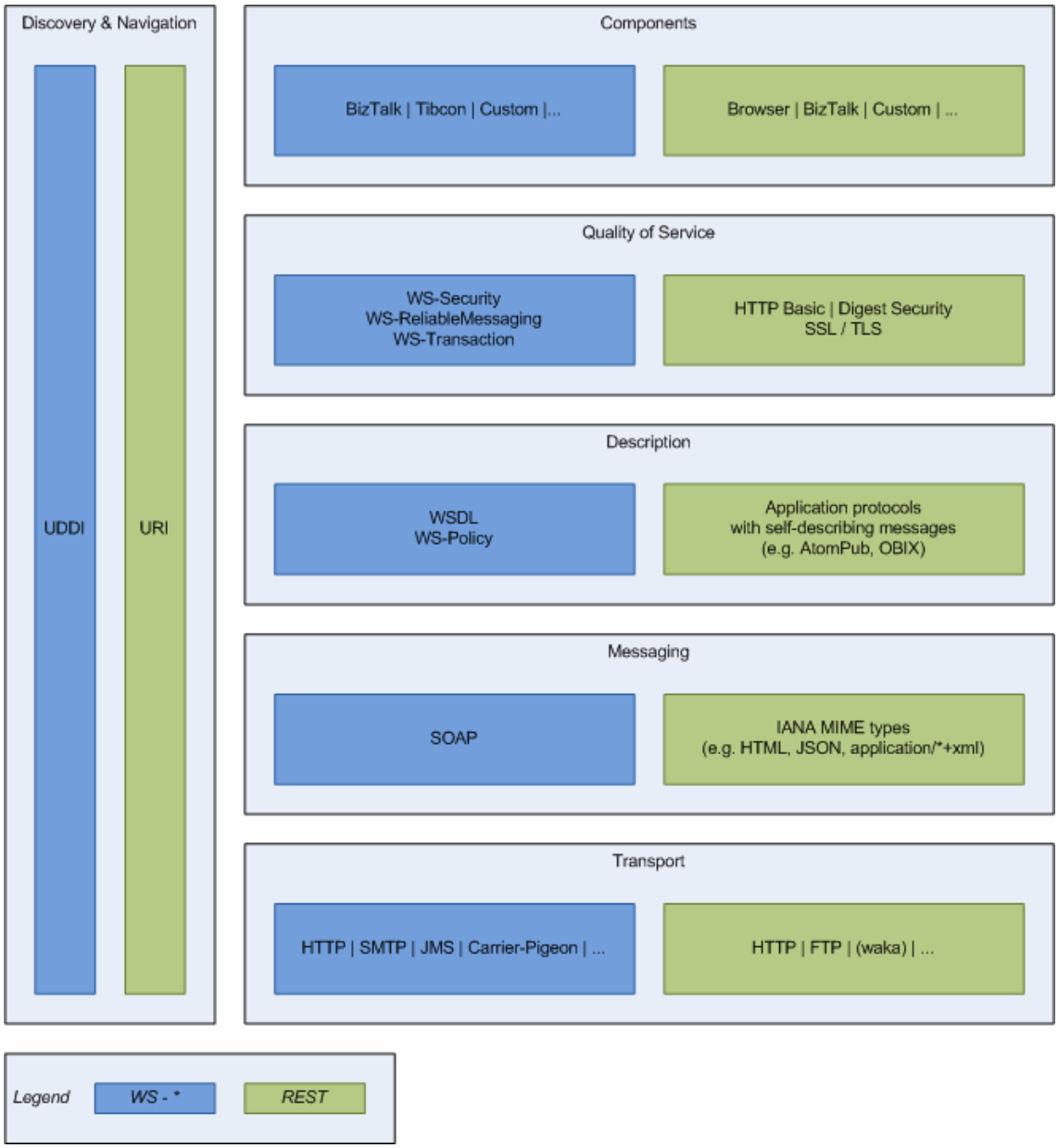

Figure 1: One example of a SOA stack. [Dea08]

2.3 Web services

Webber and Parastatidis [WePa03] define Web services as an architectural paradigm that is based on messages. A Web service advertises the messages it is willing to exchange but doesn't give away any of the underlying operations. The messages are supposed to contain all the data that the requested action needs to complete. A Web service is basically a front-end of the set of actions the hosting company is prepared to execute. Services and their consumers share only the vocabulary that is defined with XML and XML Schema. A Web service is also stateless because received messages are

not thought to have any association with any prior messages. Web services architecture promotes loose-coupling because all the needed information between parties is shared just before implementation and deployment.

2.4 eBusiness

eBusiness functions exactly like conventional business that is to say it is based on supply and demand. eBusiness just means that all the transactions are conducted via the Internet.

Advantages of eBusiness include reduced cost of doing business, access to a wide range of information, opportunities to adopt to new business models, better communications, 24/7 operation hours and new kinds of marketing possibilities. On the other hand eBusiness does pose some challenges such as legal and security factors. Legal factors could be for example property rights issues or differences in legislations between two countries. Security factors include frauds and even internet warfare.

2.5 UDDI

UDDI is a central protocol in the Web services stack and it is sponsored by Organization for the Advancement of Structured Information Standards (OASIS) [Oas04]. UDDI stands for Universal Description, Discovery, and Integration. The purpose of the protocol is to describe a registry of Web services and information about them. UDDI is actually a set of Web services itself that supports the discovery of other Web services providers and their technical interfaces. It is based on such standards as HTTP, XML, XML Schema, SOAP and WSDL.

2.6 WSDL

WSDL (short for Web Services Description Language) is an XML-based language that is used for describing Web services. Christensen et al. have released WSDL 1.1 specification [Chro01] in 2001. Services are defined as collections of network endpoints or ports that are capable of message exchange. WSDL describes communications in a structured way. Abstract definitions and concrete instances are separated. Because of this the abstract definitions can be reused. Ports are defined by network addresses and a

collection of ports define a service. Elements used in WSDL are:

- Types: a container for data type definitions (uses XML Schema definition of some other type system).
- Message: the abstract definition of the data being communicated.
- Operation: the abstract description of an action supported by the service.
- Port Type: the abstract operations that defined endpoints support.
- Binding: the concrete specification of the protocol and data type the port type uses
- Port: an endpoint that is defined by a network address
- Service: a collection of endpoints that define the service

Typically WSDL is used with SOAP and XML Schema. Web service provides the client with a WSDL file from which the client determines what sort of operations the provider can execute. If the operations need special data types they are presented in the form of an XML Schema. Now the client knows how to use the services provided and can send messages to the service using SOAP. Note that WSDL is not tied to these protocols so it is possible to extend the type definitions if there is a need for that.

2.7 BPEL4WS

For the sake of clarity the acronym BPEL4WS is used in the following although it is also called just BPEL or WS-BPEL. WS-BPEL is the most logical name as it uses the same naming convention that other Web services related techniques use. However because the recital is based on the original BPEL4WS specification that is the name used here.

Curbera et al. define in their specification [Cur02] BPEL4WS to be a notation for specifying business process behaviour in Web services environment. BPEL4WS is short for Business Process Execution Language for Web Services. The purpose is to define interoperable interactions between parties using the interfaces Web services provide. BPEL4WS is closely tied to WSDL and XML Schema. BPEL4WS is an orchestration

language meaning that it specifies a process that exchanges messages with other systems and is controlled by an orchestrator. Services described in WSDL can be modelled to work together with BPEL4WS.

Business processes are divided to two groups in the specification, both of which the BPEL4WS is able to handle. These are executable business processes and business protocols. Executable business process models the “actual behaviour of a participant in a business interaction”. Where as business protocols describe the message exchange between parties but don’t reveal the internal behaviour or participants to each other. These descriptions are called abstract processes.

2.8 ACID

ACID semantics is the basis of the traditional transaction systems. ACID is an acronym of the following properties that should be guaranteed in the transactions:

- Atomicity,
- Consistency,
- Isolation and
- Durability.

Little and Freund [LiFr03] define the properties as follows. Atomicity means that either all of the effects caused by the transaction are committed or they are all undone. Consistency means that results of transactions are consistent. Isolation means that transactions are not visible to each other when they are being executed and they appear to be executed serially. Durability means that the effects of a committed transaction are permanent and not lost.

2.9 Two-phase commit protocol (2PC)

The two-phase commit protocol is an atomic commitment protocol that can be used in transactions. It is based on a distributed algorithm that coordinates all the participants. The protocol is widely utilized because it is highly resistant to failures.

As the name of the protocol suggests it consists of two phases: preparation and commitment. Little and Freund [LiFr03] describe the steps as follows. In the

preparation phase the coordinator process prepares all the participants processes. Participants make their state changes so that they can be rolled back or committed. After this has been done the participants vote either ”yes” or ”no”. If all participants voted ”yes” the transaction succeeds. If anyone votes ”no” the transaction fails. In the commitment phase the coordinator sends commit message to all participant processes who complete their operations and free all blocks. Because of this behaviour the two-phase commit protocol is a blocking protocol. All participants who voted ”yes” must block their resources from other uses until the coordinator sends a phase 2 message.

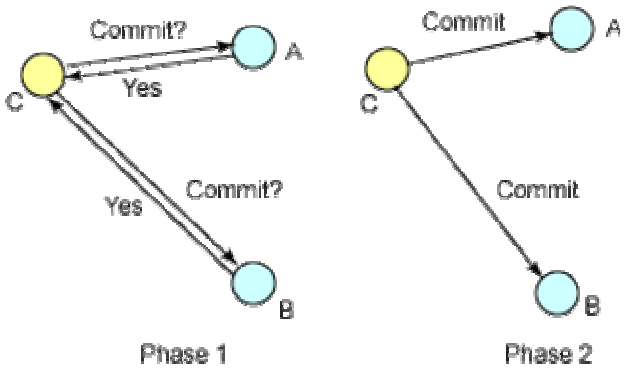

Figure 2: Two-phase commit protocol. [LiFr03]

3 Problems with current techniques

Service-oriented Computing paradigm is a promising concept for eBusiness as it enables the construction of new kinds of organizations and unprecedented dynamicity. Loose-coupling and business-to-business automation offer a whole new kind of freedom to build services but at the same time it does pose a challenge for the existing techniques. The ability the compose services at runtime based on previously defined business-processes is without doubt alluring to companies who wish to have competitive edge and make the most of the available technology.

Montegut and Molva propose in their paper [Momo03] that here are two challenges that arise from the use of composite Web services. These are relaxed atomicity and dynamicity. Composite services do not require as strict atomicity as traditional transactions. It should be possible to keep some results of composite service execution even if it fails to complete all of its tasks. Existing transactional models are not designed

to handle dynamic service collaboration or runtime composition of services. The existing techniques allow transactional requirements to be validated only after the composite Web service has been created. The validation should rather be done right at the actual constructing process.

Web services based inter-organizational transactions are basically distributed systems. This means that there can be failures on the network, computers, operating systems or applications. Obviously failure of one part of the system shouldn't paralyze the whole transaction and lock every process involved in it.

Little and Freund note in their work [Lifr03] that Web service based transactions have some qualities that makes traditional transaction semantics unsuitable. These are long execution periods, commitment negotiations between participants and relaxed isolation levels. Web services are a way to integrate processes between enterprises and to achieve this there has to be a consensus. Different sets of participants should be able to get different outcomes and participants should be able to exit groups when it's appropriate. These sorts of requirements are impossible to handle with traditional transaction protocols. This sort of flexibility is the greatest strength of the Web service transactions and at the same time it is the thing that poses a great challenge. Before these challenges are solved enterprises may have hard to trusting the systems to work correctly and offer the benefits they are supposed to.

4 Proposed ways to solve issues

This section presents three proposed techniques or protocols that are designed to deal with the challenges raised by inter-organizational eBusiness transactions. BTP and WS-C are based on the article by Little and Freund where as the unnamed method is based on the work of Montegut and Molva.

4.1 Montegut and Molva's approach

In their paper [MoMo03] Frederic Montagut and Reifik Molva propose a new way to automate transactions in Web services. Their proposed method takes both the functional and the transactional requirements into consideration in the composition of the service. The following recital tries to present the steps of the approach as simply as possible

without explaining the algorithms used in detail.

The method provides designers with an Acceptable Termination States model (ATS) which should be used to define the transactional requirements of the composite Web services. AT S defines the termination states a work-flow is allowed to reach so that its execution is judged consistent. The paper describes four types of transactional properties which are

- Compensatable,
- retriable,
- retriable and compensatable or
- pivot.

Compensatable means that the results produced by the task can be rolled back where as retriable means that the task is sure to complete successfully after a finite number of tries. Retriable and compensatable is self-explanatory and pivot means that the task is neither compensatable nor retriable.

Workflow designers define the transactional requirements of the activity which the method takes as an input and then uses it to compose the wanted service. The authors claim that the method can be used to augment current systems or deployed as rules Web services coordination specifications. The main steps of the method are:

- Analysis of TS(W)
    - The termination states of the workflow are analysed and the acceptable termination states should be formed based on the termination states.
- Forming ATS(W)
    - In this step the acceptable termination states of the workflow are designed and checked to be consistent.
- Deriving composite services from ATS(W)
    - This is called Transaction-aware assignment procedure. An algorithm uses the previously defined acceptable termination states of the workflow to find suitable services from a list of services. The process is iterative.

The method seems to take the transactional side into consideration and work well in that aspect. Problem is that the paper is presented in a very mathematical filled with algorithms and theorems so it's easy to misunderstand some of the aspects and it can prove difficult to read. Also this is just a theoretical model so it is not yet actually implemented. Because of this the results in a working system might not be what are expected. These points aside the method seems to be a very rational approach to the transactional challenges and is definitely worth considering as an option.

4.2 BTP

BTP is short for Business Transaction Protocol. It utilizes the two-phase commit protocol however it doesn't enforce ACID semantics. Relaxation of ACID semantics is taken very far in BTP and durability and isolation aren't handled in anyway.

Essentially BTP steps out of the domain of a transaction protocol and mixed non-functional and functional aspects together. The protocol doesn't care how consensus is achieved in the transactions as long as it is. BTP also enables the controlling of the time between the two phases. BTP ties coordination to the transactions itself by allowing business-logic to injected after the prepare phase of two-phase commit protocol.

BTP introduces two types of extended transactions. These are *atom* and *cohesion*. Atom is similar to a typical atomic transaction in that all participants see the same outcome and it can be either accepted or rejected. However the properties of an atom (like isolation and durability) are not so strictly defined as they are in a standard atomic transaction. Cohesion is suited for long-running business activities and therefore it displays relaxed atomicity. Cohesion type let's participant see different outcomes and some may accept them and some reject. Cohesion type will have to decide a *confirm-set* which is essentially an atom type and decides the final outcome of the activity. All participants of the confirm-set see the same results and must either accept or reject them. In order for cohesion types to work services have to expose their back-end implementation. This is obviously against the whole concept of Web services architecture.

Quite surprisingly BTP isn't tied to Web services and has its own service stack inside the protocol. So it doesn't contain WSDL and doesn't take advantage of Web services architecture. This means that the users of the specification have to put a lot of effort to make BTP work with Web services. On the other hand BTP offers a lot of flexibility because of its relaxed restrictions.

4.3 WS-C and WS-Tx

WS-C is short for Web Services Coordination and WS-Tx for Web Services Transactions specifications. WS-C is a framework that allows the use of different coordination protocols. WS-C and WS-Tx are designed just for Web services and use therefore standards like WSDL. In this technique the transactions are separated to use WS-Tx and the coordination to use WS-C. In other words WS-Tx kind of works on top of WS-C. Unlike BTP WS-Tx defines all semantics clearly so all parties know what to expect from the transactions.

WS-Tx introduces two completion patterns. These are *Atomic Transaction* and *Business Activity.* Atomic Transactions are basically based on traditional ACID semantics that are used in Web services inter-operatibility scenarios with back-end systems that are optimized for ACID. This way the legacy systems can be used in new usage scenarios. Business Activity is for the long-duration interactions and it can be choreographed with existing languages like BPEL4WS. This is suitable for reservation scenarios for example. Service can inform the Business Activity that requested action can be rolled back if the Business Activity wants to.

WS-Tx doesn't provide services with the actual compensation mechanics. This is something that the service providers have to do themselves and this way they can define mechanisms that are best suited for their needs. It should be noted that unlike BTP WS-Tx allows new protocols to be supported when the need for them arises and techniques are invented.

5 Summary and conclusions

The need for a protocol or technique that could safeguard inter-organizational eBusiness transactions is obvious because more and more business is being conducted in an electronic environment. eBusiness also enables new kinds of organization models the use of which could considerably increase competitive edge of companies. Because of this safeguarding mechanisms become increasingly important and the lack of a proper eBusiness transaction protocol keeps hindering the progress and usability of modern systems.

The three methods presented here enable the business-semantics to be injected in to the transactions but every one of them takes a different approach on this. Based on the papers found while researching this topic it would seem that the WS-C/WS-Tx is the most mature and capable of the suggested solutions to the problems raised by Web services. The obvious benefit of WS-Tx is that it is designed just for Web services and is therefore compatible with a lot of existing standards. This means that it also benefits from all future improvements in the SOA stack.

BTP seems to be the most flexible proposition but this flexibility comes with a price. Lot of the decisions needed in the transactions are left for others to handle and it’s likely that back-end resources of a service have to be exposed to service users which should never be done in a Web service architecture. The relaxed approach to two-phase commit protocol is also an interesting concept. Whilst designers may it useful and even practical in some cases it also creates situations where the coordinator of the activity has to have close relations to the services.

Montegut and Molva’s approach is also intriguing as it should be possible to implement on existing protocols or use it just as rules on adding the needed functionality. Mathematical nature of the paper is a little aversive and that takes away of the use of the work.

Overall it seems that the problem is taken seriously and industry is trying to come up with solutions to make eBusiness more safe and reliable. Time will tell which solution gets chosen as a standard and which are forgotten.

References

Bel03 Bellwood, T. and L. Clement, and D. Ehnebuske et al, 2003. UDDI Version 3.0, Published Specification. http://uddi.org/pubs/uddi-v3.00-published-20020719.htm

Cur02 Curbera, F. and Y. Goland, J. Klein, F. Leymann, D, S. Thatte, and S. weerawarana, 2002. Business Process Execution Language for Web-Services 1.0. http://www.ibm.com/developerworks//library/ws-bpel/

Chr01 Christensen, E and F. Curbera, G. Meredith, S Weerawarana, 2001. Web Services Description Language 1.1. http://www.w3.org/TR/wsdl

Dea08 Dean, Alan, 2008. SOA STACK. http://service-orientated-architecture.pbworks.com/SOA%20Stack

LiFr03 Little, M and Freund, Thomas, 2003. A comparison of web services transaction protocols. http://www.ibm.com/developerworks/webservices/library/ws-comproto/

Mac07 MacVittie, Lori, 2007. SOA, Stacks, and Standards. http://devcentral.f5.com/weblogs/macvittie/archive/2007/06/13/2857.aspx

MoMo03 Montagut, F. and R. Molva, 2006. Augmenting web services composition with transactional requirements. icws, 0:91-98

Oas04 Organization for the Advancement of Structured Information Standards, Introduction to UDDI: Important Features and Functional Concepts, 2004.

Tsai06 Tsai, W. T. and Y. Chen, G. Bitter, D. Miron, Introduction to Service-Oriented Computing, 2006. http://www.public.asu.edu/~ychen10/activities/SOAWorkshop/Background.pdf

WePa03 Webber, J. and S. Parastatidis, 2003. Demystifying service oriented architecture. Web Services Journal, 3(11):40-44.