B2B Contract Implementation using Windows DNA

Ning He, Zoran Milosevic
CRC for Enterprise Distributed Systems Technology (DSTC)
The University of Queensland, QLD 4072 Australia
<ninghe, zoran>@dstc.edu.au

Abstract

This paper describes our implementation of a support infrastructure for electronic contracting – an important ingredient of Business-to-Business (B2B) e-commerce. The paper first explains the main benefits of the new generation of Microsoft technologies - Windows Distributed interNet Applications Architecture (DNA) and BizTalk. This is followed by a detailed description of how we take advantage of the XML tools provided by these technologies - to implement our enterprise model of contracts. We use a real-world contract scenario as a test-bed for examining our e-contract architecture and for implementing our prototype. The prototype was developed using publicly available BizTalk preview technology.

Keywords: Business contract, XML-message, BizTalk

1 Introduction

A contract is an agreement whose purpose is to decrease/eliminate risks associated with the interactions between trading partners. At present, business contract processes, including negotiation, signing, validation and monitoring, are manually carried out by humans. The rapid growth of Internet and B2B e-commerce technologies, especially XML-based messaging systems, makes it possible to produce and exploit electronic representations of contract forms. These new technologies can also be used to facilitate management of electronic contracts, other business documents and corresponding business transactions, such as checking validity of contracts, contracts negotiation and monitoring. For example, the Windows DNA infrastructure can be used to support many aspects of B2B processes from business document transmission and database updating to business partner management, which makes the contract automation process achievable.

This paper describes our general architecture for supporting electronic contracts and demonstrates how BizTalk technology can be used to implement its key features. The rest of the paper is structured as follows. Section 2 outlines key aspects of our enterprise contract architecture and sets the foundation for e-contract design and implementation in a technology neutral manner. Section 3 describes the Windows DNA infrastructure and its main B2B component – the BizTalk family. Section 4 presents our approach for using BizTalk capabilities to implement our contract architecture described in section 2. It also elaborates on how other DNA components (e.g. SQL server and Windows 2000 Server) are integrated with BizTalk in our prototype implementation. Section 5 provides some discussion of the related work. Section 6 concludes the paper and outlines future work directions.

2 Electronic support for business contracts

This section introduces key concepts for an architecture to support electronic contracting (i.e. a B2B enterprise model), derived from a more detailed model described in [3]. It also describes a typical business scenario based on this model. These concepts have been implemented in our prototype and the scenario is used throughout the paper to illustrate our implementation approach.

2.1 B2B Enterprise Model

The B2B enterprise model for electronic contracting is depicted in Figure 1. This model is described in terms of roles and their relationships which together support contract establishment, execution, monitoring and enforcement stages in a contract life cycle. Key roles in the model are as follows [3]:

- *Contract Repository* (CR), to provide electronic repositories to store standard contract forms and optionally, standard contract clauses.
- *Notary*, to store signed instances of standard contracts forms, which can later be used as evidence of agreement in contract monitoring and enforcement activities.
- *Contract Monitor* (CM), to enable monitoring of the business interactions governed by a contract and to signal the contract enforcer if violations are detected.

- *Contract Enforcer* (CE), to enforce the compliance with contract terms. When signalled by the CM, enforcer may send a warning notice to various parties informing them of the violation and possibly prevent further access to the system by non-conforming parties.

B2B electronic contracts can be used for many real estate, banking and insurance transactions, purchase and sale of goods and services and so on. When businesses instantiate contracts, specific contract clauses can be dynamically bound with a contract template to define a new business contract. This process enables flexible changing/updating of existing contracts. After a contract is signed by all the parties, it may be useful to monitor business interactions agreed by the contract. This can be done by auditing the electronic logging of business interactions. In cases when contract terms and conditions were breached, contract enforcer can be employed. The actions performed by enforcer are different from business to business; the ultimate enforcements are to be executed by human decision makers.

We note that there can be several business processes identified in our business contracts model, but our architecture is essentially role-based - to enable support for many types of underlying contracting scenarios (i.e. business processes implementing them).

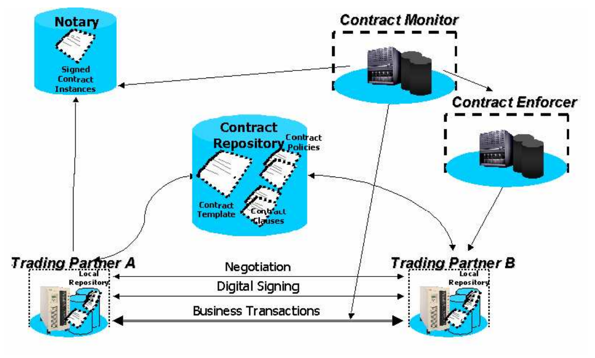

Figure 1 B2B Enterprise Model

2.2 Real-world Contract Scenario

A contract scenario has been developed to provide a definite setting for the implementation of the contract prototyping system. This scenario was initially described in [1]. There are three parties involved in the scenario:

- Muzac.com: A music producer that makes MP3 files and sells them over the Internet.
- eShop.com: A provider of portal and retail e-commerce site services.
- B2B.com: A company that provides software, facilities and services for development of B2B e-Commerce, including those that are needed for electronic contracting.

Muzac.com is interested in using eShop.com's portal and retail services for selling its MP3 music files to customers. After an agreement has been achieved (possibly via several negotiation steps) and both parties have signed the complete contract, B2B.com will be responsible for storing the contract, monitoring the business transactions and enforcing contract execution. In Fig 1, eShop is Trading Partner A and Muzac is Trading Partner B. Other roles, such as Contract Repository, Notary, Contract Monitor and Contract Enforcer reside at B2B.com.

Figure 2 depicts a simplified UML sequence diagram identifying main steps in the interaction between parties involved. A detailed elaboration on how we realize this scenario is presented in section 4. As depicted in Fig. 2, we assume that Muzac initiates a contract offer and sends it to eShop. This trading partner inspects the offer and if correctly signed by Muzac, it adds its own signature and sends the completed contract to Muzac.

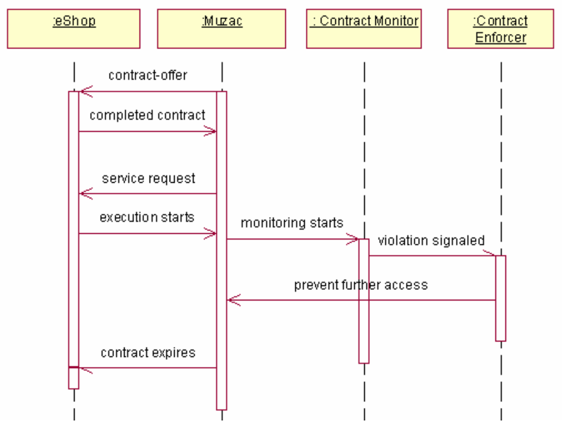

Figure 2 UML Sequence diagram of contract scenario

3 Windows DNA Overview

Business partners that participate in B2B transactions need to manage their business interactions, including the transmission of contracts and other business documents, the execution of business contract terms and storage of the complete contract documents. Microsoft's BizTalk technology addresses these needs and provides an XML-based messaging transmission framework. BizTalk is a technology within the Windows DNA architecture, which also includes the Windows 2000 family, SQL Server, Visual Studio, Host Integration Server, Application Center and Commerce Server. The contract prototype presented in this paper is developed using Windows 2000 , SQL server 7.0, Visual Studio 6.0, BizTalk Server and BizTalk Jumpstart Kit (see figure 3). The overall Windows DNA architecture supports distributed applications interoperability by using standard XML messaging specifications.

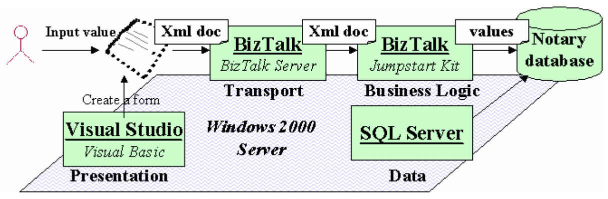

Figure 3: Windows DNA components

3.1 Windows 2000 Server

Windows 2000 Server is the foundation layer upon which BizTalk Server resides. One major feature of the Windows 2000 platform that we will employ is the MSMQ message queuing service.

MSMQ is one transport protocol supported by BizTalk. The advantages that message queuing has over other transport protocols, such as HTTP, SMTP, and FTP, are that it guarantees one time message delivery and enables both asynchronous and synchronous communication. For example, if organizations choose an asynchronous MSMQ mode for sending e-contracts between them they do not have to wait for the reply online. Besides, it is guaranteed that once sent, the message will be stored in the queue and will not be lost even if there is some transmission error (which can be also used for debugging and trouble shooting purposes).

On a receiving side on the other hand, an organization can define a trigger that checks if the message satisfies the condition specified in the trigger. If so, an appropriate behaviour needs to be executed. This can be done by using an application or COM components associated with the trigger. Hence, MSMQ trigger is a message queuing application, which enables the definition of business rules describing how the queue's incoming messages can be linked to the COM components or other standalone executable programs. These programs can then implement part of the business logic.

For example, in the case of our contract prototype, the contract policies and business rules can be applied and administered by the contract monitor in response to the service transaction messages and the contract enforcer can be triggered in case of violation.

3.2 BizTalk Overview

As stated previously, BizTalk technology is aimed at facilitating and integrating XML-based business processes within and between organizations for supporting e-commerce. BizTalk technology consists of:

- BizTalk Framework, which provides specifications for the XML-based messaging implementation;
- BizTalk.org web site [4], which hosts a library of BizTalk XML schemas, BizTalk framework specification and a forum for developer community;
- BizTalk Server 2000 for server side document transformation and routing;
- BizTalk Jumpstart Kit (JSK) for client side document execution and business logic application. The components of the JSK are anticipated to be included as part of final BizTalk product release (planned for the end of this year).

3.2.1 Server Side BizTalk

BizTalk Server is a data-translation and application-integration server for exchanging XML-based business documents across the Internet. By using BizTalk Server, organizations can receive XML documents, parse the documents based on specific schema and deliver the documents to their respective applications inside the organization.

BizTalk Server 2000 Technical Preview is the current version of the BizTalk Server that we have used for the prototype implementation. It provides server side automated support for document transmission, integration and management. We use its services and utilities for prototyping of B2B contract specific activities. The services include XML schema translations, various transport protocols (HTTP, SMTP, FTP, file, message queuing, etc) and BizTalk XML formats to build B2B systems. The utilities are: BizTalk XML Schema Editor to create and modify XML schema specifications; a BizTalk Mapper to provide data transformations; Management Desk to provide a web-based graphical user interface for simplifying the management of organizations and trading partners; and Administration Tools to provide control over document flow, document tracking, business analysis and troubleshooting.

All the messages sent to BizTalk Server will be stored in the queues. If an error occurs during transmission, the document can be traced from the suspended queue in the Administration Tool. Otherwise it stays in the work queue waiting to be consumed.

The main component in BizTalk Server for transmission of data across the network is a *pipeline*. Pipeline directs the server through the steps necessary to transport the documents to the trading partner or applications within the organization. A pipeline is configured at the pipeline editor in the Management Desk, where trading partners need to specify the *inbound* and *outbound agreements* for the *document definition*.

The Document Definition indicates what types of documents the source organization can send to the destination organization using agreements. An agreement, (defined in the agreement editor in the Management Desk), represents the rules that regulate electronic data exchanges from the source to the destination organization. In creating agreements, a trading partner needs to specify the source and destination of the document, the types of the message (document definition) and the transport protocol. The agreement editor can also define how documents are enveloped, mapped and secured.

There are two types of agreements: inbound and outbound. Outbound agreements regulate how documents are directed out of BizTalk Server, and inbound agreements determine how messages come into the server. One or more pipelines can then be specified to describe how a document definition from one inbound agreement is to be linked to one from an outbound agreement. This link enables message routing at the BizTalk Server and mapping between document definitions, if these are different message types.

3.2.2 Client Side BizTalk

The BizTalk Jumpstart Kit (JSK) provides client side tools and a framework for integration between business applications. Our contract prototype uses the Jumpstart Kit to direct XML messages of different types (that contain contract and other business documents) to the corresponding applications maintained by this BizTalk Server. We also use the JSK to facilitate access to databases and perform business logics.

Figure 4 shows main components of the BizTalk JSK involved, e.g. in processing incoming contract offer at eShop site (BTS represents BizTalk Server 2000, I.A. is the inbound agreement, O.A. is the outbound agreement and A.A is the application adapter). Note that a more detailed description of the contract-specific message flows and application interfacing is given in 4.2.2.

The components provided by the JSK include adapters, plug-ins, envelops, selector and namespace service as summarized below.

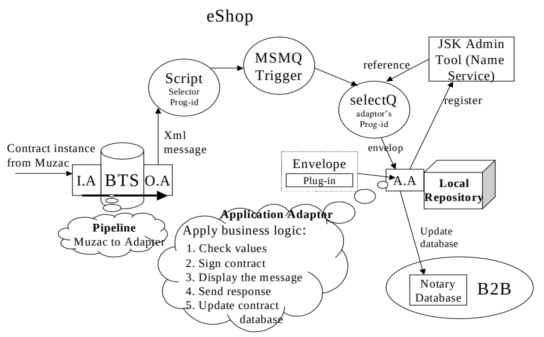

Figure 4: Contract Transmission

- Application adapters are COM objects whose creation is supported by a JSK application adapter template. An application adapter is used to consume incoming messages of certain types and implement business logic that handles this message. The flow of messages targeted for a particular adapter is as follows. When a message arrives at the BizTalk Server of an organization (via a transport adapter, e.g., MSMQ), it is passed to the selector provided by the JSK. The selector will then refer to the name service (i.e., JSK Administrator Tool) to hand the message to the appropriate application within the domain of this organisation. There is one application adapter per message type.

- The plug-in wizard is a JSK development tool used for the creation of plug-ins. A plug-in is a COM object hiding the details of the underlying XML management logic and node tree structure. It is based on the DOM standard. [11].

- Selector is a subsystem in the BizTalk Jumpstart Kit that is used for forwarding of the incoming messages to the corresponding business applications. It consists of: two MTS/COM+ components (selector.submit and selector.engine), a message queue (a private queue named ‘biztalk’), and an NT service (called ‘selectQ’). The SelectQ forwards messages coming into the queue into the selector.engine responsible for processing the message. Essentially, Selector represents an embodiment of an MSMQ and MSMQtrigger facilities (as described in 3.1) for the purpose of a BizTalk developer. In fact, MSMQ trigger is another and a more reliable way of routing messages.

- Namespace service resides inside the BizTalk Jumpstart Kit administration tool. The service is used by selector.engine, which decides what components to call based on the contents of the message and values of the namespace.

- Envelope represents a wrapper for one or more business documents (which, along with the corresponding attachments, constitute a manifest element type within the envelope schema). In the JSK, there is a COM object created by the selector that mirrors the XML envelope message. This envelope object will reference the corresponding plug-ins.

By using JSK facilities, we developed our prototype to handle the incoming XML messages and manage the business logic. We start by using the plug-in wizard provided by the JSK to create a contract plug-in based on our XML contract schema (called contractTest). This plug-in helps us to insert the specific contract values into contract instances that will be exchanged until the final agreement has been reached. We use the application adapter template to implement the business rules for handling the sequence of steps at the receiving side. This includes the processing of contract instances and finally their storage into databases, e.g. storing signed contracts into the notary database and transaction data (during business transactions execution stage).

If BizTalk Server is regarded as the XML-based infrastructure for B2B document transmission, the BizTalk JSK is the application-to-application integrator.

4 Implementation of Contract Support using Windows DNA

This section presents our prototyping approach to realize the business scenario based on B2B enterprise model outlined in section 2. We first describe our working environment, followed by detailed presentation of the two stages in a contract life cycle: the pre-contractual stage and service execution stage.

4.1 Working environment

The complete contract scenario involves three organizations (Muzac, eShop and B2B) and thus the real implementation infrastructure should consist of three Windows 2000 Servers - one at each of these sites. On top of each Windows server, there is a BizTalk Server for coordinating the communication among the organizations.

Our initial working environment consisted of one machine used initially for ‘proof of concept prototyping’ - and thus we begun by simulating three servers logically. We are currently in the process of porting the existing implementation onto two servers. However, the design principles and implementation are independent of the numbers of machines used. This section focuses on these design principles, rather on the deployment issues.

We note that each server (at Muzac, eShop or B2B site), needs to have Windows 2000 server, BizTalk Server 2000 Technical Preview, BizTalk Jumpstart Kit, and Visual Studio 7.0 installed. At B2B.com, SQL server 7.0 is required for Notary Databases - used to store the contract instances and keep records of the contract transactions.

4.2 Support for Pre-contractual stage

The pre-contractual stage covers all the necessary steps involved in the business contract negotiation and establishment process. The stage is decomposed into three phases: initiation of a contract offer by one party, transmission of XML based contract instances during negotiation, and processing and validation of contract terms after the messages are recieved.

Standard contract templates are stored in the Repository at B2B.com. Either eShop or Muzac can then download a specific business contract template from B2B, customize them and integrate them into business operations. In the current implementation we simplified the scenario by assuming that an authorized person at Muzac initiates a contract offer by filling out a contract template, signing it and forwarding the contract offer to eShop.

The contract is then processed by the B2B software running at eShop’s server. To further simplify the scenario, we assume that such contract instance received by eShop is acceptable to the eShop, after which this organization only needs to ensure that Muzac has signed the offer. If so, the eShop will then sign this contract instance and forward it to the Notary at B2B. The Notary is implemented as a SQL database. Once an authorized person at eShop approves the contract, the application adapter dealing with the receipt of the contract message will automatically update the contract table inside the Notary (Fig 5). This figure shows the working procedures of the pre-contractual stage (BTS denotes BizTalk Server 2000, I.A. is the inbound agreement, O.A. is the outbound agreement and A.A is the application adapter).

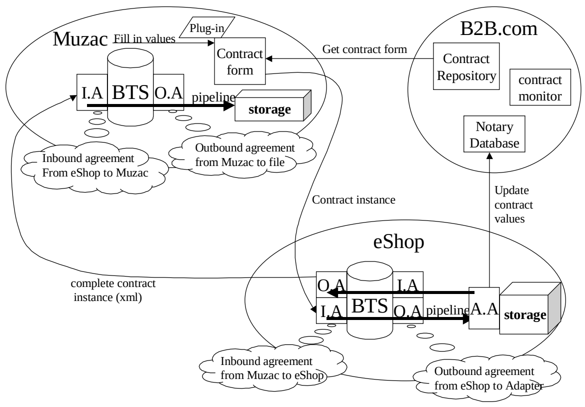

Figure 5: Pre-contractual Stage

4.2.1 Contract Initiation

The scenario begins with Muzac opening an electronic contract form and completing the contract by entering the values required in the contract form and then sending the completed contract offer to eShop. The contract form is a VB form created based on the contractTest.xml schema. For the purpose of our prototype, the XML schema is simplified and contains only six elements: the names of two trading partners, service levels and corresponding price fields and the start and end date of the contract. The schema is compliant with the BizTalk Framework 1.0 specification so that it is understandable by BizTalk Server.

We note that we are also experimenting with more comprehensive schemas that reflects real business contracts, such as the one presented in [1] and [2]. The validation of the contract schema can be done using BizTalk Editor – to ensure its compliance with the BizTalk Framework specification. We store the validated schemas into the BizTalk Server's document specification repository - for future retrieval of document definition

In order to make a valid offer, several fields are required in the form: the names of the related parties in the contract (Muzac and eShop in our case), the start date and the end date of contract and service level which determines the price of the contract. For instance, the price for service level 1 is $2000, level 2 is $2500 and level 3 is $3000. The party that makes the offer must sign it before forwarding it to the other party.

Once the submit button is pressed, the values entered by Muzac will be added into the XML instance. Based on the specification of the corresponding pipeline, the business document would be sent to the BizTalk Server at eShop (BizTalk's Interchange.Submit function is used here, where the pipeline is one of the parameters used).

4.2.2 Contract Transmission

In our prototype, all the document transmission and message transformation processes among trading partners are centrally coordinated by the BizTalk Server at each site. Thus the way contract instances are transmitted is entirely specified by the configuration of BizTalk Server. Namely, before receiving, processing and forwarding XML messages, the organization management, document definitions, inbound outbound agreements and pipelines should be configured properly in BizTalk Management Desk.

In the case of our proof-of-concept prototype the major communication flows among three parties (as shown in Fig.5) are:

- Muzac to eShop via BizTalk Server at eShop
- eShop to Muzac via BizTalk Server at eShop
- Direct communications between eShop and B2B
- Direct communications between Muzac and B2B

In order to setup the channel between Muzac and eShop, two pairs of inbound and outbound agreements are defined along with pipelines that connect them. For example the bottom-depicted pipeline in Fig.5 at eShop links an inbound agreement from Muzac to eShop and the outbound agreement from eShop to eShop's adapter. This pipeline enables a contract message arriving at eShop to be ultimately processed by eShop's corresponding application adapter (which among other things will finally store it into a local storage at eShop). However, in order to reach the application adapter from the BizTalk Server one needs to use facilities provided by JSK, as follows (see also Figure 4).

Inside the outbound agreement, we have written a VB script that instantiates a JSK selector component. Thus the XML contract instance will be handed over from BizTalk Server to BizTalk Jumpstart Kit by calling the selector.submit API. As mentioned above, the selector will place the message into the biztalk queue (provided by JSK), which is associated with MSMQ Trigger. If the trigger detects that it is a BizTalk standard XML message, the selectQ will start executing and, by referring to the namespace service provided by JSK Administrator Tool, selectQ will find the specific application adapter to handle the contract instance.

4.2.3 Processing of received contract messages

Upon reaching the adapter, the contract instance message will be processed by the application adapter. This adapter implements business logic consisting of the following steps:

1. Validation of contract
   Before signing the contract, eShop needs to make sure that Muzac has already signed the contract and all the values supplied are acceptable.
2. Signing the contract
   After validating the values, eShop will sign the contract.
3. Sending of replies to Muzac
   After eShop has signed the contract, it will send the complete contract instance to Muzac via BizTalk Server.
4. Storing the signed instance in the Notary database
   Finally, the application adapter will fill the contract values into the contractTable inside the Notary database.

We note that our future prototype will have another BizTalk Server at Muzac and in that case the contract instance will firstly be sent to the local server, then forwarded to the remote BizTalk Server. Specific business requirements for Muzac will determine the method for handling messages received on its side (i.e. whether to use another application adapter or to store it in a file).

4.3 Support for service execution stage

Upon successful establishment of the contract, Muzac and eShop can engage in business transactions as governed by the signed contract (stored in the Notary). Business transactions include transmission of MP3 files from Muzac to eShop, payment by Muzac to the eShop and so on. We describe here the transaction of Muzac sending MP3 files to eShop as governed by the contract agreements. To ensure that contractual terms are respected by both parties we employ a Contract Monitor (CM) at B2B.com to closely monitor the behaviours of both parties (especially Muzac) in these transactions. The monitoring is based on the policy specifications that are derived and/or refined from the contract description. This derivation is beyond the scope of this paper and is presented in more detailed in [1].

In our prototype the monitoring is applied to the service level contract element, reflecting various file quotas for files to be stored in the eShop. For example, if the service level specified in the contract is level 2, the related policy for service level 2 is that eShop provides 2000 MB disk space for Muzac to store the MP3 files. The monitoring behaviour of the CM consists of checking the transaction record of transactions performed until this point in time, and checking whether the new transaction initiated by Muzac exceeds the disk quota. If true, the CM will trigger the contract enforcer, which has permission to prevent the transaction. The following figure shows the service execution stage, which includes service execution and contract monitoring.

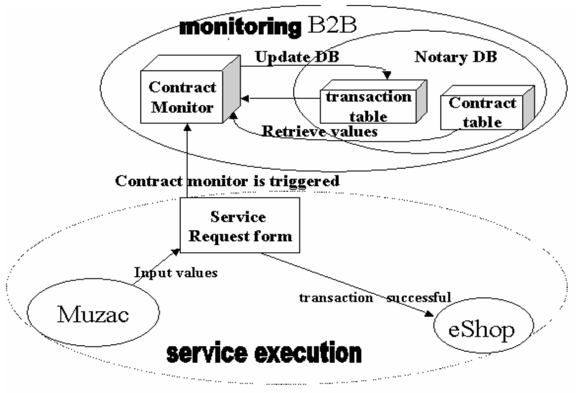

Figure 6: Service Execution Stage

4.3.1 Service Request

Service in our scenario (MP3 file download) is requested by Muzac via an electronic service request form, used to enter the desired service values and submit service requests. The service request entries include values for the MP3 music files quantity, the transaction date and a contract reference that associates the contract with this transaction. This contract reference is an important piece of information, that provides the reference to the contract table stored in a Notary.

Once the submit button is pressed, the contract monitor will be triggered to observe the transaction behaviour by referencing the contract instance, and the transaction records from the notary database according to the contract reference number.

4.3.2 Monitoring and Enforcing

The CM is implemented as a COM object and it performs monitoring activities in the background. Each time when the service is requested by a user, the CM will be triggered to examine the behaviour of both parties. The

CM at B2B's site will thus check the service level contract element (as well as transaction date and contract number) of the contract and compare it to the signed contract instance in the Notary and the available quota based on the previous transaction records. If the transaction complies with the contract rules/policies, the transaction will be processed and the values will be updated into the transaction table. Otherwise a warning window will pop up and the transaction will be terminated immediately.

5 Related Work

To the best of our knowledge there is currently limited support for electronic business contracting as presented in our paper. Some support for electronic contracts has been included as part of the CrossFlow project [7]. The focus of this work is on contracts used to describe agreements between organizations as to how they are to handle business processes crossing organisational boundaries. Our approach to contract architecture does not mandate the way in which contracts are used – this is left to a particular business scenario. However, we do support expression of contractual dependencies for inter-organisational business processes. Some of the initial ideas are presented in [8].

Architecturally, our work is perhaps most closely related to work from COSMOS project as reported in [9]. Our implementation presented here takes advantage of new generation of infrastructures that use XML as a basis for transmitting messages and is based on a more comprehensive framework for enforcing policies associated with contracts.

Finally, the recently published tpaML specification [12] addresses some of the issues reported in our work. The major difference between the tpaML and our work is that tpaML concentrates on specifying the messages sent between the trading partners, their sequences and the choice of the underlying security and other infrastructure mechanisms. Or work is attempting to reflect the business level of electronic contracting, including its legal perspective, and to simulate processes and practices used in real business operations.

6 Conclusion and Future Work

In this paper we presented an approach to implementing a distributed B2B e-contract architecture. We use Windows DNA infrastructure and in particular its BizTalk family of tools, namely BizTalk Server and BizTak Jumpstart Kit. They provide flexible and reliable transmission of XML messages embodying business documents. In addition, they provide several facilities for implementing application logic associated with the transmission of business documents. We found these tools of an advantage when designing and implementing support of electronic contracts.

There are several open issues in this paper that we plan to address in future. First, we plan to provide support for Web-based approach for creating Web contract templates based on XML schemas. This will employ XSLT standard. Second, we plan to implement support for the use of digital signature to ensure security in business documents flow across organizations. Third, we plan to include more comprehensive real-world business rules and policies into the contract prototype and to experiment with multiple BizTalk Servers operating in various environments, ranging from fully trusted to highly un-trusted environments. Next, we plan to adopt the recent BizTalk Framework 2.0 and provide support for SOAP-based messaging systems as well as to use other similar technologies such as MQSeries family from IBM. Finally, our prototype will follow further developments of our contracts meta-model as part of our work on adapting our current architecture to relate to the ebXML meta-model [10] for business processes and contracts. Some of the initial ideas are presented in [13].

7 Acknowledgements

We wish to express our gratitude to Charles Herring for his help in providing us with a better understanding of the intricacies of BizTalk and for his guidance in our implementation efforts. Charles also proposed the scenario that we used as a test-bed for the ideas presented in this paper. We would also like to thank Jim Steel for his comments on an earlier version of this paper.

The work reported in this paper has been funded in part by the Co-operative Research Centre Program through the Department of Industry, Science & Tourism, Australia.

8 References

[1] Herring, C. and Milosevic, Z. "Implementing B2B Contract Using BizTalk"
[2] Goodchild, A., Herring, C., Milosevic., Z., "Business Contracts for B2B", CAISE'00 Workshop on Infrastructures for Dynamic B2B Service Outsouring, Stockholm, June 2000.
[3] Milosevic,Z. and Bond, A. "Electronic Commerce on the Internet: What is Still Missing?"; Proceedings of 5th Conference of the Internet Society; pages 245-254, Honolulu, June 1995.
[4] BizTalk Org Net: http://www.biztalk.org/

[5] BizTalk Initiative: http://www.microsoft.com/biztalk/
[6] Microsoft BizTalk Jumpstart Kit Documentation
[7] www.crossflow.org
[8] Schulz, K. and Milosevic, Z. " Architecting Cross-Organizational B2B Interactions", EDOC'2000, Sep.2000, Japan.
[9] F.Griffel, M.Boger, H. Weinreich, and W. Lamersdorf; M. Merz. "Electronic Contracting with COSMOS – How to Establish, Negotiate, and Execute Electronic Contracts on the Internet" EDOC'98 Workshop, La Jolla, California USA, November 1998
[10] Business Process Project Team, "The ebXML Business Process Metamodel Second Draft" 6/23/00, available from www.ebxml.org
[11] XML DOM Reference: http://www.w3.org/TR/REC-DOM-Level-1/
[12] M. Sachs et al., "Executable Trading-Partner Agreements in Electronic Commerce", IBM T.J. Watson Research Centre, 2000.
[13] J. Cole, Z. Milosevic, "Extending support for contracts in ebXML", submitted to Workshop on Information Technology for Virtual Enterprises (ITVE), Gold Coast, QLD, Australia 29-30 January 2001.