Implementing B2B Contracts Using BizTalk

Charles Herring[1] and Zoran Milosevic[2]
Department of Computer Science and Electrical Engineering[1]
CRC for Enterprise Distributed Systems Technology[2]
The University of Queensland
Brisbane, Queensland QLD 4072
(herring, zoran}@dstc.edu.au

Abstract

We describe our work in implementing Business-to-Business (B2B) contracts using Microsoft's BizTalk. BizTalk is a Microsoft product for the development of XML-based document messaging systems specifically for e-Commerce. We have developed a B2B enterprise model that supports a wide range of economic transactions associated with intangible goods and services. This model guides implementation of our business contract architecture software. The model identifies specific roles such as a repository for contract schemas and instances, auditor, monitor and enforcer. A contract scenario is used to illustrate the architecture and concepts described. We go into detail on the BizTalk implementation of contracts and show how a contract can be used as the basis for policies and plans that govern the behavior of the parties involved in the contract.

1. Introduction

The role of business contracts is to reduce uncertainty associated with the interactions between organizations. This uncertainty can arise due to partial information that trading partners have about each other and/or due to circumstances that are beyond their control. A contract is an agreement whose purpose is to mitigate such uncertainty – by defining obligations of parties to each other - and to have this enforceable by law.

Currently, business contracts are printed on paper and humans carry out the contractual operations and decision-making. The Internet and the rapid move by business to adopt B2B e-commerce enable automation of many aspects of electronic contracting [1]:

Providing repositories for *standard contract forms*. These can be used by businesses when agreeing on the specifics terms of a contract and instantiating contracts. Examples are contract forms (templates) that govern real-estate transactions, banking and insurance forms, purchase and sale of goods and services and so on. Such repositories can also contain forms for standard *contract clauses* that can be reused when deriving *new* contracts that govern specific business interactions. Availability of standard contract clauses also enables flexible changing/updating of existing contracts by simply providing references to the new contract clause from the existing contract. These changes are quite frequent in cases of long-term contracts and are known as *contract variations* and *contract escalation* clauses.

Digital signing of contracts, once specific contract terms have been agreed. This can bring significant savings, in particular in cases where contracts involve multiple, geographically distributed trading partners, such as those related to international contracts, and which can involve significant time and transaction costs associated with handling the contract signing process. This is also a useful way of speeding up contract negotiation process.

Monitoring of the business interactions governed by a contract, so that contract violation can be detected and thus acted upon. This can be done either by logging of business interactions and their audits at a later stage, or by a more pro-active monitoring which can be particularly applicable in cases of electronic services delivery. A special kind of monitoring particularly suitable for longer termed and timed contracts *is tracking* of contracts. This allows timely reaction to some important deadlines such as contract termination, thus making it possible to re-negotiate a subsequent contract and put it in place, before or immediately after the existing contract terminates. Such tracking of contracts can be also seen as a mechanism that prevents situations in which businesses continue their interactions after the contract has expired – thus avoiding undesirable circumstances such as penalties and fines.

Eenforcement of contract terms, in cases when contract terms and conditions were breached. Although some of the enforcements can be done electronically, such as in cases of services provision and billing, the ultimate

enforcements are to be executed by human decision makers.

Essentially a business contract enables forming federations between organizations as it specifies their mutual obligations governing their interaction points. With this premise as a starting point, we have developed a role-based architecture to support such federations and thus realize the automated aspects of electronic contracting identified above. Key roles in the architecture model are [1]:

- *Contract Repository* (CR), providing 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 violation activities are detected.
- *Contract Enforcer* (CE), to enforce the compliance with contract terms. When signaled 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.

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).

This paper focuses on how we implement this architecture using Microsoft's BizTalk, ranging from those facilities that enable producing representation of contract forms to those that facilitate building application logic for each of the roles, taking into account XML messages sent between the roles. In addition, we discuss advantage of having a declarative representation of policies that govern behavior of trading partners and hoe these can be derived from the natural language contract representation. Note that in this paper we assume that possible negotiation, digital signing and other pre-contractual operations have already been performed and we start from the point where there exists a signed contract instance representing an agreement between trading partners.

We begin by introducing a scenario for trading of a specific kind of intangible goods – MP3 format music files (section 2). This is followed by a brief description of Microsoft's BizTalk infrastructure that we use to implement support for automated contracting (section 3). In section 4 we show the use of XML schemas to represent various documents needed for the operations of the business contracts architecture. We also provide a description of the use of BizTalk Jump Start Kit components and BizTalk server (technology preview version) to support execution of the business contract components. Section 5 outlines related work and section 6 provides conclusions.

2. A Scenario for Trading Intangible Goods Using B2B Contracts

We have developed a scenario for the trading of intangible goods to provide a definite setting for the implementation of B2B contracts. The scenario is based on three (fictional) parties:

- Muzac.com[1]: A producer of music in the form of MP3 files that are sold over the Internet.
- eShop.com: A provider of portal and retail (B2C) e-commerce site services.
- B2B.com: A B2B service that provides standard business documents, message formats and software for B2B. This software is based on the specification of roles identified in the business contracts architecture described above. These include contract repository, notary (for digitally signed contract instances) and contract monitor.

Musac.com wishes to use eShop.com's portal in order to sell its MP3 format music files to customers. Both the Muzac.com and the eShop.com use B2B.com's software services. This permits them to easily enter into a trading partner agreement or contract based on B2B.com's standard business documents such as contracts, purchase orders, etc. They also use B2B.com's services for monitoring and enforcing the contract. Hence, eShop.com and Muzac.com can download standard business documents from B2B.com, customize them and integrate them into their operation. The scenario begins with a person (authorized to sign contracts at Muzac.com) filling out a contract form on eShop.com's web site for portal services. (This contract was developed using B2B.com's standard contract templates.) He digitally signs the contract and presses the "submit" button.

The B2B.com software running on eShop.com's server then processes the contract. Since this is a standard contract it can be approved and signed automatically and then routed to B2B.com to start the contract implementation process, as follows. The signed contract instance is first lodged in the Notary when it arrives at B2B.com. B2B.com's contract management software then takes this contract instance and assembles a set of

[1] Use of the names Muzac.com, eShop.com and B2B.com imply no relation to any real companies.

Business Policy documents that will govern the behavior of all parties to the contract. These policy documents can be generated automatically based on the contract instance (containing the particulars of the contract), the contract template (that associates contract clauses with business rules) and the policy templates containing generic rules. We note that these policy templates are derived manually and represent refinements of high-level policies in contract clauses (this will be explained in detail in sections 4.2.and 4.3). Such an assembled policy document can then be transmitted to Muzac.com and eShop.com. The policy documents contains the set of rules each party must follow. The rules specify the obligations, prohibitions and permissions for each party relative to the contract.

Similarly, contract clauses and policy templates are the bases for deriving a collection of actions (called plans) that all three parties need to implement to satisfy the contract. That is, the policy document serves as the basis for generating a Plan Document. The plan is a set of activities specific to that party's performance of the contract. Finally, the activities are mapped onto business object (attributes, methods and events) to implement the actual behavior in software. Also, as a result of this setup process all the necessary logic is in place for monitoring and enforcement of the contract. It is at this point that business transactions may begin. For example, Muzac.com may begin transferring MP3 files into its allocated storage area on eShop.com. eShop.com will generate invoices based on the contract-specified billing criteria. All of these transactions will be monitored by software running at each party's site as well as by B2B.com.

The above scenario provides us with a "laboratory" to develop and experiment with our B2B contract implementation. We can devise tests to verify that each term and condition specified in the contract is correctly implemented via the chain of polices, rules, actions and mappings unto objects. A simulator was developed to model the behavior of Muzac.com and eShop.com. This simulator produces the events that drive the system of business messages, monitoring, etc. In addition to verifying that the system performs correctly under normal operating conditions, we can also simulate behaviors that violate the conditions of the contract. For example, if eShop.com's server fails then various actions are permitted by Muzac.com under the contract such as termination, fines, etc. Simulation aspects are beyond the scope of this paper.

3. BizTalk™: Microsoft's Business Process Integration Strategy

The "BizTalk Initiative" is Microsoft's approach to XML-based business process integration focusing on support for e-commerce. It consists of four major parts: the BizTalk.org community, the BizTalk Framework, the BizTalk JumpStart Kit and the BizTalk Server [2]. Each of these is briefly described below.

Microsoft established the *BizTalk.org* web site in order to foster a community of developers and promote use of standard XML Schemas for common business processes. The web site hosts a library to store and share BizTalk document schemas. It also provides newsgroups where developers can discuss aspects of the technology and help each other use the various tools.

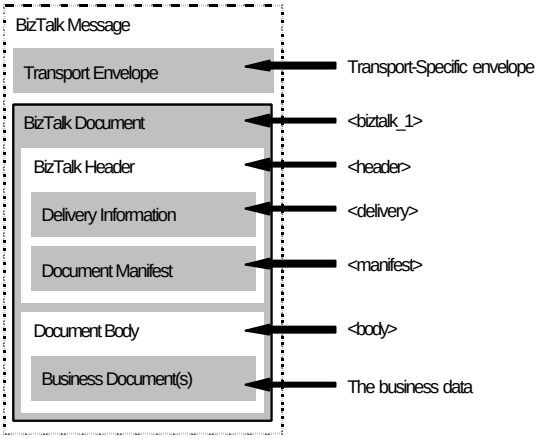

Figure 1. BizTalk Message Structure

The *BizTalk Framework 1.0a Independent Document Specification* is a framework for application integration and electronic commerce based on XML. It consists of a set of XML schemas that define BizTalk Messages, BizTalk Documents and BizTags for routing the messages. *BizTalk Messages* that are the unit of interchange between BizTalk Servers. The structure of a BizTalk Message is shown in . The elements of a BizTalk Message are as follows. At the outer level is a *Transport Envelope*. BizTalk supports a range of transports including HTTP, SMTP, MSMQ, etc. Each of these has own "wrapper." Inside the transport wrapper is the BizTalk Message itself. It is an XML stream formatted according to the 1.0a specification and consists of a BizTalk Header and one or more Business Documents. The header is standard for all BizTalk Messages and contains routing information needed by BizTalk services. It also permits services and applications to determine the types of Business Documents in the message. The *Business Document* is the payload of a message. Also called the *body*, it is an XML stream containing data such as purchase orders, invoices, etc.. The BizTags are fields in the BizTalk Message header

that provide a standard way of specifying routing and handling information such as the destination and source. These tags are used by the BizTalk server for processing the message.

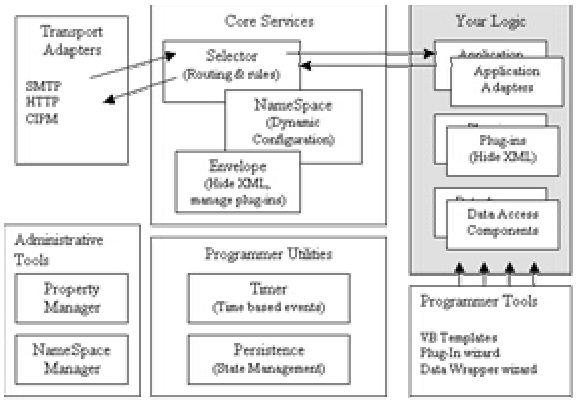

Figure 2. BizTalk JumpStart Kit Architecture

Microsoft introduced the *BizTalk JumpStart Kit* (JSK) to prepare programmers for XML-based application development and as first step toward the BizTalk Server. The JSK provides an environment for developing, and deploying software components that manage BizTalk Messages, database access and business logic. It consists of a number of parts as shown in Figure 2. A set of Transport Adapters and Core Services take care of transporting BizTalk messages and routing them from application source to destination. Administrative tools provide a convenient way to specify how messages are routed between applications. A number of programming support tools are provided. The most important of these are the *JSK.Envelope* object, the Visual Basic (VB) Wizards that aid programmers in generating *Plug-Ins*, and *Application Adaptors*. The JSK.Envelope object is part of the BizTalk library and facilitates handling all BizTalk documents. It is responsible for encapsulating documents as XML messages. The JSK.Envelope is a generic wrapper that knows about the general structure of a BizTalk message and is used in routing messages. The Plug-In is called that because it "plugs in" to a JSK.Envelope object. A Plug-In is generated for each Business Document type using its XML Schema. The Plug-In takes the form of an ActiveX DLL (Dynamic Load Library). It is used to provide the programmer with an easy way to manipulate the data that is specified by the XML Schema. That is, the programmer accesses the data using standard VB data types and does not have to work directly with the XML DOM (Document Object Model). The Application Adaptor Wizard generates a COM+ wrapper that lets the programmer connect the application's business logic with the JSK message routing infrastructure. The Application Adaptors are used to receive the BizTalk Messages (and using the Plug-In) apply the specific business logic.

The *BizTalk Server 2000 Technology Preview release (BTS-TP)* is the current version of the BizTalk Server. BTS-TP greatly extends the XML-based messaging concepts introduced in the JSK. BTS-TP consists of a set of BizTalk Services and Server Components. *BizTalk Services* include receiving incoming documents, parsing the documents to determine their specific format, extracting key identifiers and identifying specific processing rules, and delivering documents to their respective destinations. *BizTalk Server Components* that provide data translation capabilities, organization and trading partner management, server management, and document tracking. These include:

- BizTalk Editor for editing XML schemas.
- BizTalk Mapper for translating between difference message schemas.
- BizTalk Management Desk provides a graphical user interface to manage the exchange of data between trading partner organizations and applications.
- BizTalk Server Administration Console is a Microsoft Management Console (MMC) snap-in used to manage and maintain servers or server groups.

4. Implementing B2B Contracts Using BizTalk

This section describes the implementation of our B2B contract system. Continuing with the scenario above, we give the specification of key business documents and how they are generated and processed. We explain how and where the various BizTalk tools are used to manipulate and manage these documents. The goal of this section is to go into detail on the structure of these documents and their relationships to each other based on the B2B scenario. From the scenario we are concerned with the following three major business document types: Contract, Policy, and Plan. The section is organized accordingly around these documents.

4.1 Business Contract Documents

The contract for eShop.com's services is shown at Appendix A (based on the contract proposed in [3]). Business contracts are written by humans and are usually based on existing contracts that are, in turn, assembled from standard clauses. Therefore we have chosen to express a contract as a collection of clauses. This takes

the form of an XML schema named *B2BContract* as shown in Appendix B. Notice the contract schema contains element and attribute type definitions corresponding to the variables in the contract (shown between square brackets in Appendix A). Also note that the key data structure, the Contract, contains a Preamble, one or more ClauseGroups and Signatures. The latter is possible to be included in the contract schema because we use this schema to represent both the contract forms that need to be filled by specific values of a contract (referred to as contract templates) and the signed contract instances.

A *Contract Template* is the basic document that specifies an instance of a contract and is an instance of B2BContract. Appendix C shows the contract template corresponding to the eShop.com contract. The purpose of the Contract Template is to specify the collection of clauses that make up the contract. Note they are specified as URNs and the contract template identifies the URN of the server on which they can be found. Thus each clause is stored in a separate XML file on a server. An XML Schema called B2BClause defines the clauses themselves. In the current implementation the clause schema corresponds to the element type "clause" as defined in the *B2BContract* schema. Instances of the B2BClause schema, called *Clause Templates*, store the "meta data" associated with a particular clause. This includes the human readable text, specification of the contract variables or fields and an URN that identifies the clause's corresponding Policy file. The motivation for describing contracts as described is to permit flexible generation, efficient transmission and easy manipulation of the various related documents required in our approach – based on that contract representation. Specifically, our scenario starts with a person from Muzac.com completing a contract form on eShop.com's web site. This HTML-based form (as it appears in Appendix A) is generated from the Contract Template by following the URNs to the Clause Templates and retrieving the metadata associated with each clause (e.g. the text and fields). Similarly, the Active Server Page (ASP) that processes the HTML form is generated from the contract template. This ASP takes the values entered by the user into the input form and builds a *Signed Contract Instance*.

A *Signed Contract Instance* is an XML document that is an instance of the *B2BContract* schema. However, the signed contract instance contains only the values of the fields for each clause the user entered. This approach results in a very small XML document that can be efficiently transmitted. All other information associated with the contract can be retrieved as needed. That is, there is no need to ship around the large amount of text that is meaningful only to humans. Other needed documents such as the Business Policy Documents can be assembled based on the signed contract instance. Policy Documents are discussed in the next section.

This completes the description of the business documents related to the first part of the scenario: a signed contract instance is placed on the eShop.com web site. Now we show where the various BizTalk tools provide support for managing these XML documents and building the application that takes the signed contract instance and results in it being placed in the Notary on B2B.com (Figure 3). The main BizTalk tools used are the Plug-in and Application Adaptor from the JSK and the "agreement" for document handling from the BizTalk Server.

The Plug-in greatly simplifies the task of dealing with complex XML documents from the programmer's perspective. They provide an abstraction that lets the programmer access the underlying schema's data model using common Visual Basics data structures. Thus, the programmer uses the Plug-in Wizard to generate a Plug-in for each XML schema defined above. These can then be used to access XML files or to generate XML strings. For example, in the ASP page mentioned above, the user-supplied data is taken directly from the HTML "input" fields and placed into an instance of the B2BContract Plug-in object on the e-Shop.com server. Once all the data is input, the header fields of the BizTalk message are filled in with the address of the sender and receiver. Remember, a Plug-in is always "plugged into" an associated BizTalk Envelope object, i.e. the Envelope objects points to the corresponding plug-in (Figure 3) shows this in the case of an Envelope object that is created for signed contract instance messages that will be sent from the e-Shop to the B2B.com server).

Now, at the e-Shop.com site the signed contract message is retrieved from the Envelope as an XML string and handed over to the BizTalk Server running on eShop.com's machine to route (for simplicity Figure 3 does not show envelope objects and other objects on the e-Shop.com). This is where the BizTalk Server "agreement" mechanism takes over and routes the document to the BizTalk Server running on B2B.com's machine.

Once a properly formatted BizTalk message arrives at the B2B server an application adapter for that message will create an Envelope object which will hold the message arrived - effectively, the message is held inside the Envelope's XMLDOM (Document Object Model). The programmer uses the Application Adaptor Wizard (in Visual Basic) to build a COM+ application object that has interfaces known to BizTalk Server. Specifically, the Application Adaptor generated object has a *ReceiveMessage* interface. The inbound and outbound

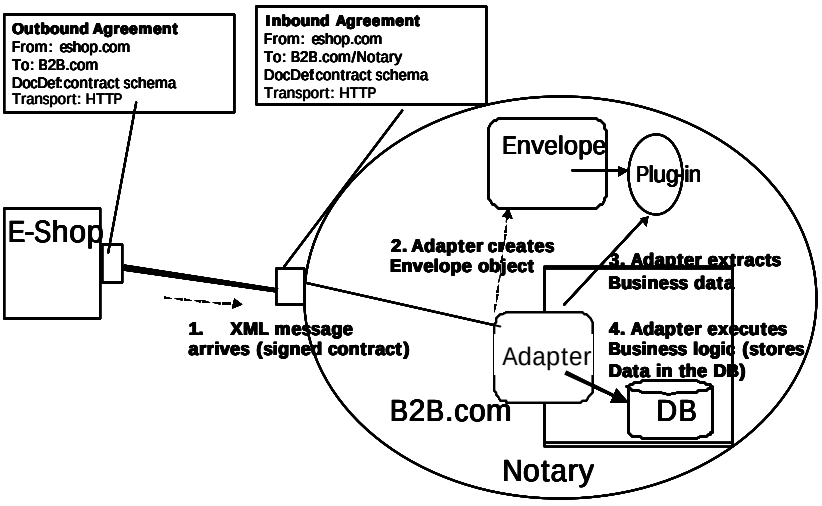

Figure 3. Use of BizTalk infrastructure: Notary component

agreements, linked by a pipeline, tell BizTalk Server what message types to deliver to what Application Adaptor. In this case, the signed contract instance messages are delivered to the Notary COM+ object on B2B.com's machine. Once again, the Plug-in's for the contract schemas are used to aid in processing the XML document. The Notary adapter takes the values from the signed contract instance and places them in a SQL server database. A GUID (globally unique identifier) is assigned to the contract instance and the message is handed off to the Contract Management software for further processing, as described in the following section.

4.1 Business Policy Documents

Once a signed contract instance has been lodged in the Notary it is the job of the Contract Management software running on B2B.com's server to start the process of implementing the contract between the parties. The first step is to assemble the Business Policy Documents for each of the contract clauses. These documents form the basis for specifying, monitoring and enforcing the behaviors of Muzac.com and eShop.com relative to this particular contract. Thus, the policies derived from the contract govern the range of behaviors the parties may take while the contract is in effect.

According to [4], policy is defined as a set of rules related to a particular purpose, and a rule can be expressed as an obligation, permission or prohibition. In order to specify rules we use the notation introduced in [5], which is based on this policy definition:

Rule #: <role> [is] (obligated | forbidden | permitted) [to] [do] (<action> [before <condition>] | satisfy <condition>) [, if <condition>][, where <condition>][, otherwise see Rule <#>]

Currently humans must still write contracts, and so humans must write policies in our approach. Each clause in the contract must be manually (by human) mapped onto one or more business rules (Appendix D shows examples of these rules for eShop.com's contract.) The purpose of these rules is to isolate each of the "terms and conditions" in the contract so they may be translated into Business Plan Documents for implementation, monitoring and enforcement. (This is described in the next section.) Inspection of the rules shows there are two *roles* – Purchaser and Supplier – just as in the contract. Each rule relates a contract variable (e.g. start date) to an action of the role given some conditions.

The XML schema, *B2BContractPolicy*, for the policy language is included in Appendix E. The schema is a direct mapping from the rule language given above to XML elements. Each of the business rules (in Appendix D) is expressed as an instance of this schema. These are called Policy Templates and are stored in files on the B2B server. When combined with the actual values specified in a contract instance these policy templates are instantiated in policy rule instances. The contract management software on B2B.com's server takes the contract instance and processes each clause to assemble the Policy Documents for eShop.com (supplier) and Muzac.com (purchaser). This is done by loading the clause template into its Plug-in and accessing the policy URN that leads to its particular Policy Template. The values from the contract instance are

extracted (from the contract's Plug-in) and placed in the appropriate elements of the policy template rules thus forming a policy instance. The routing information in the BizTalk message header is completed and the policy documents are handed over to the BizTalk server to deliver to the appropriate party.

Now, B2B.com has been employed by eShop.com and Muzac.com to provide services for auditing, monitoring and enforcement of the contract. This takes the form of two additional Policy Documents: one to each of the parties to the contract and one to B2B.com itself. These policies specify what information is required by B2B.com to perform its services. For example, in order to carry out auditing, B2B.com randomly requests that eShop.com send a copy of the last several invoices so as to verify that billing is being correctly calculated. In terms of monitoring, B2B can monitor exchange of messages between trading partners (e.g. sending of purchase orders by Musac.com to eShop.com) or monitor behavior of both the e-shop.com and Musak.com servers. Certain messages, such as notification by Muzac.com of a breach of contract, are always (by default policy) copied to B2B.com. A number of implementations are possible, for example, the use of MSMQ triggers, as described in [3]. In terms of BizTalk Server, each of the rules requiring messages to be sent from the parties to B2B.com is implemented as an agreement. BizTalk Server provides the "Management Desk" which is a graphical user interface for manual input of agreements. BizTalk Server also exposes the underlying COM objects in a library that can be accessed programmatically from Visual Basic. Thus, it is possible to dynamically establish the message channels for B2B to perform its services.

It is important to note that in terms of implementing the contract management software, the process of business rule specification explained above essentially determines the logic design of the underlying business objects. Our goal is to eventually build up a general set of business objects to support a range of B2B contracts and the required contract operations. Even then, when new contracts reveal new rules, the business objects must be extending manually, i.e., by programming. This is the topic of the next section.

4.2 Business Plan Documents

At this stage in the scenario eShop.com and Muzac.com have received their copies of the signed contract instance and the related business policy document. The BizTalk tools that facilitate this are: BizTalk agreements that route the messages from server to server, the Application Adaptors that receive the messages (via the agreements) and the Plug-in that aid the programmer in manipulating the XML-based message data. Now, from this point forward we leave the domain of BizTalk services. That is, BizTalk Server and the JSK do not address the implementation of the business objects and their processing logic. Future releases of BizTalk may offer more support tools or even provide domain specific (e.g. Banking) frameworks, but this is not currently the case. Therefore, aside from BizTalk message routing, our contract implementation is that of a general Windows DNA application.

Continuing with the scenario, having received the policy and contract documents, it is the job of the B2B.com's contract management software (running on eShop.com and Muzac.com servers) to map the specifics of the policy documents into the business objects. To accomplish this mapping the contract management software generates a number of BizTalk messages that we collectively refer to as Business Plan Documents. These documents contain parameters and other specific instructions that implement the behaviors required by the contract. There are two categories of documents: those that deal with implementation of the business logic locally and those that relate to contract monitoring via B2B.com. The local implementation plan messages result in parameters being set on the business object such as the start and end date of the contract, the URN for obtaining the price list, etc. These are parameters to existing business logic that must be developed based on the analysis of business rules as described in the last section.

Time-based and periodic behavior is also derived from the policy document. The *B2BPlan* schema defines a structure for specifying *activities* based on rules from the policy document. The resulting plan is lodged in a component responsible for maintaining and executing the activities. An example of a scheduled behavior is eShop.com sending a bill based on the agreed upon billing interval. Event-based behaviors are also implemented based on the policy document. When Muzac.com receives a bill from eShop.com this results in an event-notification sent to the accounting component to prepare a payment.

Referring to the policy language shown in the last section, the above describes the conceptually straightforward mapping of *obligations* and *prohibitions* onto business objects. *Permissions* imply that local policy may be applied to further refine the business rules contained in the policy documents. This provides a level of autonomy for decision-making at the local level. For example, the contract contains a clause that permits the purchaser to levy a fine on the supplier for non-performance. This type of local autonomous behavior may be deferred to a human for decision-making by local

policy. A default local policy could be to defer approval of permissive policy rules to a human.

In summary, the policy documents are translated into a number of business plan documents that are then transmitted to the business objects that implement the desired logic. In our implementation these objects are COM+ applications. These applications are designed and programmed based on the analysis that determines the business rules as described in the last section. Our current implementation uses the range of Windows DNA support to implement the business objects and contract monitoring. This includes COM+ queued components, COM+ event subscription and notification service, MSMQ triggers and SQL Server stored procedures. Also, we are reviewing *Visual Rule Studio* [6], a Visual Basic Designer that provides true rule-based (forward and backward chaining) application development. We think this tool will permit us to develop a more general approach to implementing business rules and more flexible business objects.

5. Related Work

Our XML-based approach for specifying contract forms is similar to some of the results from the CrossFlow project [7]. We add additional features that reflect the legal aspects of contracts such as description of contracts as a set of obligation policies between trading partners. Additionally, we support a precise description of policies in terms of roles and actions that these can undertake – and express our policy notation using XML. Further, our approach of defining and implementing the roles needed to support contract operations has some similarity to recent IBM work on contracts [8]. Again, our approach is focused on supporting business and in particular legal views on contracts and thus our business contract architecture does not include low-level concerns such as security and transport mechanisms used to support contract operations.

The EU-funded COSMOS project [9] provides a set of services that facilitates the use of e-contracts. Rather than attempting to model the full complexity of contracts as stated in the introduction, their model identifies only those parts that are amenable to efficient automation. Hence, much of the system deals with lower-level, communication and representation issues, rather than more contract-specific issues, though they do provide a basic architecture and a meta-model outlining the structure of a contract. In addition, they provide some tools for contract-specific functionality, such as tools for the contract-negotiation process and derivation of a workflow (based on Petri-nets) from the contract, for its enactment.

6. Conclusion and future work

This paper has outlined our approach of supporting contracting for the sale of intangible goods – MP3 files among the producers of these files and an e-shop where these are made available for sale. We have developed a contract that represents business relationships between these two trading partners. This natural language contract provides the starting point for developing an XML schema for that contract template and XML schemas for contract clauses that constitute this contract. For each of the contract clauses we have then produced one or more business policies that represent refinements of the high-level policy statements from contract clauses. These policies can be used for the monitoring purposes and also as a starting point to implement behavior of the trading partners according to these policies. We have used Microsoft's BizTalk infrastructure to facilitate process of producing XML schemas for contract forms and contract policies and to facilitate exchange of these documents by the components implementing the logic needed for business contracts architecture.

We plan to extend this work by providing a better link between policies that describe obligations of trading partners and the behavior that implements these obligations (business activities). In addition, we plan to further investigate the issues associated with contract negotiation, conflict detection and resolution of the legal rules when composing customized contracts.

7. Acknowledgements

The authors would like to thank Andrew Goodchild for developing the contract shown in Appendix A, initially presented in [3]. We would also like to thank Andy Bond and Kerry Raymond for their comments on the earlier version of the paper. The work reported in this paper has been funded in part by the Co-operative Research Centre for Enterprise Distributed Systems Technology (DSTC) through the Federal Government's AusIndustry CRC Programme (Department of Industry, Science & Resources).

References

[1] 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.

[2] BizTalk Initiative: http://www.microsoft.com/biztalk/

[3] Goodchild, A., Herring, C., Milosevic, Z., "Business Contracts for B2B", CAISE'00 Workshop on Infrastructures for Dynamic B2B Service Outsourcing, Stockholm, June 2000.

[4] ISO/ITU-T Recommendation X.902, Open Distributed Processing, Reference Model - Part 2: Foundations, 1994.

[5] Steen, M. and Derrick, J. "Formalizing ODP Enterprise Policies", Proceedings of Enterprise Distributed Object Computing Conference (EDOC'99), 1999.

[6] Visual Rule Studio, www.rulemachines.com

[7] Cross-Organizational Workflow Support in Virtual Enterprises ESPRIT Project 28635; http://www.crossflow.org

[8] IBM XML specification for business-to-business transactions; http://www-4.ibm.com/software/developer/library/tpaml.html

[9] Griffel, F., et al. "Electronic Contracting with COSMOS - How to Establish, Negotiate, and Execute Electronic Contracts on the Internet", EDOC'98 Workshop, La Jolla, California USA, November 1998

NOTE: Only fragments of the documents and schemas can be shown, as they are each several pages long.

Appendix A: Contract

CONTRACT FOR PORTAL SERVICES

This Deed of Agreement is entered into as of the Effective Date identified below.

BETWEEN [NameAddress]
(To be known as the (Purchaser) in this Agreement)
AND: [NameAddress]
(To be known as the (Supplier) in this agreement)
WHEREAS (Purchaser) desires to enter into an agreement to purchase from (Supplier) [Item] (To be known as (Goods) in this Agreement).
NOW IT IS HEREBY AGREED that (Supplier) and (Purchaser) shall enter into an agreement subject to the following terms and conditions:

1. Definitions and Interpretations

1.1 Price is a reference to the currency of the [Country] unless otherwise stated.
1.2 This agreement is governed by [Country] law and the parties hereby agree to submit to the jurisdiction of the Courts of the [Country] with respect to this agreement.

2. Commencement and Completion

2.1 The commencement date is scheduled as [Date].
2.2 The completion date is scheduled as [Date].
2.3 The contract may be modified by agreement as defined in Section [Section].

3. Purchase Orders

3.1 The (Purchaser) shall follow the (Supplier) price lists [PriceList].
3.2 The (Purchaser) shall present (Supplier) with a purchase order for the provision of (Goods) within [Days] days of the commencement date.
3.3 Purchase orders are to be sent electronically, and are to be interpreted under standards and guidelines outlined in Supplement A [Supplement].

4. Service Delivery

4.1 The (Supplier) shall ensure the (Goods) are available to the (Purchaser) under Quality of Service Agreement [QoS]
4.2 The (Supplier) shall on receipt of a purchase order for (Goods) the (Supplier) shall make them available within [Hours] hours.
4.3 If for any reason the conditions stated in 4.1 (a) or 4.1 (b) are not met, the (Purchaser) is entitled to charge the (Supplier) the rate of [CurrencyUnits] for each hour the (Goods) are not delivered.

5. Payment

5.1 The payment terms shall be in full upon receipt of invoice. Interest shall be charged at [Percentage] on accounts not paid within [Days] days of the invoice date. The prices shall be as stated in the sales order unless otherwise agreed in writing by the (Supplier).
5.2 Payments are to be sent electronically, and are to be performed under standards and guidelines outlined in Supplement B [Supplement].

6. Rejection

6.1 If the (Goods) do not comply with the Order or the (Supplier) does not comply with any of the conditions, the (Purchaser) shall at its sole discretion be entitled to reject the (Goods) and the Order.

Sections 7– 9 are not shown.

SIGNATURES

In witness whereof (Supplier) and (Purchaser) have caused this agreement to be entered into by their duly authorized representatives as of the effective date written below.

Effective date of this agreement: [Date]
[Signature] [Signature]

Appendix B: Contract Schema

```
<?xml version="1.0"?>
<Schema name="B2BContract"
        xmlns="urn:schemas-microsoft-com:xml-data"
                xmlns:dt="urn:schemas-microsoft-
        com:datatypes">
```

Only the Contract Element is shown below. The ClauseGroup contains Clauses.

```
    <ElementType name="Contract">
        <attribute type="ContractID"/>
        <attribute type="ContractRepositoryUrn"/>
        <attribute type="ContractPolicyUrn"/>
        <attribute type="ContractTemplateUrn"/>
        <attribute type="ContractType"/>
        <element type="Preamble"/>
        <element type="ClauseGroup" minOccurs="1"
                maxOccurs="*"/>
        <element type="Signatures"/>
        </ElementType>
</Schema
```

Appendix C: Contract Template

```
<?xml version='1.0' ?>
<Contract xmlns="urn:http://localhost/uB2Bu/B2BContractSchema.xml"
        <ClauseGroup ClauseGroupID="1." ClauseGroupTitle="Definitions and Interpretations">
                <Clause ClauseID="1.1" ClauseUrn="B2BClauseTemplate0002.xml">
                </Clause>
                <Clause ClauseID="1.2" ClauseUrn="B2BClauseTemplate0003.xml">
                </Clause>
        </ClauseGroup>
```

ClauseGroups 2 – 9 are not shown.

```
        </Contract>
        </body>
</biztalk_1>
```

Appendix D: Contract Policies

Preamble:Purchaser is obligated to Purchase(Goods) Where(Goods=Item) Otherwise Rule 7.1
Supplier is obligated to Supply(Goods) Where(Goods=Item) Otherwise Rule 7.1

1.1 Purchaser is obliged to Use(Price) Where(Price = DenominatedIn(County)) Otherwise Rule 7.1
Supplier is obliged to Use(Price) Where(Price = DenominatedIn(County)) Otherwise Rule 7.1

1.2 Purchaser is obliged to Respect(Law) Where(Law = County) Otherwise Rule 7.1
Supplier is obliged to Respect(Law) Where(Law = County) Otherwise Rule 7.1

Rules 2–7 are not shown.

Appendix E: Policy Schema

```
<?xml version="1.0"?>
<Schema name="B2BContractPolicy"  …/>
```

Some attribute and element types are not shown.

```
<ElementType name="PolicyRule"  content="eltOnly" model="closed">
<attribute type="RuleNumber"/>
<attribute type="RuleGUID"/>
<attribute type="RuleClauseID"/>
<element type="RuleRole"/>
<element type="RuleConstraint"/>
<element type="RuleAction"/>
```

```
<element type="RuleWhere"/>
<element type="RuleCondition"/>
<element type="RuleOtherwise"/>
</ElementType>
<ElementType name="ContractPolicy"  content="eltOnly" model="closed">
<element type="PolicyRule" minOccurs="1" maxOccurs="*"/>
</ElementType>

</Schema>
```