Business Service Interface Structure

M. Vranić, D. Pintar and Z. Skočir
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, CROATIA
E-mail: mihaela.vranic@fer.hr, damir.pintar@fer.hr, zoran.skocir@fer.hr

***Abstract*— EbXML is the emerging standard intended to facilitate efficient and flexible B2B collaboration. One of the crucial issues is the implementation of Business Service Interface (BSI). BSI is an intermediary between other companies and internal business applications of a company that is engaged in electronic business. BSI is not directly specified. Its behavior and its assignments are specified indirectly through other specifications. This paper discusses and describes general structure of BSI so it would comply with all the requirements which are also discussed. BSI would enable companies to engage in electronic business collaborations according to the previously specified business processes that are agreed with other party. It would enable conducting business with minimum human interference. BSI would also enable human control of delicate business transactions.**

I. INTRODUCTION

EbXML consists of a set of specifications which form a framework for conducting e-Business services [1]. This set of specifications describes mechanisms for registering the corporation into an ebXML Registry/Repository [2], finding a business partner and the actual business collaboration. EbXML vision is to form a global electronic market place where enterprises of any size, anywhere can find each other electronically and conduct business through the exchange of XML-based messages. This framework could be very important for developing countries like Croatia which have 90% or more of their economy in SME[1] [3]. Although BSI[2] has an extremely important role in the actual collaboration, its specification as such is not standardized. Its main tasks and functions are described through the different ebXML specifications.

The general functions of the BSI are universal and don't change from corporation to corporation. However, since different corporations have different internal business systems, the BSIs have to be different on the implementation level. Our goal was to generate a BSI structure that would comply with all of the requirements and would enable easy reuse of most of its parts.

The functional requirements on BSI are given in section II. Our view of the actual implementation, which will comply with the requirements, is presented in section III. Section IV describes crucial BSI module 'Business Collaboration'. Conclusion given in section V is followed by references.

II. BSI FUNCTIONAL REQUIREMENTS

BSI functional requirements are superimposed from different ebXML specifications. BSI itself should play a role of an intermediary between the internal business applications of some corporation on one side, and its business partners and the ebXML Registry/Repository itself on the other side [4]. This is clearly visible in the Fig. 1.

For further understanding some terms must be clarified. CPP (Collaboration-Protocol Profile) is a document that describes specific capabilities of one trading partner to engage in electronic message exchange with other partners. CPA (Collaboration-Protocol Agreement) is a document used to specify the interaction between two trading partners. CPA is derived from the intersection of two or more CPPs. CPP and CPA are specified in [5]. CPA forming is described and discussed in [6].

BSI in specific interaction with the other party should behave as is defined in agreed CPA.

It is important to differentiate the phase of registering to the ebXML framework from the phase of communicating with other organizations with purpose of conducting e-business. Each phase has different requirements for BSI use. Since registering phase happens only at the beginning of the ebXML[3] use and could be the same for different companies (because it does not depend on internal business applications), universal applications (tools) could be provided for accomplishing tasks for that phase [7]. Much more delicate is the phase of communicating with other parties since its actions have legal and economic consequences for the company conducting e-business.

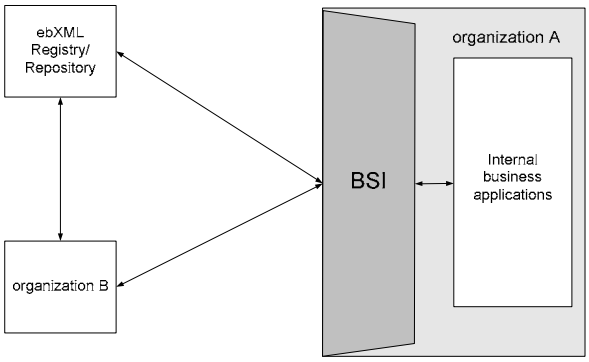

Figure 1. BSI as an intermediary between internal business applications and external entities

[1] SME- small and medium-sized enterprises

[2] BSI- This abbreviation for Business Service Interface will be used further in the paper

[3] Organizations can afterwards change its CPPs, business processes definition, and what is most often expected, organization can add new definitions of processes it supports.

UML Use Case diagram, presented in Fig.2., shows a number of use cases which BSI has to support. First four tasks (functional requirements) are part of the registration phase of using ebXML. Last four are part of run-time phase and take much more time and effort to implement. Task 'CPA negotiation' could be solved generally, equally for all companies, like tasks from first phase. Accordingly, we expect that applications which will enable easy forming and monitoring of CPA forming will be available for download and use from the ebXML site.

Our main concern in this paper will be the last two functional requirements shown in Fig. 2: BSI adaptation to agreed CPA and business collaboration.

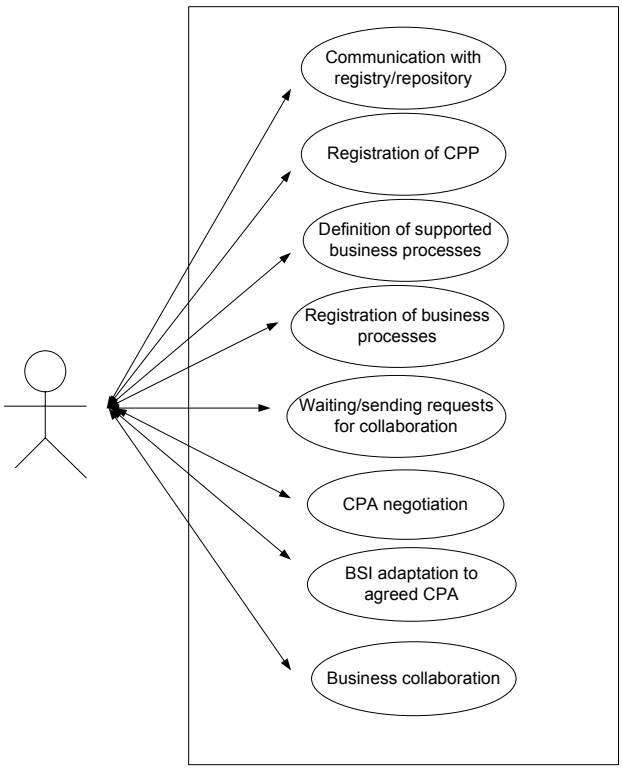

Figure 2. Use Case diagram – functional requirements that BSI has to support

III. BSI STRUCTURE

Prior to designing of BSI, we have to define its input and output and make some assumptions about internal business applications. We will concentrate on BSI adaptation to agreed CPA and business collaboration itself.

After negotiation with the other business partner, it is assumed that CPA is agreed and that this is undoubtedly known by both parties. Agreed CPA represents electronic handshake between parties. CPA specifies which business process will take place and which business documents must be exchanged. Those messages must comply to agreed safety and other terms [8].

Business processes in ebXML must be defined as specified in BPSS[4] [9]. For design of the business processes a special design tool could be used [10]. EbXML does not specify the appearance of business documents. However, since it has to communicate with internal business applications, BSI must be able, if not to interpret those documents then at least transfer them to a form that internal applications recognize. That is why we assumed that documents exchanged between organizations will be standardized. Open Applications Group gathers many worlds biggest companies and we expect that its standardized forms of business documents [11] will be mostly used. Our solution requires standardization of exchanged documents but the chosen standard does not affect our BSI structure.

The input of BSI are standardized business documents. They are transformed to an output readable by internal business applications. Output could be some kind of electronic business documents, electronic spreadsheets or some other electronic data representations. On the other side, BSI must transform nonstandard output of the internal business applications to specific standardized documents expected by the other party in collaboration.

For the internal business applications some assumptions must also be made. We assume them being able to take care of accounting and can respond to some electronic requests as a whole. They must be able to read and generate some electronic form of business documents which organization declared in business processes.

These internal business applications can be any of the following systems of modules: Oracle Applications, SAP business applications, BAAN business applications or any other proprietary business application solution.

BSI has to enable business communication between different corporations with different communication requirements. The parallel execution of a number of different business process is implied. The behavior of BSI in one communication should not affect the behavior of the other business collaborations which are being concurrently executed. Our solution is that every business communication should be supervised by an instance of the "Business Collaboration" class. The relationships between different elements are shown in Fig. 3.

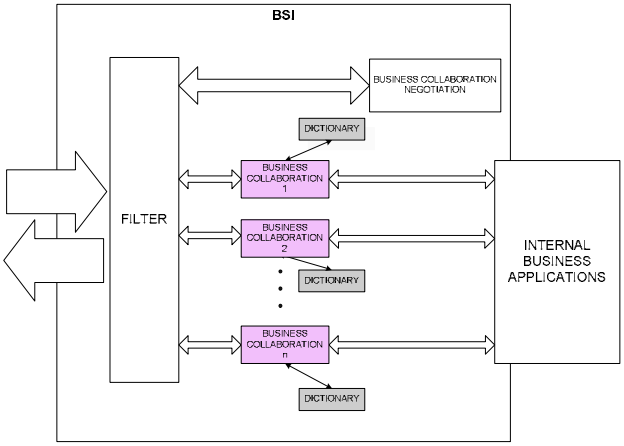

Figure 3. Relations between BSI elements and 'Business Collaboration' as a main actor

BSI must recognize the type of the message received; whether it be a negotiation message or a part of an ongoing business collaboration. The "filter" is made just for that purpose – to forward the message to a correct instance of the "Business Collaboration" class or a part of the application responsible for negotiating. In the following text, the term *Business Collaboration* is used in the meaning of "instance of the 'Business Collaboration'

[4] BPSS – Business Process Specification Schema

class" which main function is to execute a defined, established business process and to manipulate the data used by the internal business applications.

Internal business applications commonly use an already established terminology which frequently doesn't exactly correspond to the standardized terms. This is especially evident when trying to build a standard BSI which would be used by corporations from multilingual backgrounds. We are proposing a solution for that problem by introducing the idea of a *Dictionary*. BSI would consult the dictionary and translate the standardized terms into terms easily understood and used internally by a corporation.

The following example is a part of a Dictionary which translates some of the more common business terms used in [11], into terms used in a Croatian consortium Agrokor d.o.o.

```
<Dictionary>
        <Term>
            <OAGIS>Sender</OAGIS>
            <Agrokor>Pošiljatelj</Agrokor>
        </Term>
        <Term>
            <OAGIS>Receiver</OAGIS>
            <Agrokor>Primatelj</Agrokor>
        </Term>
        <Term>
            <OAGIS>Task</OAGIS>
            <Agrokor>Zadatak</Agrokor>
        </Term>
...
```

This example shows a very simple type of dictionary. Additionally, a more complex type of dictionary could be devised, one that would, for instance, use XSLT (Extensible Stylesheet Language Transformations) to produce a completely different data-structured business documents from the input of standardized documents used by the ebXML architecture and specified business process.

Dictionary doesn't necessarily have to be in an XML form. It could also be entered in the form of a relation database, or any other conceivable form with which a *Business Collaboration* can communicate.

Depending on a choice of a business partner, different dictionaries can be used. Building of a dictionary is a delicate process in which many people from different business backgrounds have to collaborate, especially those who are actually using the internal business applications.

IV. BUSINESS COLLABORATION MODULE

In order to clarify the behavior of BSI Fig.4 shows structure of a Business Collaboration module in more detail.

Business Collaboration has two main parts;

- one which is independent of the internal business applications and which could be used by any corporation which implements ebXML architecture
- the other which should be tailored to each and every company separately depending of its requirements and its existing internal business applications

For better understanding every part of module 'Business Collaboration' will be explained.

A. Sending/Recieving module

This module is responsible for the ebXML messaging. It packages/unpackages the ebXML messages, communicates with lower data layers which route the messages to their correct destination, and concerns itself with security and other agreed terms of communication.

B. Process Flow Control

This module has built-in knowledge of BPSS and can understand processes defined by this specification.

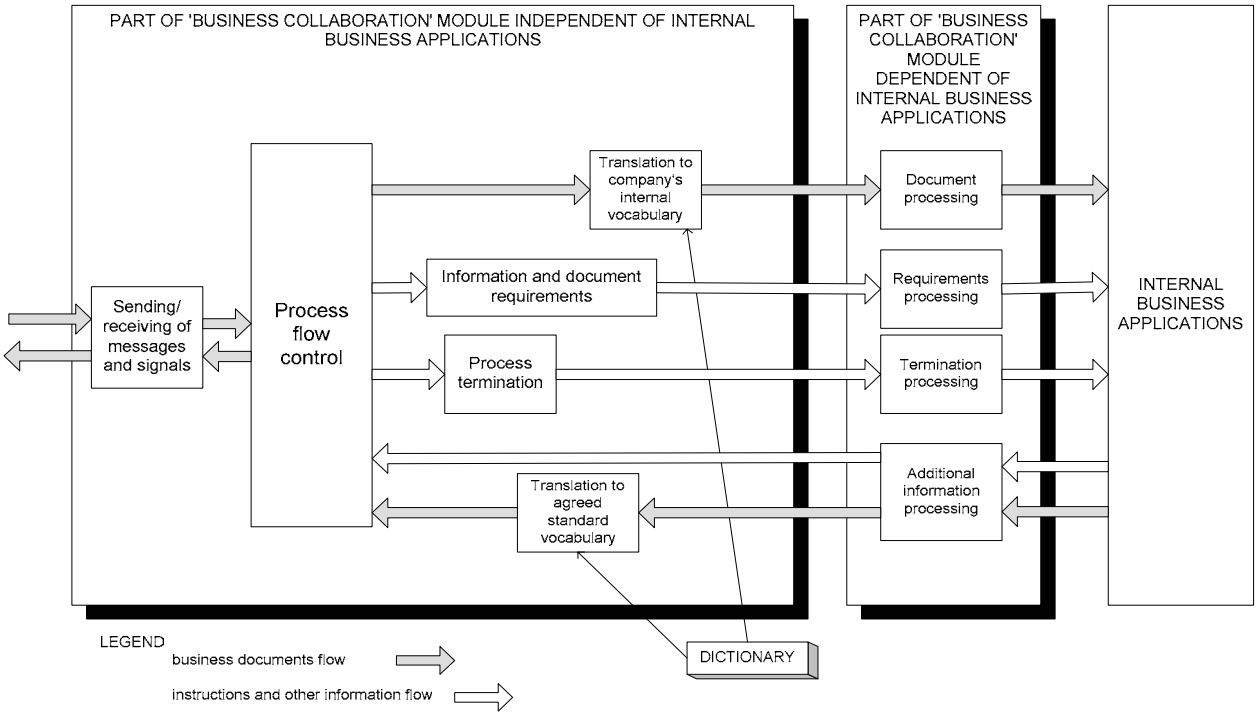

Figure 4. Structure of a 'Business Collaboration' module

It monitors the flow of the business process through the messages and signals received, and also controls the sending of messages and signals. In other words, since it knows the choreography of the business collaborations used in the business process it can expect what messages should be received and detect if any errors occur.

C. *Process termination modules*

Process Flow Control can detect if something has gone wrong with the execution of the business process (wrong type of message, exceeded timeout, etc.). When this occurs, there has to be some sort of mechanism that will rollback the parts of the process which have become unwanted, of course if that is possible for specific process. The Process termination module is responsible for communicating with internal business application when it comes to these matters. It has to send information about what went wrong, and has to suggest a scenario for the previously mentioned rollback. This module has its counterpart in the part of the BSI dependent of the internal business applications, which translate the standardized error and rollback scenario information to the specific tasks that internal applications have to execute.

D. *Translation modules*

There are two such modules:

- the first translates standardized business documents to a form used by internal business applications
- the second takes internal documents as input and transforms them into a standardized form which is defined by the ebXML business process

Both modules do essentially the same thing, although in reverse. They both question The dictionary for the correct translation of the terms.

Of course, it is impossible to build such modules that will be the same for the use in every corporation, with all sorts of different internal business applications. It is assumed that the part of the BSI which is customly tailored for the internal business applications will have to do some extra information processing to transform the data into a format used by the internal applications.

E. *Information and document requirements module*

Previously described modules are taking care of business documents and their transformation to required form. However, Business Collaboration module must inform internal business applications of what is expected of them to do with business documents. Certain information and business documents are also required as an output of internal business applications. Information and document requirements module serves for that purpose.

V. CONCLUSION

BSI has a crucial part in ebXML B2B architecture, acting as an intermediary between internal business applications and external business entities. If we don't take into account the initial tasks of registering the corporation into the ebXML system, BSIs main functions are:

-negotiating about future business collaborations

-controlling the execution of the business collaborations

These two functionality sets can be implemented apart from each other, in the way that after negotiations a new instance of the "Business collaboration" class is created. This instance has to control all the aspects concerning the specific collaboration it was created for.

BSI has to contain the knowledge about the specification of the business process it executes. These specifications are described in BPSS. The corporations also have to decide on the types of business documents they want to exchange. There is a big range of widely accepted standardized OAGIS documents to choose from, and the corporations have to decide which are best suited for their business needs.

The BSI structure presented in this article supports all the functional requirements needed. It is important to notice that the BSI consists of two distinct parts – one, which is universal for all the corporations, and the other, custom-made depending on the internal business applications system. The universal part can be implemented once and than be distributed to all the members of the ebXML business system. The custom-made part takes more time and effort to implement because it has to be tailored to fit with the internal business applications. There is a way to go around it – if the corporations use a widely accepted system of business applications (Oracle Applications, SAP, BAAN) then a sort of a universal BSI "subset" can be implemented so it can be more or less readily used in a number of corporations using the same type of internal business applications. This "subset" can be produced by the same company that created the business applications themselves, or the interested third party. The companies that use less known internal business applications, or use a number of different modules created by different creators are put in the most difficult position though, since they would need the most time and effort to produce a fully capable BSI for their ebXML B2B requirements.

REFERENCES

[1] http://www.ebxml.org

[2] Ivan Magdalenić, Ivo Pejaković, Zoran Skočir, Mihaela Sokić, Marina Šimunić: "Modeling ebXML Registry Service Architecture", Proceedings of the ConTEL2003, Zagreb, June 2003, p.p. 543-550.

[3] Ivan Matasić, Zoran Skočir: "EbXML as Developing Country e-business Strategy Proposal", Proceedings of the SOFTCOM2002, Split-Venice-Ancona-Dubrovnik, October 2002, p.p. 186-194.

[4] UN/CEFACT-OASIS: "ebXML Technical Architecture Specification v1.04", February 2001.

[5] UN/CEFACT-OASIS: "Collaboration-Protocol Profile and Agreement Specification v2.0", May 2001.

[6] Mihaela Vrhoci, Dubravka Đelmiš, Jarmila Maksimović: Collaboration Protocol Profile (CPP) and Collaboration-Protocol Agreement (CPA) in B2B transactions, Proceedings of the MIPRO2002 conference (Electronic Commerce), p.p. 45-50, Opatija , 2002.

[7] Marko Topolnik, Damir Pintar, Ivan Matasić: "Implementation of the ebXML Registry Client for the ebXML Regisrty Services", Proceedings of the ConTEL2003,Zagreb, June 2003, p.p. 551-556.

[8] Ivan Magdalenić, Ivo Pejaković, Dražen Pranjić: Business documents presentation and interchange using SOAP, Proceedings of the MIPRO2002 conference (Electronic Commerce), p.p. 13-17, Opatija, 2002.

[9] UN/CEFACT-OASIS: "Business Process Specification Schema v1.01", May 2001.

[10] Ivan Matasić, Damir Pintar, Zoran Skočir: "A Tool for e-Business Process Definition", Proceedings of SOFTCOM2003, Split-Venice-Ancona-Dubrovnik, October 2003, p.p. 440-444.

[11] http://www.openapplications.org