Technische Universität München
Forschung und Lehreinheit Informatik III
Prof. Rudolf Bayer PhD and Prof. Dr. Donald Kossman.

Hauptseminar Web Services.

„Microsoft.Net and BizTalk. “

By Roger Riff.

Referent: Roger Riff, riff@in.tum.de
Ansprechpartnerin: Dr. Angelika Reiser
Betreuer : Bernd Reiner
Abgabetermin: July 03rd, 2003
Vortrag Termin: July 10th, 2003

OUTLINES

P.S: If an Axterix * appears on top of a concept, it means that concept exists in the given Glossary of this dissertation.

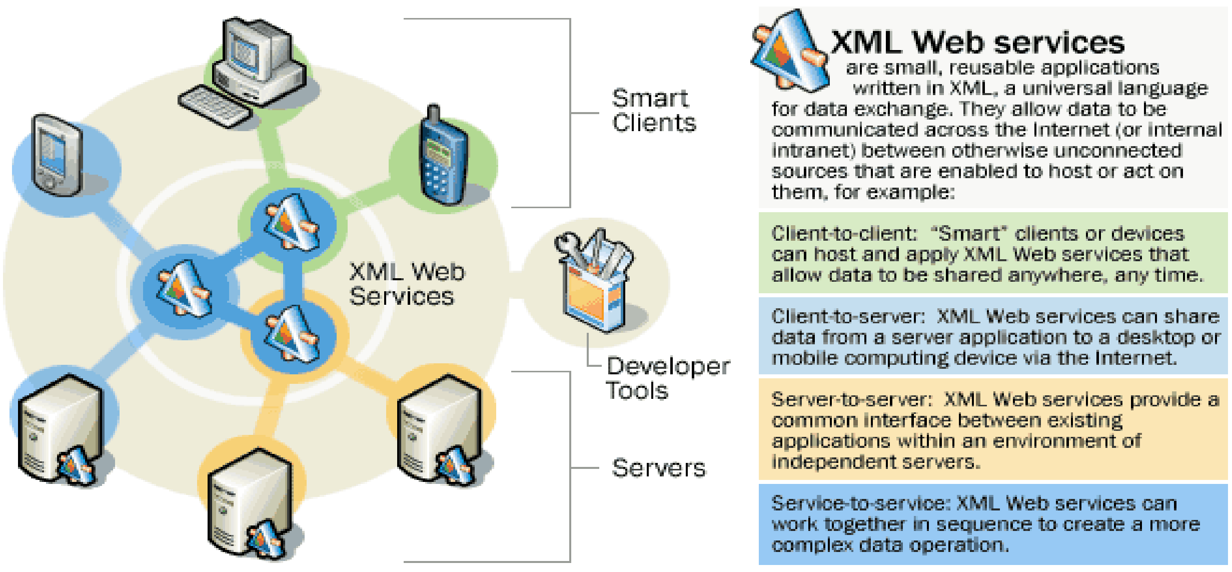
1. Introduction

Almost three years have passed since Microsoft announced the .NET strategy at the Software Forum 2000, and it has generated a tremendous amount of excitement. If one asks a handful of the IT people what .NET is, one is sure to get a variety of answers. Some people might mention aspects such as software as a service, Web Services and XML everywhere. Others might mention the Common Language Runtime (CLR), and the Microsoft Intermediate Language (MSIL). Though that not many will list all the same .NET features, one feature nearly everyone will include is Microsoft's new language, C_Sharp (C#).
But what Microsoft first tries to do here is to present itself as a Solution Provider in Web Services development, in order to make the Internet programming easier than it was years ago by introducing the .Net Product and to make E-Business become a Reality by introducing the BizTalk Features. This advanced new generation of softwares melds computing and business transactions in a new way, offering every developer and business the tools they need to realize every aspect of their computing experience. For the first time, it enables Developers, Businesses and Consumers "***to harness technology on their own terms***".

2. Microsoft .Net

Developing Web Services are based on applications written in XML, - the language for data exchange - , in order to allow datas to be communicated across the Internet. Microsoft tries to base his .Net's features on a communication following the principle of the Peer-to-Peer* communication contrary to the Server–centric* model like before. That means here the Client-to-Client; Server-to-Client, Server-to-Server, and Client-to-Server applications are supported. .Net then exposes a Software technology for connecting information and devices, and for enabling a high level of software integration through the use of Web Services.

The graphic below shows how Microsoft presents .Net with his principal basics.



XML Web services are small, reusable applications written in XML, a universal language for data exchange. They allow data to be communicated across the Internet (or internal intranet) between otherwise unconnected sources that are enabled to host or act on them, for example:

Client-to-client: "Smart" clients or devices can host and apply XML Web services that allow data to be shared anywhere, any time.

Client-to-server: XML Web services can share data from a server application to a desktop or mobile computing device via the Internet.

Server-to-server: XML Web services provide a common interface between existing applications within an environment of independent servers.

Service-to-service: XML Web services can work together in sequence to create a more complex data operation.

Figure1: .Net Basics.

2.1. What is Microsoft .Net?

Simply put, we can define Microsoft.Net as **Microsoft's XML Web Services Platform**. Microsoft.Net contains all that's needed to build and run software based on XML. .Net solves several problems underlying software developments today: Interpolarity, Integration and Application extensibility which are too hard and too expensive. That is why Microsfof.Net reliance on XML - an open standard managed by the World Wide Web Consortium (W3C) - removes barriers to data sharing and software integration.

The Integration and the Application challenges are compounded by the sea of competing proprietary software technologies that plague the Industry. .Net is built on open standards and embraces all web programming languages.

And talking about End users which evokes the problem of Interpolarity, what they (End users) experience when using their software is normally not simple or compelling enough. End users feel frustrated because they are unable to share datas among their application, or act on information when they need it or want to access it. This is a bit the same with Web sites which provide not enough value to users, due at least in part to the fact that their applications and services don't play well with others and act as islands of information that aren't connected. But XML makes it easy to exchange data, and .Net Software gives them the ability to work with datas once it's received, enabling aggregation of value and services from multiple sites and companies into coherent experiences for users.

2.2. Benefits of .Net

Microsoft.Net provides many benefits to **Programmers**, to **Businesses**, to **IT Departments**, and to **Consumers.**

Programmers are relatively scarce and expensive. .Net makes programming easier, maximizing the return on programming investments. Developers can build reusable XML Web Services instead of monolithic applications. XML Services collaborate through XML messaging, but are independent, so modifying one won't break another. Because XML service can be part of many .Net Experiences, updating a module effectively upgrades all it's related. And since XML Web services can be written in virtually any programming language (including C, C++, Visual basic, COBOL, Perl, and Java…), the programmers can use the language they are most familiar and productive with in while retaining the ability to debug across services or components written in different programming languages. Microsoft.Net then reduces the amount of code the programmer has got to write. One XML Web Service works with all devices, eliminating the need to write a different version for every device. Uncoupling the display characteristics from the .Net Experience makes it easy to add new interface technologies, like speech and handwriting recognition, without needing to rewrite the application.

Microsfoft.Net also creates new **Business Models** by allowing a company to commercialize its expertise in new ways. For example, a telecommunication's company could expose access to voicemail and Caller ID as XML Web Services that would allow users to get to them from inside an instant messaging application, email, or another messaging aggregator of the user's choice. Technology vendors can migrate their current software packages to become XML

Web Services and then sell those services to third parties who want that functionality or to .Net's experience vendors building a new software package.

Microsoft.Net also allows **IT Departments** to tap others vendors XML Web Services for expertise and outsourced services, reducing internal costs and expanding the capabilities they can deliver to their customers:

And .Net redefines **End Users (Consumers)** friendly. End Users don't control their personal informations and datas when working in the internet, leading to privacy and security concerns. With Microsft.Net, they can move around an intelligent, personalized internet, which remembers their preferences and delivers the appropriate data at the appropriate time to any smart device they choose.

2.3. Changes for Users, Developers and Businesses

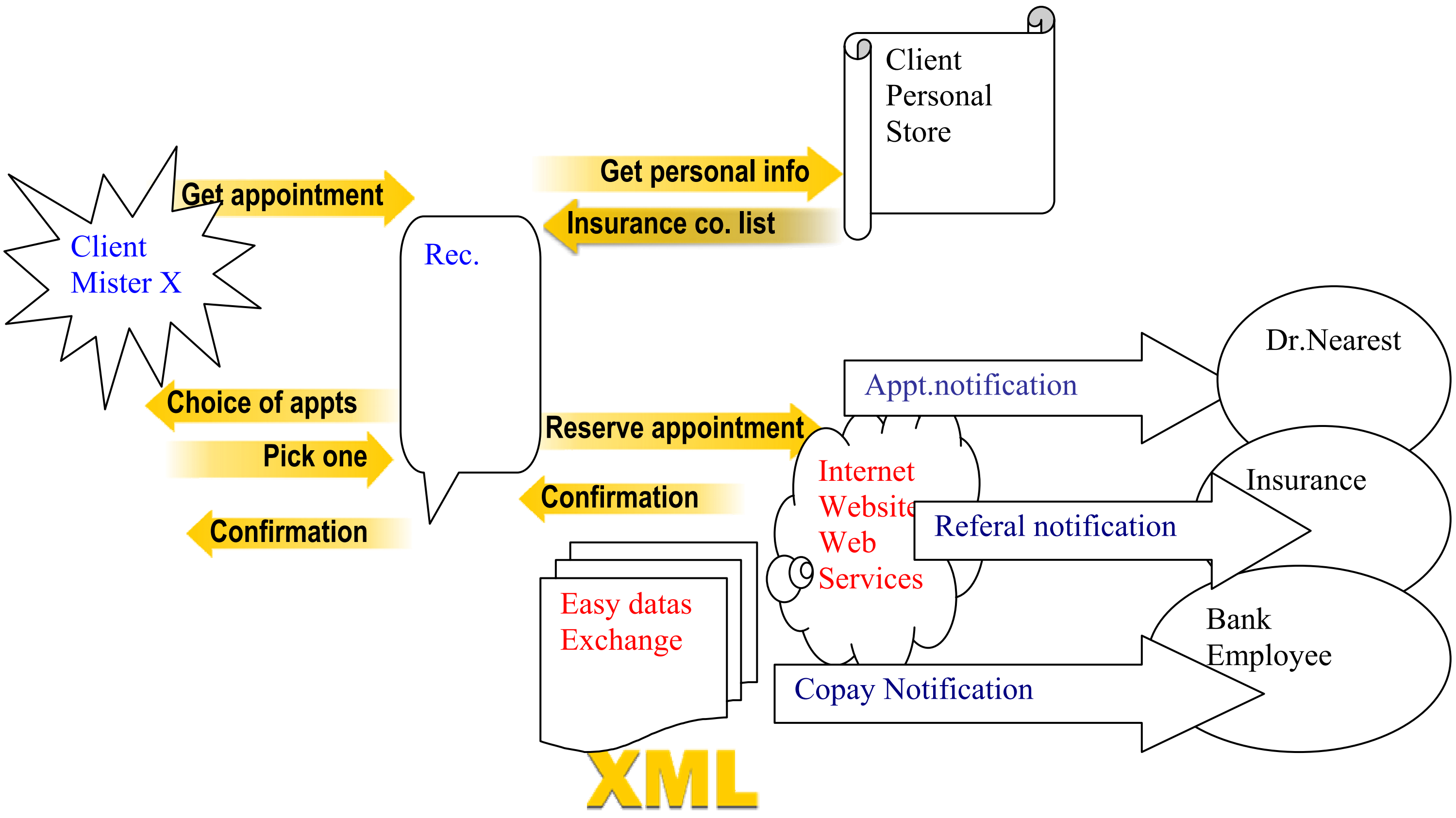
As already mentioned, Microsoft will fundamentally bring a change in the way we think and use our computers. Right now two concepts are important: **the Server and the Desktop**, which dominate the computing. But because .Net is a distributed paradigm, it does not use the traditional distinction between Desktop and Server; instead, processing occurs wherever it makes the most sense, whether that is on a Server, PC, Handheld or other smart devices**.** We ready know that .Net's computing model affects Businesses .End Users, and Developers but in different ways. The following example tries to illustrate how those different ways look like in real life

2.3.1. Scenario of a Web Service based on .Net's principles

Our Client (End User) here is Mister X and there is this Secretary-Receptionist called Monneypenny. The client Mister X while leaving the house one morning, trips getting off the escalator and he severely twists his ankle. Through the pain, he manages to call Dr. Gut's office, where he gets Moneypenny, the receptionist .After he confirms his identity on the phone, he gives Moneypenny the permission to access his location and information so that she can find an orthopaedic clinic near him. Moneypenny can determine which clinic has opening, how far they are from him, and whether it is covered by his insurance. After X gives her permission by touching the appropriate buttons on the smart phone, she can make an appointment and he jumps off the next taxi to that nearest clinic. Meanwhile the cure fees have to be paid from the Bank to the Insurance. Since Mister X has marked personal charges, the Bank company looks up his profile and generates a bill that goes straight to him-personally-, using the method he has speficified (In this case for example , he prefers a direct withdrawal from his checking account. But he also has requested a physical copy of his statement with the Insurance charges on it so he has documentation on the medical expense for his taxes). Based on these options, the two enterprises (Bank and Insurance) exchange all the informations easily because of the fact that those datas exchanges are made easily through XML.

The following diagram tries to illustrate the orchestration of this scenario example and exposes the tasks brought and done by the Internet based Web Service.

Abbreviations: Appt, appts are for Appointment. Rec. is for the Receptionist Moneypenny



From the **Enterprise point of view**, .Net can handle many tasks automatically, freeing up an employee's time. By linking systems and Web Services through XML, data exchange is significantly easier and processing that datas requires little and less effort through Smart devices. What the **Users** and the **Businesses** are not doing here is filing some forms to make the transactions. And what the **Developers** will now deploy to develop such Web Services is in hours, not weeks, because of the outstanding scalability and the transparent interpolarity that .Net offers. For **End Users**, the changes will produce unparallel access to a dramatically more personal, integrated computing experience. With **Businesses**, it changes the way they build software and sell their products.

2.3.2. What stays the same though?

Although Microsoft.Net brings about some radical changes in computing, many things will remain the same. End Users will still work with familiar interfaces within Microsoft products like Microsoft Office. This reduces retraining costs and means that End Users can begin using .Net enabled software immediately. Hardware will still run operating systems like Windows, UNIX, Windows CE, and PalmOs. Because XML Web Services only communicate with devices through XML, any smart device* can use an XML Web Service. Developers will still use their preferred programming language

2.4. Microst.Net Architecture

As long as .Net Programmers write XML Web Services, rather than focusing on standalone applications for servers and clients they can snap together those services in order to build a software solution that performs a particular task.

In order to achieve a XML Web service software development, one normally needs:
A Software Platform *to build new type of personal, integrated user experience.*
A Programming model and Tools *to build and to integrate XML Web Services.*
A Set of programmable services *to provide the foundation for applications and Services.*

Microsft.Net strategy delivers those concepts and .Net includes in his Architecture:
The .Net Platform: This is a set of programming tools and infrastructure to enable the creation, deployment, management and aggregation of XML Web Services.
The .Net Experiences: Which are the means for End Users to interact with .Net.

2.4.1. The .Net Platform

The .Net Platform comprises the tools one needs to create and run XML Web Services. It exposes four main Components. that are connected each to other following their roles and tasks .The Figure 2 below pinpoints those connections with regard of their properties.

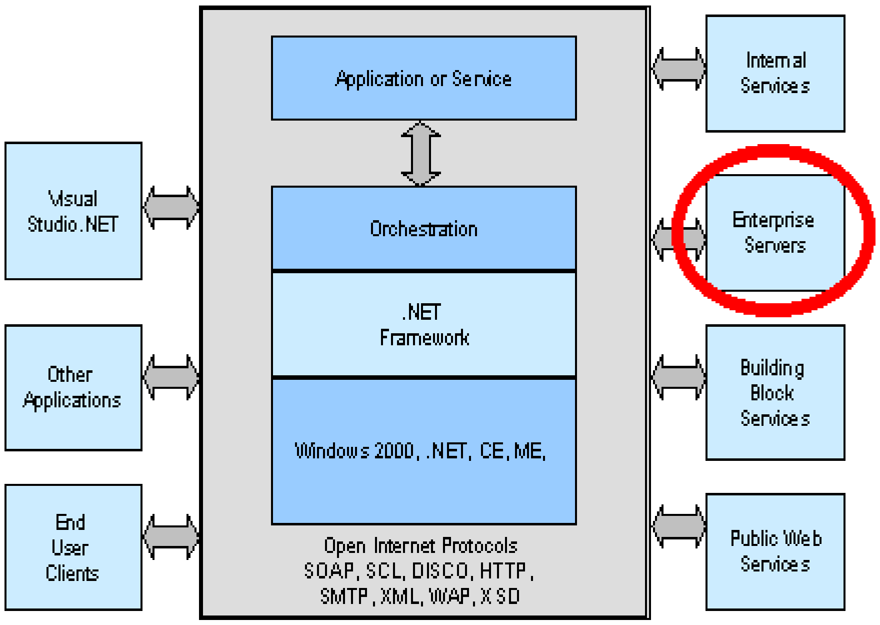


Figure2: .Net Platform connectivity.

The following table specifies the main four components of the .Net Platform.

.Net Platform Components.
.Net Framework and Visual Studio.Net: These are the developer's tools used to build applications and XML Web services. The *.Net Framework* is the set of Programming interfaces at the heart of the .Net Platform. And *Visual Studio .Net* is a suite of Language independent developer tools.
.Net Server Infrastructure The infrastructure to operate XML Web Services relies on the windows operating system and the .Net Enterprise Servers, as suite of E-businesses infrastructure applications that run XML Web Services. These servers include: Application Centre 2000 to enable scale-out solutions. BizTalk Server 2000 to create and manage XML-based business process orchestration across applications or services. Microsoft Host integration Server 2000 for accessing main Frame-based data. Microsoft Mobile Information 2001 Server to enable use of applications by mobile devices like cells phones and other devices. SQL Server 2000 to store and retrieve structured XML datas.
Building Block Services (BBS): The BBS are an integrated set of XML Web Services that make it possible to focus on users and enable user control of a data. They include Passport (*for user identification*) and services for message delivery, file storage, user-preference management, calendar management, and other functions. Microsoft will offer a few BBS in areas that are critical to the infrastructure of .NET, and a wide range of partners and developers will expand the set of BBS. We will also see corporate and vertical building block services built on the .NET Platform.
.Net Device Software: Consisting of Windows XP, Windows ME, Windows CE, Windows Embedded, the .Net Framework and the .Net Compact Framework, this software enables new breed of Smart Internet devices* (*Smart about you*, Smart about the network*, Smart about the information**.....), laptops, and workstations to operate in the .Net universe.

<u>2.4.2. The .Net Experiences</u>

End Users interact with their software through .Net Experiences, which deliver a new type of interaction: a dramatically more personal, integrated experience derived from connected XML Web Services, and delivered through the new breed of smart devices. On a technical point of view, .Net Experiences are a combination of XML Web Services. Microsoft is transitioning four products into .Net Experiences.

Microsoft Product and Enhancement through the .Net Experiences.
Microsoft Office The traditional functionality of office plus features through the Web Services, such as SmartTags – Inline contextual information for key pieces of data.
MSN (Microsoft Network, MSN Messenger Hotmail for example) A complete experience for consumers, including access to services from a wide variety of companies
Microsoft bCentral *Small Business portal* which is a rich experience for small businesses including the ability to mange small business finances and inventory and automatically use services such as eBay as selling mechanisms
Microsoft Visual Studio (Deployment system) A richer environment for developers by integrating access to live MSDN's datas for corporate coding standards.

2.5. Creating Web Services with .Net

As developer, we can create XML Web Services with .Net by using the .Net Framework or directly the Visual Studio .Net

The .Net Framework as the Visual Studio .Net consists of three main parts:

The Common Language Runtime **(CLR)** *which can interpret all programming languages.*
An Unified hierarchical Class Library *that includes a revolutionary advance to Active Server Pages* ***(ASP.NET*)****, which is an environment for building smart client applications (WindowsForms).*
A loosely-coupled Data Access Subsystem (ADO.NET*) *to help connect application to Databases.*

The Visual Studio.Net provides a complete development environment for building Web services on the Microsoft .Net Platform. Using Visual Studio.Net, developers can create reliable, scalable applications and Web services faster than ever before, leveraging existing systems and skills. It then exposes everything one needs to write, build, test, and deploy .Net applications (that means documentation, samples, and command-line tools and compilers.).

Figure 3 shows the Visual Studio .Net contextual for instance.

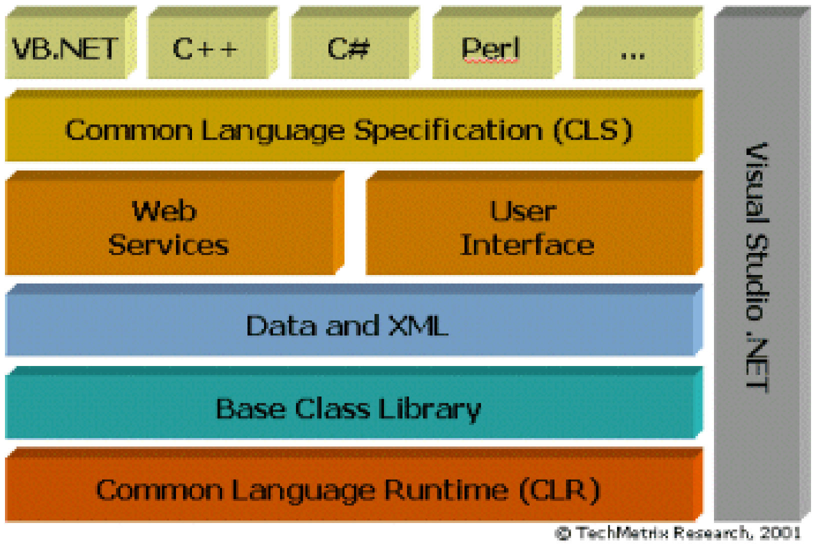
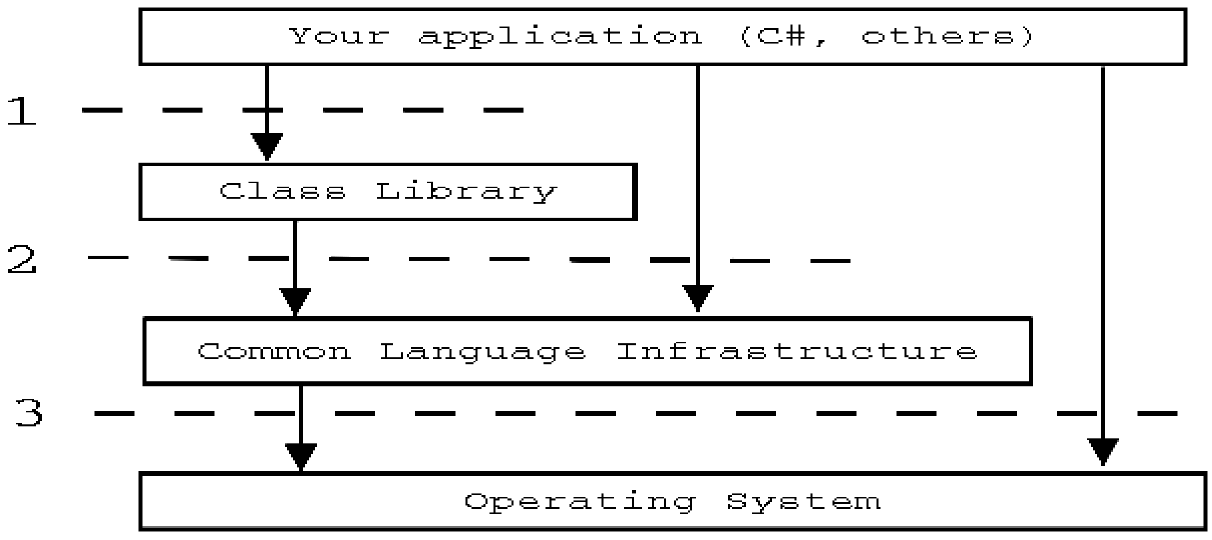


Figure 3: Visual studio .Net Contextual.

Meanwhile when approaching something as complex as .Net, it helps to have some idea of how it relates to the developing of a Web Service. The following diagram provides some meaning for where our ongoing developed application fits into .Net Platform.



```
Dashed lines denote interfaces:
1:   Library includes (e.g. "using System")
2:   Common Languages Specification
3:   Native Operating System API
```

NOTE: This is just a simplified view; we're ignoring ASP.NET and ADO.NET for now.

2.5.1. What is a Web Service again?

Simply put, a Web Service is an application that exposes a programmatic interface using standard, Internet-friendly protocols. Web Services are designed to be used by other programs or applications rather than by humans. Programs invoking a Web Service are called Clients and SOAP (**S**imple **A**ccess **O**bject **P**rotocol) over HTTP is the most commonly used protocol for invoking Web Services. Figure 4 shows how SOAP calls are remote function calls that invoke method executions on Web Service components at Location B. The output is always rendered as XML and passed back to the user at Location A.

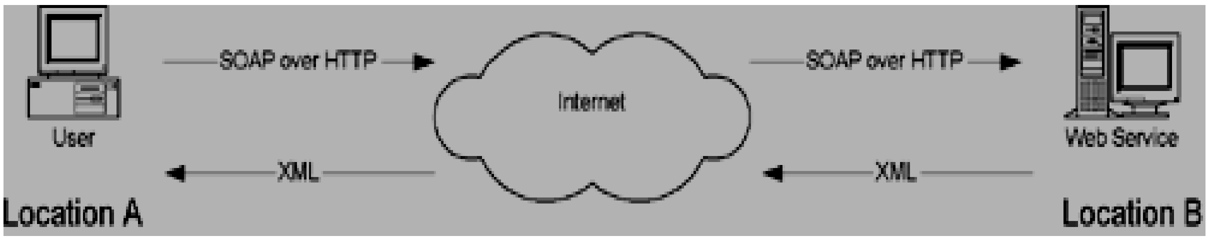


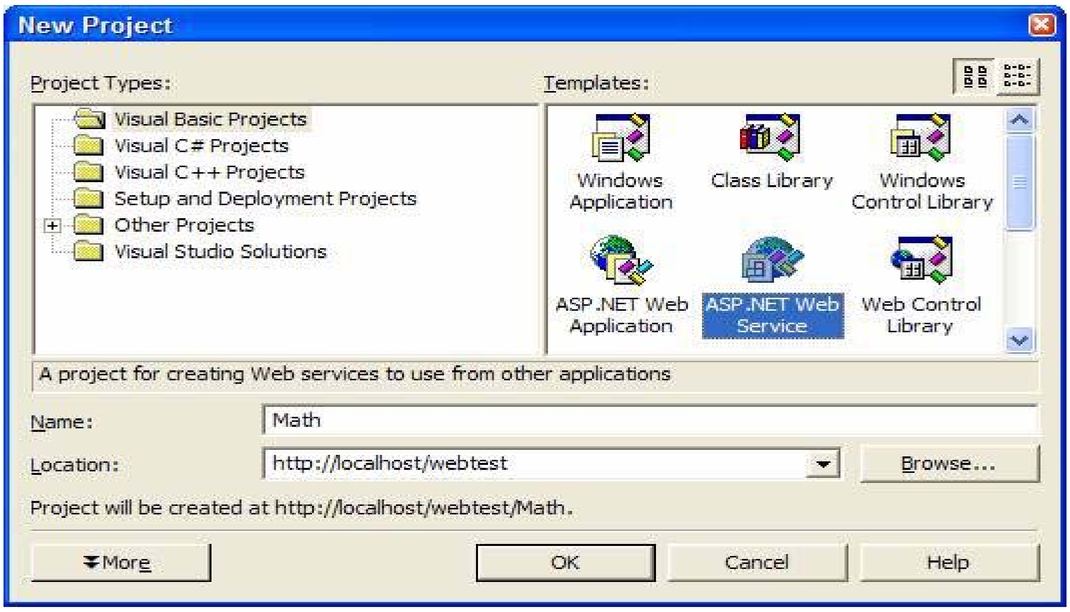
Figure 4: User – Web Service orchestration.

*P.S.: Thus, the developer or the programmer **creates** a Web service which **will be invoked** by the User or the Client if he or she ever needs that Service.*

2.5.2. Creating a Web Service with .Net (Steps and Screen shots)

All the necessary tools to create Web Services is built into the .NET Framework and the Visual Studio .Net which are themselves freeware downloadable from Microsoft download page www.microsoft.com/downloads or from the www.download.com page

The **easiest way** to build a Web Service with .NET is to use Visual Studio .NET and create an ASP.NET Web Service project as shown below.



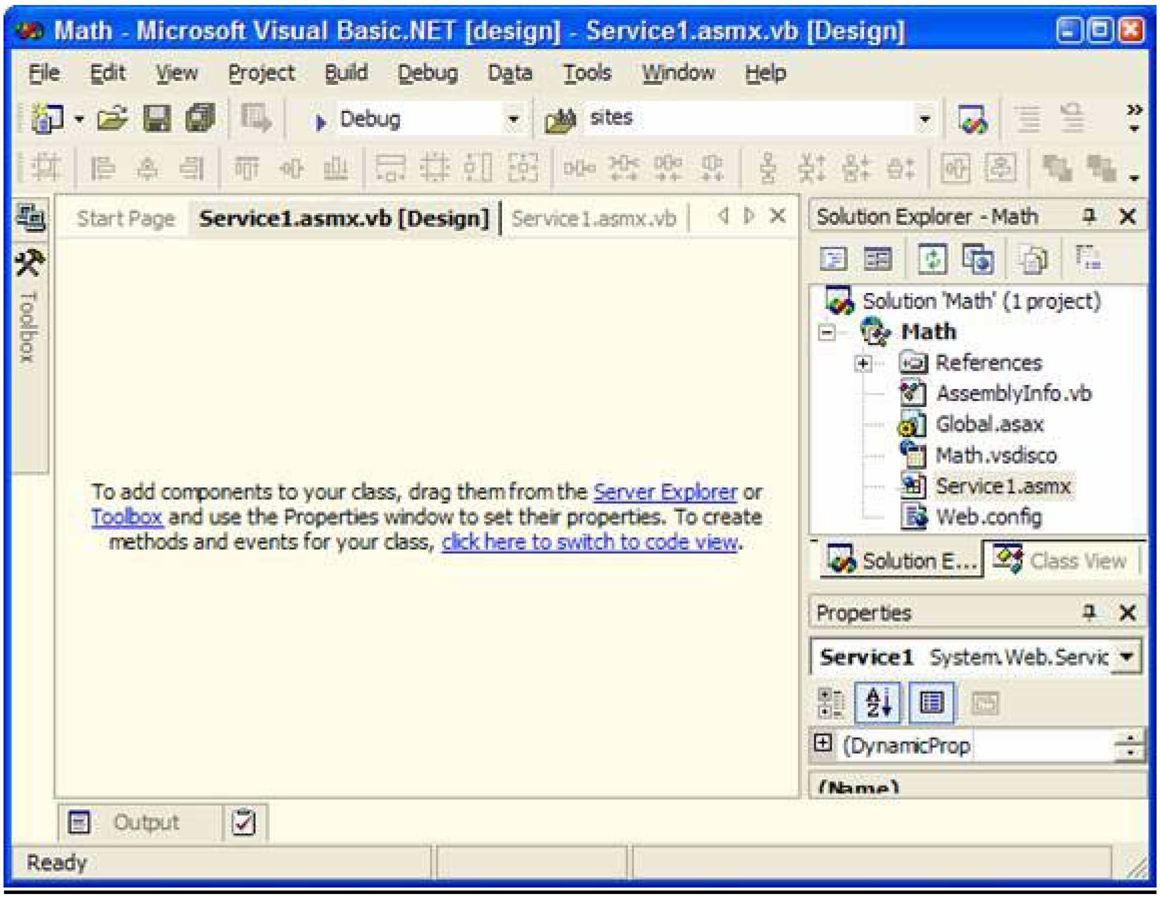
A project for creating Web services to use from other applications

Name: Math

Location: http://localhost/webtest

Project will be created at http://localhost/webtest/Math.

VS .NET will allow the necessary files including the Web service files which are called service1.asmx and service1.asmx.vb by default. The next screen we get is the Web service designer with the list of project files shown in the solution explorer on the right:


Math - Microsoft Visual Basic.NET [design] - Service1.asmx.vb [Design]
File Edit View Project Build Debug Data Tools Window Help
Debug
sites
Start Page
Service1.asmx.vb [Design]
Service1.asmx.vb
Toolbox
Solution Explorer - Math
Solution 'Math' (1 project)
Math
References
AssemblyInfo.vb
Global.asax
Math.vsdisco
Service1.asmx
Web.config
Solution E...
Class View
Properties
Service1 System.Web.Servic
(DynamicProp
(Name)
Output
Ready

To add components to your class, drag them from the Server Explorer or Toolbox and use the Properties window to set their properties. To create methods and events for your class, click here to switch to code view.

After having clicked on"click here to switch to code view", this'll open the code file called service1.asmx.vb where the Web Service code will reside. We will follow these next steps to build a simple Math Web Service:

1. ***We rename the class called Service1 to MathService.***

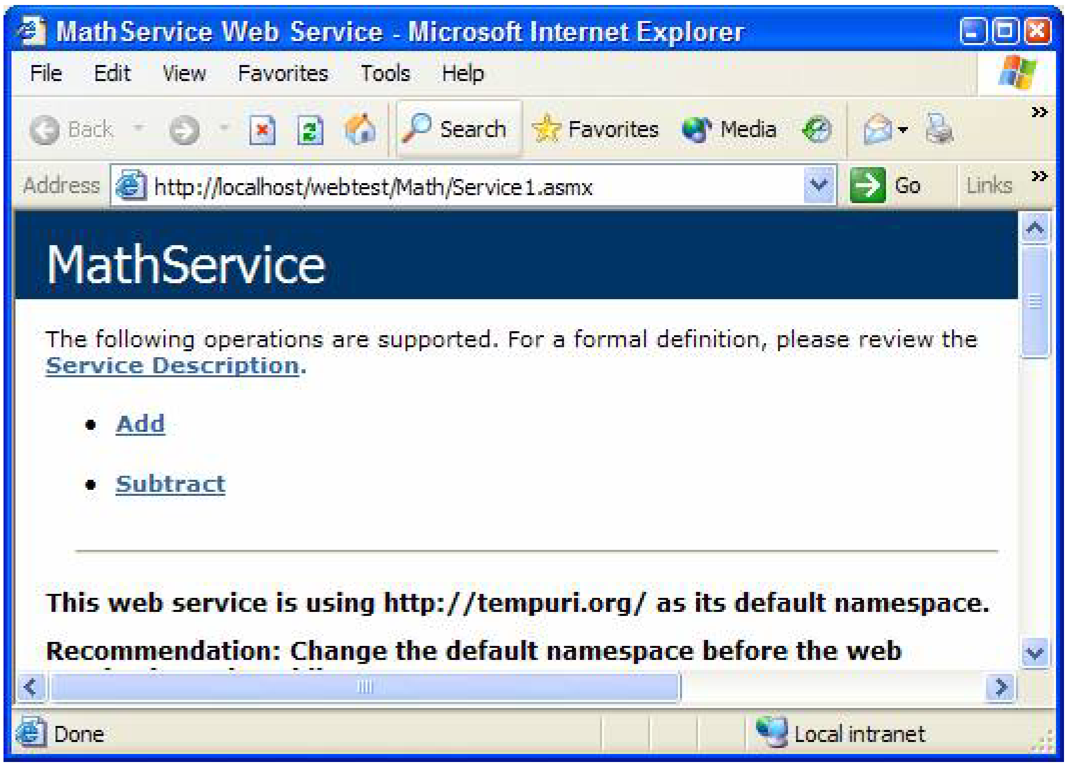
2. ***We create two functions named Add and Subtract. Note that each function must be public and must have the <WebMethod ()> attribute added to it. Here's the code for the entire class including the two functions:***

```
Imports System.Web.Services    // We import the Web service Class Library

Public Class MathService Inherits System.Web.Services.WebService

// We define the code of our Web Service
```

```
        'Add function
        <WebMethod ()> Public Function Add (_
                ByVal a As Double, _
                ByVal b As Double) As Double
            Return a + b
        End Function
        'Subtract function
        <WebMethod ()> Public Function Subtract (_
                    ByVal a As Double, _
                    ByVal b As Double) As Double
            Return a - b
        End Function
    End Class
```

3. *We build the project by going to the Build menu and selecting Build.*

4. *We click the play button on the toolbar and wait for Internet Explorer to be launched. We should see a page that looks like this:*


MathService Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/webtest/Math/Service1.asmx Go Links
Done
Local intranet

MathService

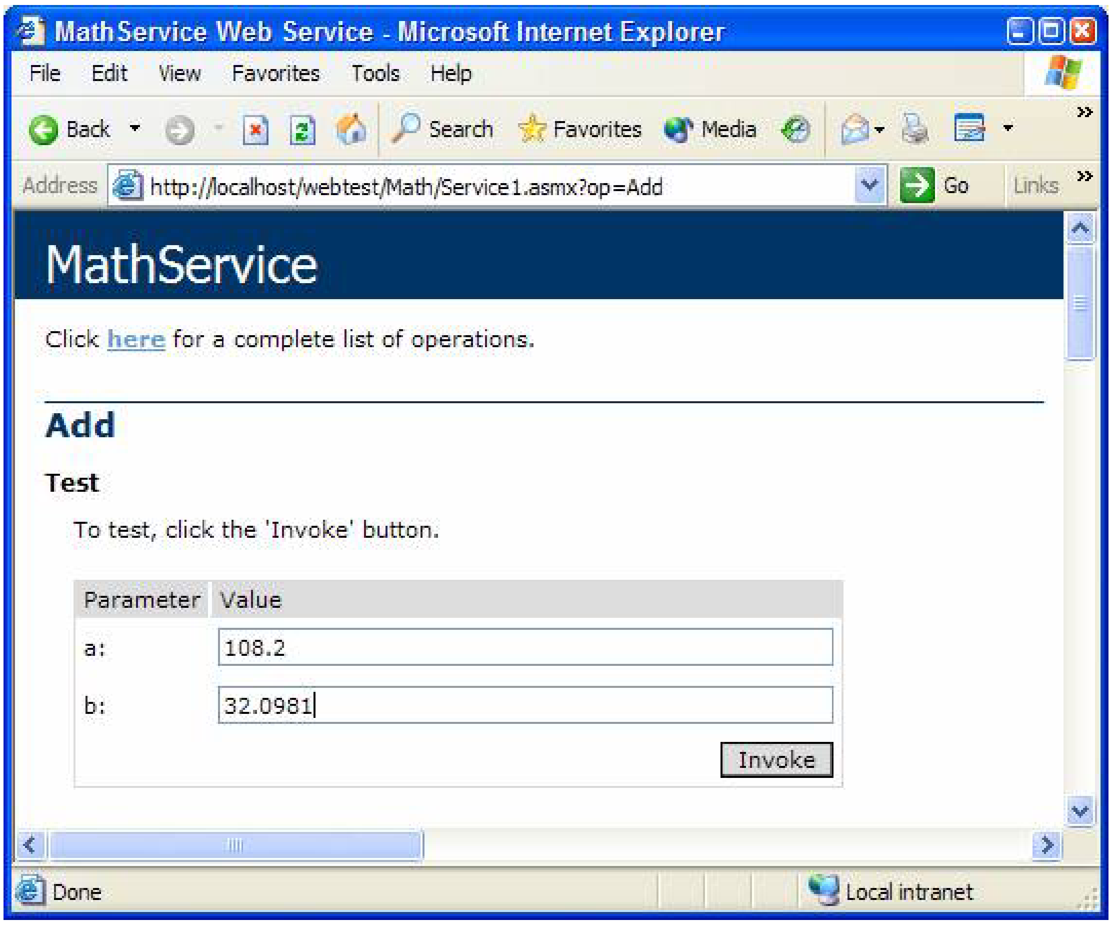
The following operations are supported. For a formal definition, please review the Service Description.

- Add
- Subtract

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the web

5. *This page is automatically generated by the .NET when a browser navigates to the Web service URL. In the above example, our Web service's URL is http://localhost/webtest/Math/Service1.asmx. Clicking on the Add link in the above page, gives us a page where we can supply the input parameters and invoke the Add method like this:*


MathService Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/webtest/Math/Service1.asmx?op=Add
Go Links

MathService

Click here for a complete list of operations.

Add

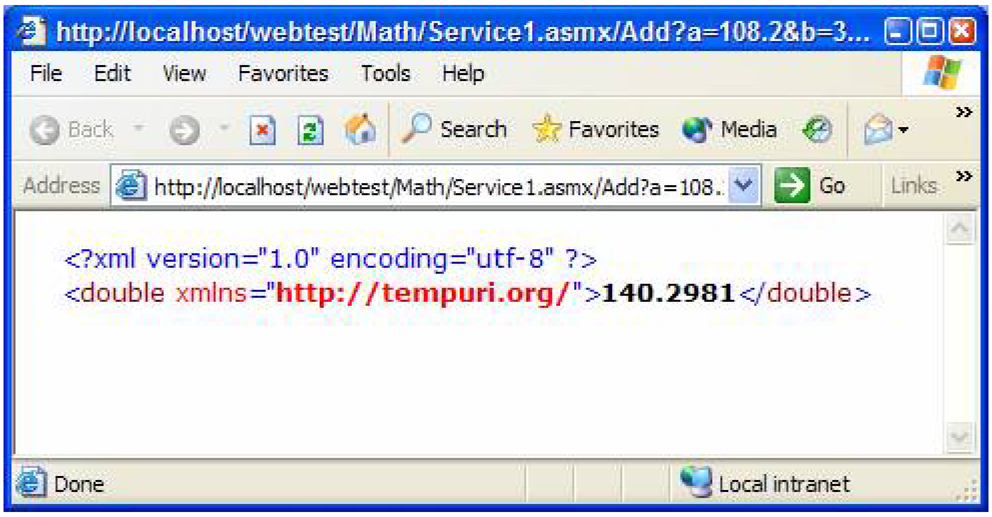
Test

To test, click the 'Invoke' button.

Parameter	Value
a:	108.2
b:	32.0981

Invoke
Done
Local intranet

6. *Let us enter two numbers in and click Invoke. This will invoke the Web Service and give us back the result in this format:*


http://localhost/webtest/Math/Service1.asmx/Add?a=108.2&b=3...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/webtest/Math/Service1.asmx/Add?a=108.
Go Links

```
<?xml version="1.0" encoding="utf-8" ?>
<double xmlns="http://tempuri.org/">140.2981</double>
```

We just built and invoked a Web Service with Microsoft .Net by using the Visual Studio .Net of the .Net Platform.

3. BizTalk

Today's Business to Business (B2B*) requires to companies from different industries to communicate, although those companies often use dramatically different packages to develop or to expose their businesses. Microsoft has tried and is still trying to incorporate itself in that problem's solution by offering a new product that tries to reduce the process cycle on B2B by:
- Making better decisions faster.
- Leveraging dynamic business relationships.
- Quickly responding to customer demands.

Those are basically the new economical business imperatives, and the product that Microsoft is now offering is called BizTalk, which certainly refers to the say "Let's talk Biz"

3.1. What is BizTalk?

BizTalk is essentially a **Design Framework for implementing XML schemas, and a Set of XML elements used to pass messages between applications which describe business forms or business transactions**. Microsoft ceases the opportunity here to support B2B services, InterCompany and E-Commerce by putting in trend his initiative this way with the following three components that describe what BizTalk is in general:

- **The BizTalk Community** (Biztalk.org from Microsoft) which exposes:
  - A Library for business document specification.
  - A Stimulation for reuse of business documents.
  - It submits and retrieves document specifications.
  - A Third-party XML business documents for registered organizations
- **The BizTalk Framework** which helps for defining:
  - XML Specification for document routing and exchange (SOAP 1.1 compliant).
- **The BizTalk Server** which exposes:
  - A Microsoft Server product for processing, transforming, signing, routing of documents in business.

3.2. Benefits of BizTalk

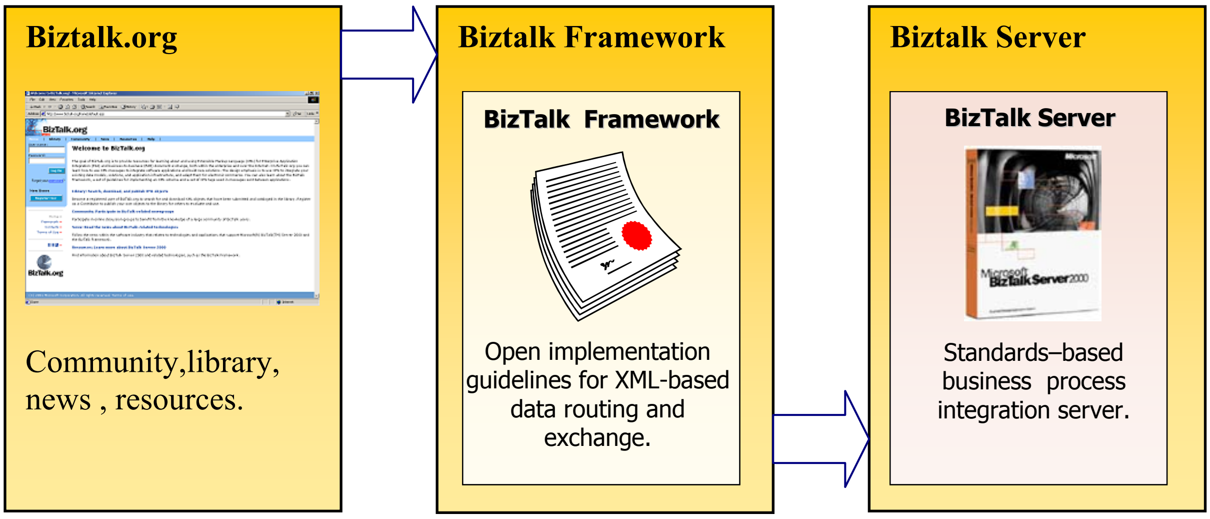
The goals of the product BizTalk are to improve the Business challenges that Enterprises face in their **E**nterprise **A**pplication **I**ntegration (EAI)*, in their **E**nterprise **R**esource **P**lanning (ERP)* and especially to support the **B**usiness **to B**usiness (B2B) exchange.
The benefits that BizTalk exposes are then:
Integrating existing systems. Rapidly build and deploy secure, reliable business processes within and between organizations. The tools and services in Microsoft BizTalk Server help businesses quickly integrate applications regardless of operating system, programming model, or programming language, while minimizing the amount of custom coding and maintenance needed.
Automating business processes. BizTalk makes it fundamentally easier to automate business processes. The accelerator and BizTalk Server have been designed and tested to handle the throughput, security, and reliability requirements of the largest organizations.
Reduce Claims Processing Cycle Times
Giving businesses control over their claims process. BizTalk enables organizations to define and monitor meaningful metrics that allow them insight into the operational health of their businesses. Plus, powerful business process automation offers more control over the claims process.

Shortening accounts receivable cycles. BizTalk includes document tracking and analysis functionality that enable organizations to observe and analyze their data flow and business processes in real time. They can use the Claims Processing Orchestration Schedule to view requests for and subsequent payments on claims, and to sort claims according to payment status
Reducing of administrative costs for enterprises: BizTalk can help your organization reduce the cost compliance if the organization properly employs those tasks to develop their businesses.

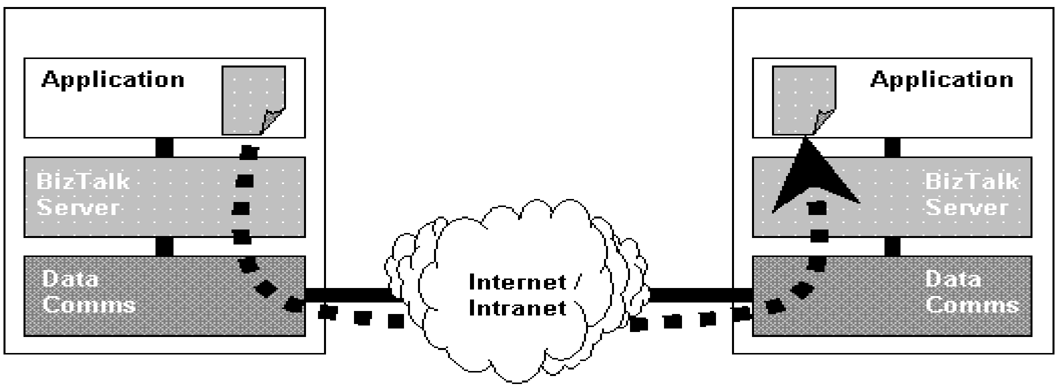
In one sentence we can simply say that the general benefits of BizTalk through the Microsoft initiative include: a Roadmap for better E-Business, a schema's repository for exchanging datas between Businesses, and a mapping across business schemas and transaction's forms.

3.3. BizTalk Sever : Architecture and Tools

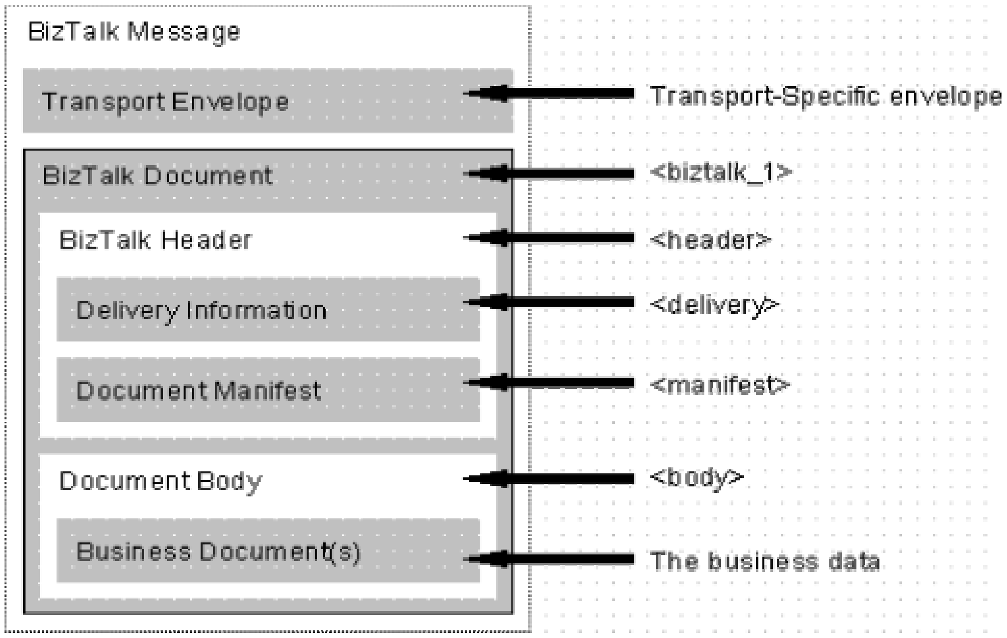
We have already evoked the three basics components that Microsoft offers to support the BizTalk product, known as the three below:


Biztalk.org
Community,library, news , resources.
Biztalk Framework
BizTalk Framework
Open implementation guidelines for XML-based data routing and exchange.
Biztalk Server
BizTalk Server
Standards–based business process integration server.

The **Biztalk.org** is simply defined as a Community. The logical application model for the **BizTalk Framework** is implemented in layers. These logical layers include **the Application** (and appropriate adapters), the **BizTalk Server**, and **Data Communications** as shown below.



Meanwhile, the BizTalk Server is the one that businesses use for **processing, routing and signing business documents** which are the XML streams containing the business transaction datas which refer to the inside of the **BizTalk Message**.
BizTalk Messages typically contain one BizTalk Document along with some additional information used by a BizTalk Server or application to handle and process the document. A BizTalk Message is constructed as shown in the diagram below.



So for processing, signing, transforming and editing those messages, the BizTalk Server provides a set of many graphical tools that businesses use to manage and track and analyse their documents. Those tools actually are designed to integrate applications and business systems and ensure the designed functionality.

BizTalk Server includes the following graphical tools:

BizTalk Orchestration Designer enables users to visually define and build robust, distributed business processes

BizTalk Editor enables users to define schema for XML, EDI*, and flat files.

BizTalk Mapper enables users to link disparate schemas and define steps for successful document transformations.

BizTalk Messaging Manager is a wizard-based tool that enables users to manage the details of business-process transactions.

BizTalk Server Administration enables administrators to perform common administrative tasks such as adding and removing servers.

BizTalk Document Tracking enables users to track documents as they move through various stages of the business process.

Among all of the graphical tools the most common used tools while manufacturing Business documents are:
The BizTalk Orchestration Designer for Defining the Workflow.
The BizTalk Editor for Transforming the documents.
The BizTalk Mapper mainly for Linking the documents.

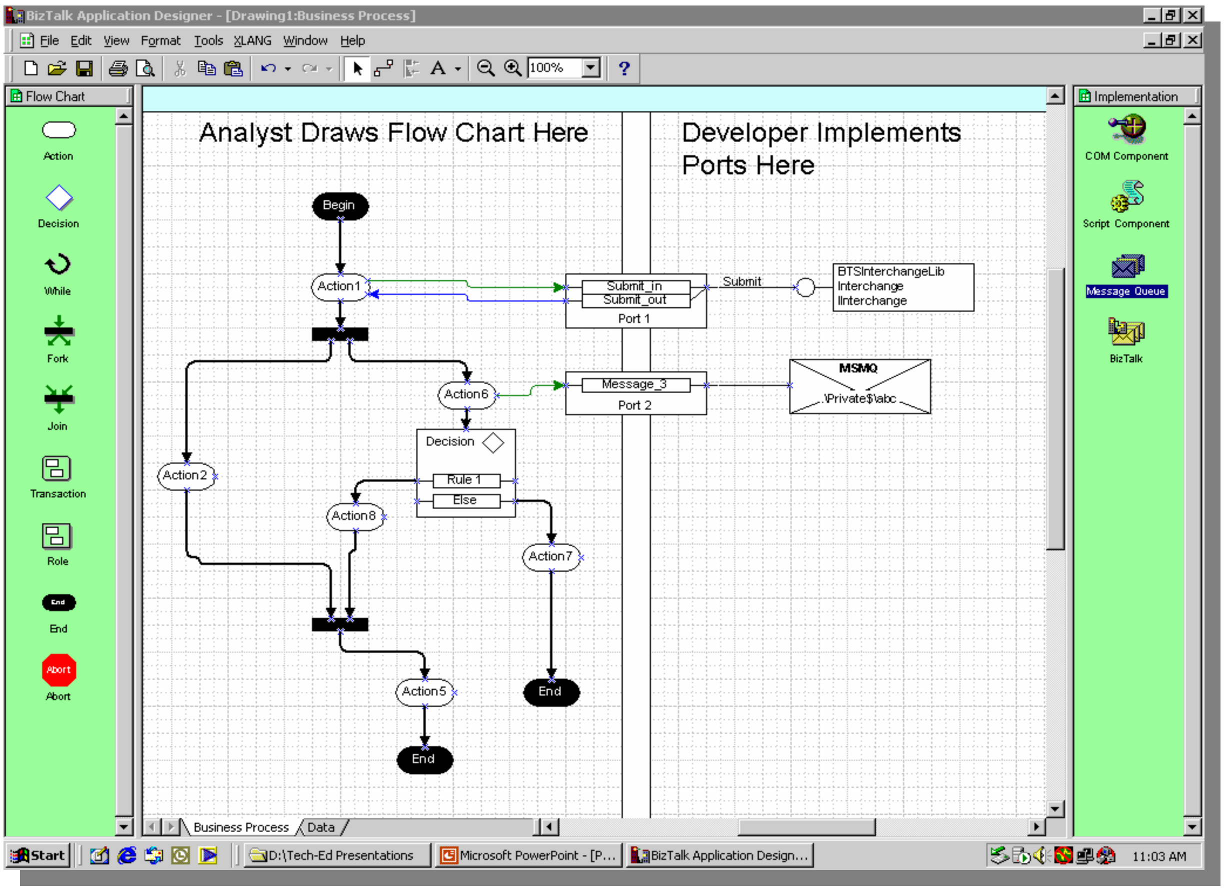
3.3.1 The BizTalk Orchestration Designer

BizTalk Orchestration is a technology for creating and orchestrating business processes. That orchestration is based on a application of XML called **XLANG.** The processes that are developed with the BizTalk Orchestration are all based on Workflow rules. BizTalk Orchestration goals are:
- The Separation of definition from implementation.
- The Dynamic processes.
- The "Any to Any" Integration as well.

3.3.1.1 Example of a BizTalk Orchestration

The business processes modeling while using BizTalk Orchestration are:
- *Biztalk Orchestration Designing.*
- *Generating XLANG diagrams from that design.*
- *Defining dataflow for the designed informations.*
- *Translating XLANG diagrams into XLANG schedules.*

The figure below shows a simple BizTalk Orchestration window.



3.3.1.2 Microsoft Orchestration Language: XLANG

From the above design, the BizTalk Orchestration tools will generate a **XLANG** document in order to guarantee an asynchronous communication, a persistent model and stateful applications;
On the Orchestration, what is important to identify for the **XLANG** document is the **Action** Element, the **Message** Element, the **Port** Element, and the **Sequence** Element as shown on the below diagram.

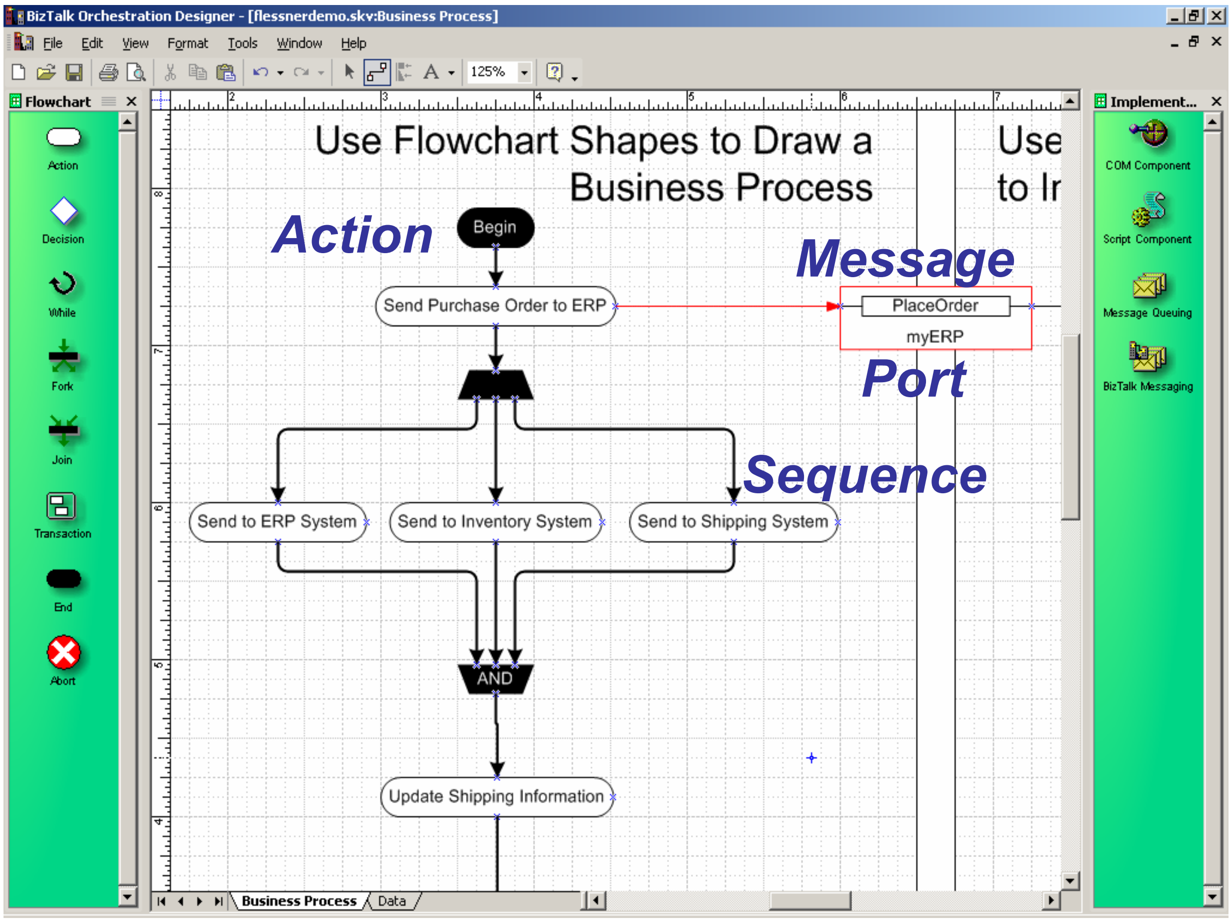
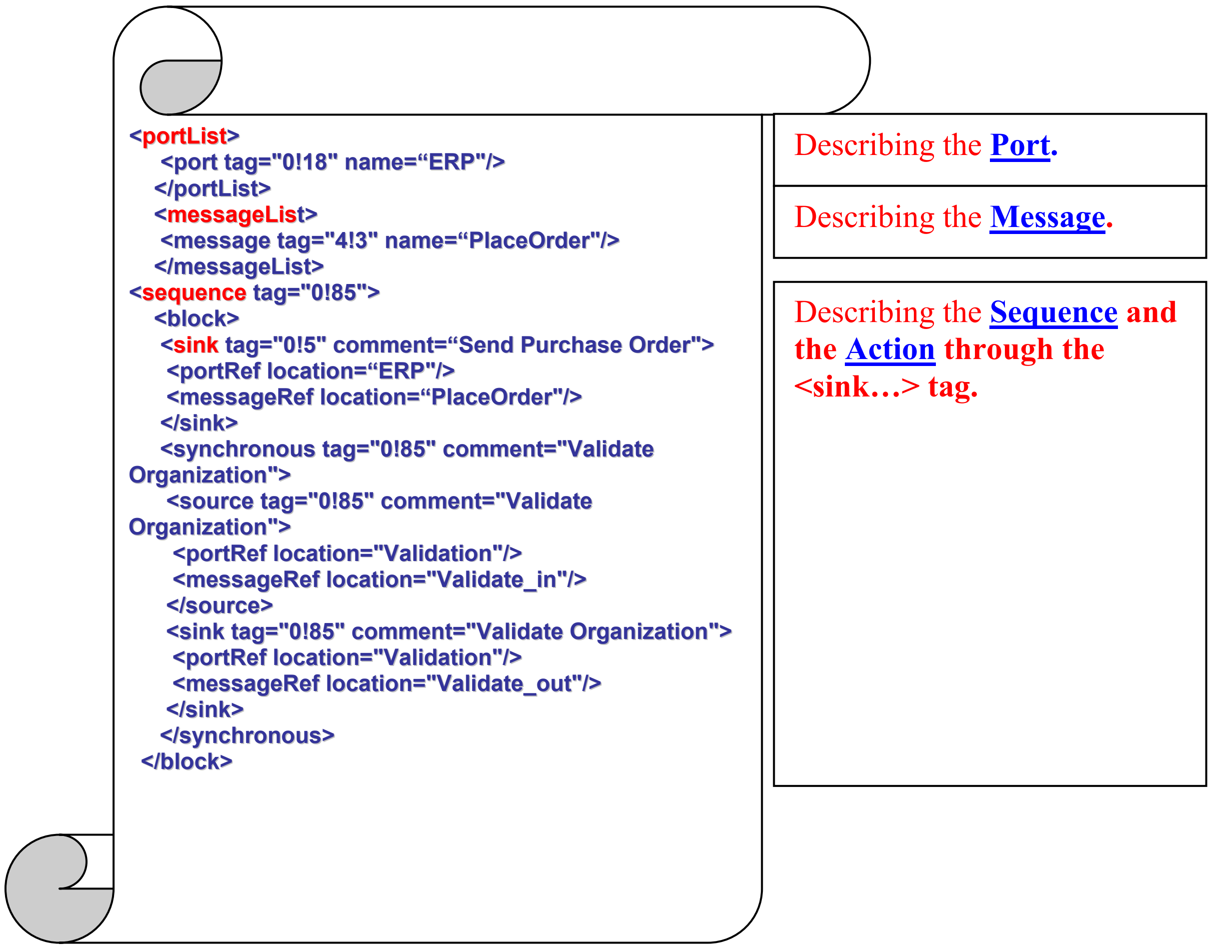


Figure5: XLANG's Elements Identification.

Thus, XLANG (pronounced SLANG) is the language with which one visually builds, using the Visio-style environment offered by the BizTalk Orchestration Designer. By dragging and dropping components, you are visually building instructions to execute the orchestration, which is why Microsoft calls it Microsoft Orchestration.

The Corresponding XLANG document will present the elements known as:
The Port for identifying the internet transport of the document,
The Message to specify the inside of the document,
The Sequence and the **Action** to describe the transmitting and the processing of the document.

The XLANG document for the taken example above will quite look like this:



```
<portList>
    <port tag="0!18" name="ERP"/>
   </portList>
   <messageList>
    <message tag="4!3" name="PlaceOrder"/>
   </messageList>
<sequence tag="0!85">
   <block>
    <sink tag="0!5" comment="Send Purchase Order">
     <portRef location="ERP"/>
     <messageRef location="PlaceOrder"/>
    </sink>
    <synchronous tag="0!85" comment="Validate
Organization">
     <source tag="0!85" comment="Validate
Organization">
      <portRef location="Validation"/>
      <messageRef location="Validate_in"/>
     </source>
     <sink tag="0!85" comment="Validate Organization">
      <portRef location="Validation"/>
      <messageRef location="Validate_out"/>
     </sink>
    </synchronous>
 </block>
```

Describing the Port.
Describing the Message.
Describing the Sequence and the Action through the <sink…> tag.

XLANG is a complete set of semantics for describing business process interactions; the communication of messages is done through ports and the tasks of the sequences can be parallel or switch as long as the transactions made are atomic and long-running.
The **XLANG**'s structure is based on WSDL*. That means the XLANG's **Modules** map to WSDL <definitions> , the **Services** map to WSDL <service> and the **Ports and PortTypes** map to WSDL <port> and <portType>.

Why use XLANG to describe the workflow of document's transaction?

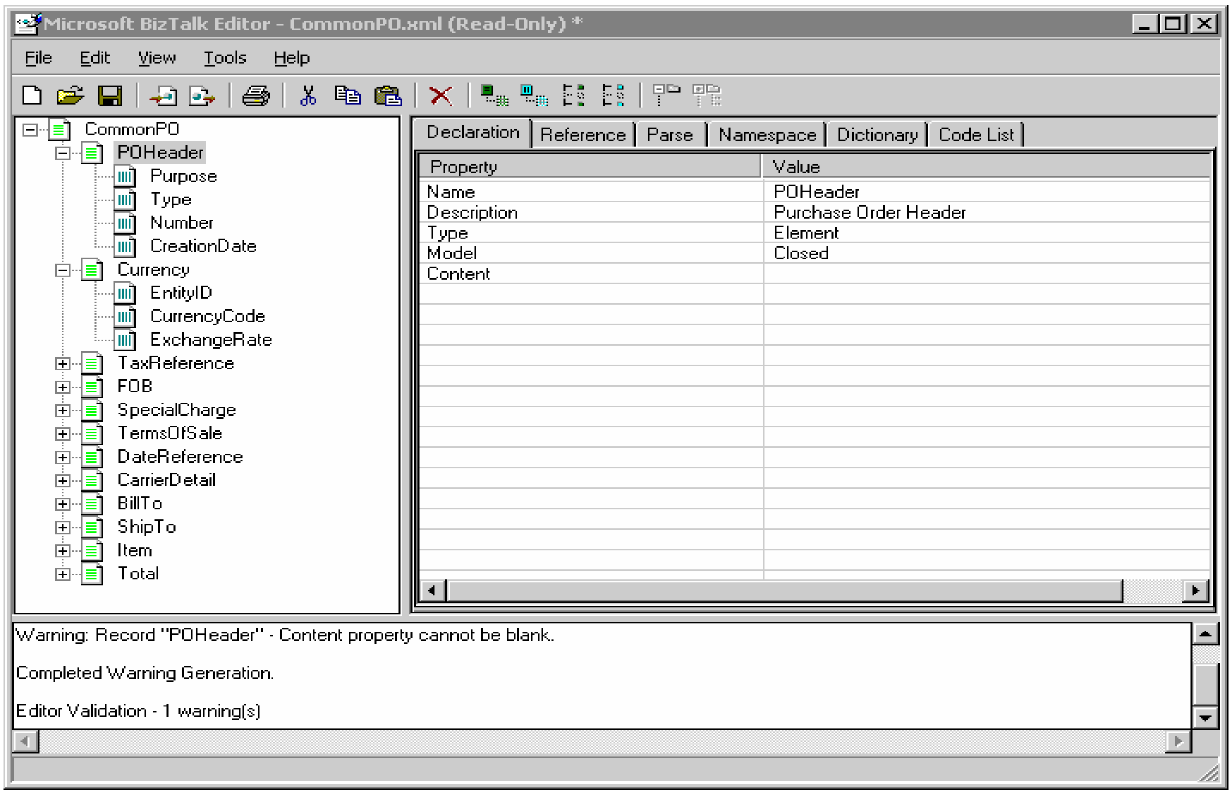
XLANG addresses common feedback to the facts that:

- It defines a programmer-friendly syntax as XML for Web Services.
- The compilation fits directly to the .NET Assembly, which means it supports C#, Visual basic….
- The integration goes with Web Services as far as it functions like XML.

The next generation of XLANG will be effective for describing BizTalk's process Orchestration, but now the XLANG language is still very young and almost unused and there is still a bunch to do to perform his functionalities.

3.3.2. The BizTalk Messaging Editor

It defines business document structures and generates the schema syntax for those documents incorporating the XSLT* parameters like templates and the XML modules like the XML's DTDs (XML **D**ocument **T**ype **D**efinition).
There is a simple Framework that helps for that without having to worry on about writing a bunch of XML tags. That framework is called the BizTalk Editor designer. On that Framework, one only needs to input the parameters of the business document. The figure below describes the example of a document in the BizTalk Editor.



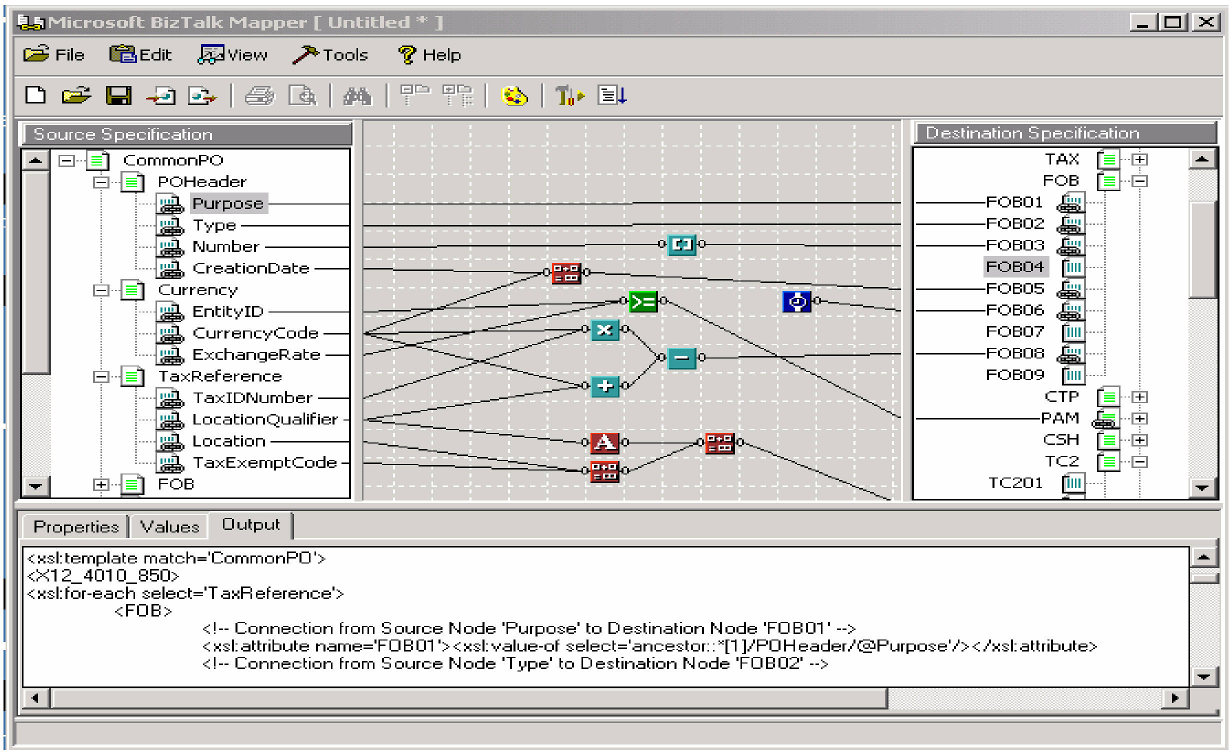
Property	Value
Name	POHeader
Description	Purchase Order Header
Type	Element
Model	Closed
Content	

```
Warning: Record "POHeader" - Content property cannot be blank.

Completed Warning Generation.

Editor Validation - 1 warning(s)
```

Figure6: A BizTalk Editor Design.

3.3.3. The BizTalk Messaging Mapper

For linking the business documents, the Biztalk Mapper generates XSLT Components in order to create maps formats which play the role of linking the files while the transaction is made. The mapping of the elements is done on a Mapper Framework to simplify the duty of having to write a code for mapping the selected objects. The figure below shows an example of a Biztalk Mapper window. We see on the below side of the Mapper Framework that for every mapping made, the Framework generates automatically XSLT tags which are helpful for linking XML objects.

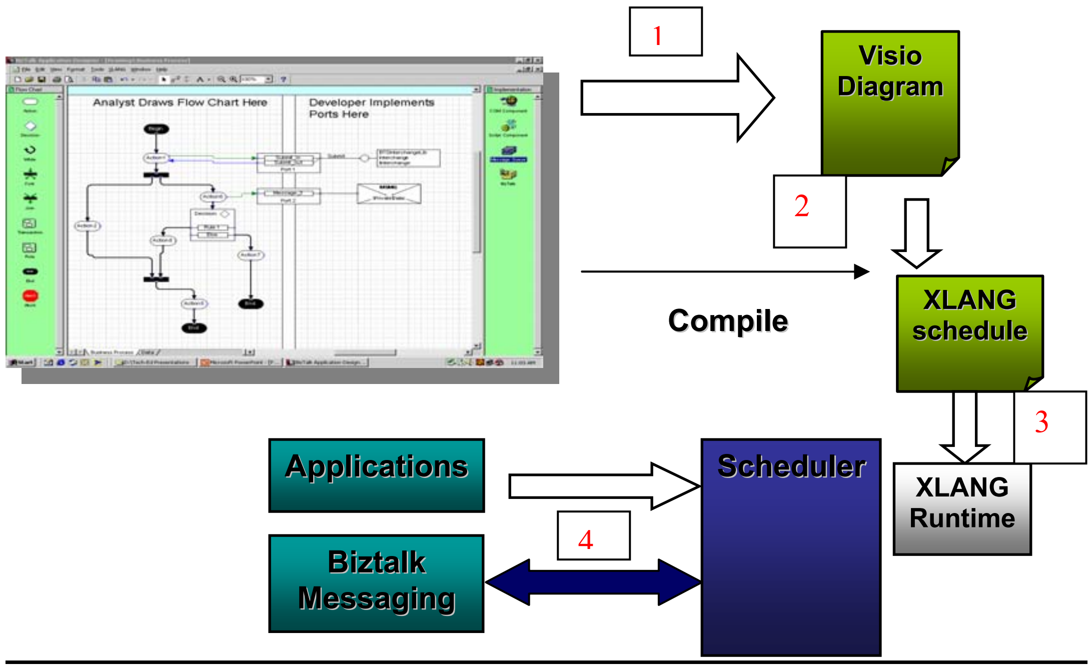


```
<xsl:template match='CommonPO'>
<X12_4010_850>
<xsl:for-each select='TaxReference'>
            <FOB>
                        <!-- Connection from Source Node 'Purpose' to Destination Node 'FOB01' -->
                        <xsl:attribute name='FOB01'><xsl:value-of select='ancestor::*[1]/POHeader/@Purpose'/></xsl:attribute>
                        <!-- Connection from Source Node 'Type' to Destination Node 'FOB02' -->
```

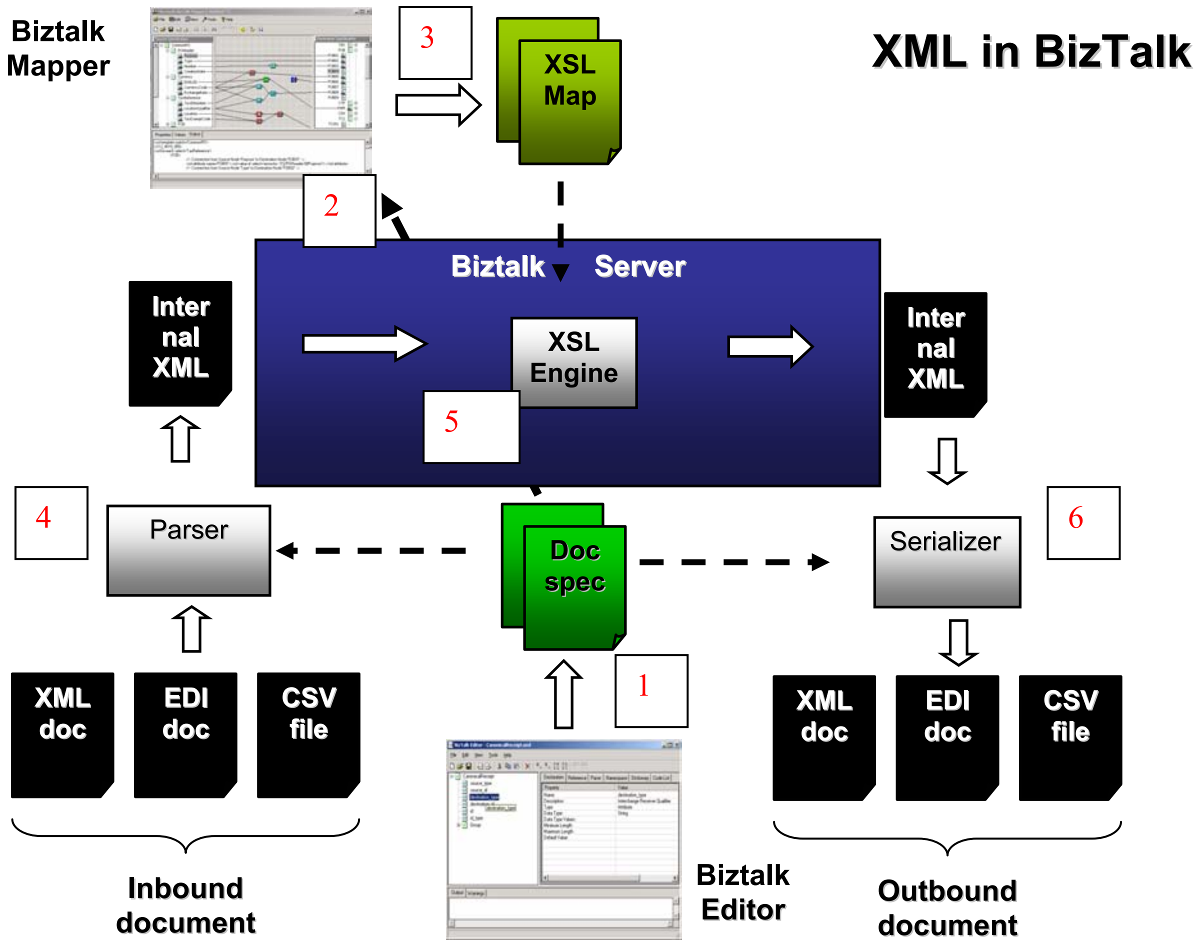
Figure7: A BizTalk Mapper Window.

3.4. How does XML function with the BizTalk technology?

The XML Language is the language of Web Service, which is why it is used for describing all in- and outbound business documents as an extended data schema and as an internal format for all processes. The translations of those documents are done with the help of **XSL**, and the new Microsoft language **XLANG** will support the workflow orchestration schedules following this diagram below.



Viewed the concept of XML in BizTalk, the typical scenario while working with all the main BizTalk features (Orchestration, Editor, and Mapper), is described following the order of the 6 steps of the diagram down here.



To sum up about business transactions, we can state that addressing, sending and receiving Biztalk Messages is not enough for ensuring complete integration solutions. Business processes require sequence of actions and a suitable message flow. So the business logic, which manages business processes, needs to be separated from implementation solutions and needs to be defined at a higher level which is why BizTalk Server is an integration server that offers us tools and services for integration on data level and business process view.

4. Summary (General Conclusion)

As far as we have been able to understand all of this, we come up with the meaning of Microsoft.Net as a XML Technology Platform to develop and manage Web Services, and facilitate the common business transactions nowadays with his BizTalk Sever features.

Microsft.Net is allowing the creation distributed Web Services that will integrate and collaborate with a range of complementary services to serve customers in ways that yesterday one could only dream of. And with the support of BizTalk as a set of guidelines for using XML to create schemas which define commonly used business forms (i.e. common business transactions), .Net is able to drive the Internet concept where the information is available at any time, any place and on any device.

If I allow myself to say a word about the whole .Net and BizTalk world, I will say that the features offered here by Microsoft let us to conclude that Microsoft provides a common solution for developing Web Services and undertakes the duty to make Electronic Business better than it was before. It is quite though tough to compare Microsoft with the others Web Services Platform Vendors(Sun, IBM, BEA..) as long as it has gone awhile ago out of the Web Services Consortium maybe because their offered products are more compatible to the user's expectations, which perhaps makes it the actual best Solution Provider for Web Services.

5. Glossary (in Alphabetical Order)

Here are the different meaning of all the expressions marked with axterix (*) for a better understanding.

ADO.NET provides consistent access to data sources such as Microsoft SQL Server, as well as data sources exposed through XML. Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data. ADO.NET cleanly factors data access from data manipulation into discrete components that can be used separately or in tandem. ADO.NET includes .NET Framework data providers for connecting to a database, executing commands, and retrieving results. Those results are either processed directly, or placed in an ADO.NET **DataSet** object in order to be exposed to the user in an ad-hoc manner, combined with data from multiple sources, or remoted between tiers. The ADO.NET **DataSet** object can also be used independently of a .NET Framework data provider to manage data local to the application or sourced from XML.

ASP.NET is Microsoft's new tool for web developers to rapidly create web applications. ASP.NET represents the latest technology to enable server-side scripting of not only Web sites, but Web services as well. As with any new technology, it is critical that developers are provided with the tools and resources that help them to get a leg up on how to properly use and integrate these technologies into the solutions that they are developing. The ASP.NET team has done some groundbreaking work at building an exciting tool to aid ASP.NET developers. This tool is "ASP.NET Web Matrix" and while it is both free to download, and just barely over a megabyte in size, it also is an amazingly full featured tool that truly makes ASP.NET development easy and fun.

Business-To-Business (B2B). A transaction_that occurs between a company and another company, as opposed to a transaction involving a consumer. The term may also describe a company that provides goods or services for another company. The most important point here is that those companies may not always be from the same category, let's say An Architect Company to a Medicine Company may uses the B2B principles to communicate.

EAI (Enterprise **A**pplication **I**ntegration**)** is a business computing term for the plans, methods, and tools aimed at modernizing, consolidating, and coordinating the computer applications in an enterprise. Typically, an enterprise has existing legacy applications and databases and wants to continue to use them while adding or migrating to a new set of applications that exploit the Internet, e-commerce, extranet, and other new technologies. EAI may involve developing a new total view of an enterprise's business and its applications, seeing how existing applications fit into the new view, and then devising ways to efficiently reuse what already exists while adding new applications and data.

EDI (**E**lectronic **D**ata **I**nterchange) is a Pre-Internet form of E-commerce. Governed by rules maintained by the United Nations, EDI set out to formalize electronic commerce as a set of standardized message formats, by which businesses would exchange documents such as purchase orders and invoices with each other electronically. Until the advent of the Internet, EDI messages had to be exchanged via dedicated Value-Added Networks (VANs). This restricted its use to large corporations who had the clout to impose EDI standardization on their supply chain partners. It is gradually being absorbed into (or, depending on your point of view, superceded by) XML-based equivalents.

ERP (**E**nterprise **R**esource **P**lanning) is the industry term used to describe a broad set of activities supported by multi-module application software that helps a manufacturer or other business manage the important parts of its business. These parts can include product planning, parts purchasing, maintaining inventories, interacting with suppliers, providing customer service, and tracking orders. ERP can also include application modules for the finance and human resources aspects of a business. Some of the bigger players in the ERP outsourcing market are SAP, Peoplesoft, and J. D. Edwards. New comers include Oracle, IBM, and Microsoft.

Peer-to-Peer (P2P):
Is a communication's model in which each party has the same capabilities and either party can initiate a communication session. Other models with which it might be contrasted include the Client/Server and the Master/Slave. In some cases, Peer-to-Peer communication is implemented by giving each communication node both server and client capabilities. In recent usage, peer-to-peer has come to describe applications in which users can use the Internet to exchange files with each other directly or through a mediating server. For example the Napster was a kind of P2P Software.

Service-Centric:
Is just the opposite of P2P, in service-centric model all the communications run through a Server, that means if an application wants to send data to another application, it would have to first contact a central server, which takes that data, reads the address of the recipient and passes it along.

Smart devices:
.NET uses software for smart devices to enable PCs, laptops, workstations, smart phones, handheld computers, Tablet PCs, game consoles, and other smart devices to operate in the .NET universe. A smart device is:
Smart about you—uses your .NET identity, profile, and data to simplify your experience and is smart about your presence, allowing tailoring of notifications in response to your presence or absence.
Smart about the network—responsive to bandwidth constraints; provides support for both online and offline use of applications; and understands which services are available.
Smart about information—allows you to access, analyze, and act on data anywhere anytime.
Smart about other devices—discovers and announces PCs, smart devices, servers, and the Internet; knows how to provide services to other devices; smart about accessing information from the PC.
Smart about software and services—presents applications and data optimally for form factor; enables input methods and connectivity appropriate for great end-user interaction; consumes Web services using XML, SOAP, and UDDI; and programmable and extensible by developers.
Microsoft is currently working on the following software products for smart devices: Windows XP, Windows Me, Windows CE, Windows Embedded, the .NET Framework, and the .NET Compact Framework.

WSDL (Web Services Description Language) is the standard format for describing a web service. A WSDL definition describes how to access a web service and what operations it will perform. Usually pronounced 'whizz-dul' (to rhyme with 'whistle'), WSDL is seen (with SOAP and UDDI) as one of the three foundation standards of web services.
WSDL Definition elements are:
```
<wsdl:definitions>
<wsdl:types></wsdl:types>
<wsdl:message></wsdl:message>
<wsdl:portType></wsdl:portType>
<wsdl:binding> <wsdl:operation></wsdl:operation> </wsdl:binding>
<wsdl:service></wsdl:service>
</wsdl:definitions>
```

XSLT is a tool for transforming an XML (eXtended Markup Language) document into either an HTML document, or a document using different XML tag set.

6. Links and papers used for this work

Here are the whole basic general links used for the research of the documentation's papers:

http://www.microsoft.com/net/default.asp
http://msdn.microsoft.com
http://www.microsoft.com/biztalk/
http://www.w3c.org (All mentioned standards)
http://msdn.microsoft.com/xml (Tutorials and reference on xml)
http://msdn.microsoft.com/soap(Tutorials and reference on xml)
http://www.biztalk.org(BizTalk organization)

Here are the specific papers used for this work.

-Understanding XML Web Services the Web Services Idea by (***Tim Ewald* Microsoft Corporation, July 12, 2002 Updated September 27, 2002)**
-Defining the Basic Elements of.NET (January 2003) (http://www.microsoft.com/net/basics/whatis.asp)
-What .NET Means for IT Professionals (July 2002). (http://www.microsoft.com/net/business/it_pros.asp)
-Four Ways .NET Can Improve Your Business (October 2002) (http://www.microsoft.com/net/business/ways.asp)
-Product Overview for .NET Framework. (October 2002) (http://msdn.microsoft.com/netframework/productinfo/overview/default.aspx)
-Overview of the .Net Framework by Microsoft.(November 2001) (http://msdn.microsoft.com/library/default.asp?url=/library/)
- Creating a .NET Web Services **by Chris Peiris. (January 2002)** (http://www.15seconds.com/Issue/010430.htm)
- TopXML Description and Transactions (http://www.vbxml.com)
- BizTalk organization (http://www.biztalk.org)
- Defining BizTalk. (http://www.microsoft.com/biztalk/)
- Product Overview about BizTalk Server and BizTalk services. (http://www.microsoft.com/biztalk/evaluation/financial/services/default/asp)
- XLANG Web services for business Process Design. (www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm)

.