3-tier OSS architecture with .NET platform

Seong Ik Hong, Young Il Kim, Woo Sung Kim
OSS Lab. KT
Taejeon, Korea
Email: {yeolin,yikim,kwsun}@kt.co.kr

APNOMS 2003

Abstract

As network technologies have tremendously evolved and the customer demand for telecommunications services has increased, the need for OSS to process various service orders and manage a lot of equipments has become bigger and bigger. So, today's service providers need to provide multiple services over a common infrastructure in a cost-competitive environment. Microsoft .NET is the Microsoft application development solution, which may significantly change how people interact with applications and devices. Lots of research centers forecast that .NET and Java platform will dominate e-business application development initiatives. In this paper, we will design a message oriented 3-tier OSS architecture and implement a simplified ADSL provision process using .NET platform and enterprise .NET servers to prove the possibilities of .NET based OSS. We will also discuss how much .NET platform satisfy the requirements for OSS platforms.

Introduction

- OSS general requirements[1]
  - Time-to-market, Extensibility, Cool user interface including web-based UI
  - Large scale transaction processing/Distributed transaction processing
  - Interoperability with other OSS, Low cost
- Recent popular OSS Platforms
  - CORBA [2],[3]
  - Java [1],[4]
  - Microsoft® .NET[5]
    - Is a set of Microsoft software technologies for connecting information, people, systems, and devices.
    - Will dominate e-business application develop initiatives during the next five years with Java
- Microsoft® .NET to satisfy OSS requirements above
  1. IDE, CBD and .NET libraries for rapid development of OSS
  2. Highly extensible architecture
  3. Windows application/Web application of VS.NET can make UI so cool
  4. COM+,MTS,MSMQ,SQL server and other products can be applied to the applications which are large-scale and distributed transactions
  5. Web service is going for international standards and other communication tools have many bridging product for interconnection

The OSS (Operations Support System) must be able to process lots of service orders and manage the networks. There are some general requirements for the OSS platforms. There have been so many discussions on how we can implement OSS with .NET platform. So, first, we have to check if .NET platform is applicable to OSS or not. Some considerations are as follows:

-Time-to-market : For quick implementation, we adopt CBD (Component Based Development) methodology and RAD (Rapid Application Development) IDE (Integrated Development Environment) VS.NET (Visual Studio .NET). Using VS.NET, software modules can be made to components like COM+ . Even web service modules can be considered as components. Components can be easily reused for rapid development and adaptation.

-Extensibility : Extensible architecture must be adopted in preparation for unexpected customer increase and new network services. We can make flexible architectures with .NET platform which can be scaled-out and scaled-up. Scale-out is the ability to grow by adding instances/computers in parallel to provide acceptable service levels. And scale-up is the ability to continue to grow within a single computer.

-Cool user interface including web-based UI (User Interface) can be made with WYSIWYG (What You See Is What You Get) method via VS.NET windows and web application.

-COM+ (Component Object Model Plus), MTS (Microsoft Transaction Server), MSMQ (Microsoft Message Queuing), SQL server (database), BizTalk server and others in .NET platform support large scale transaction processing/distributed transaction processing.

-Interoperability with other OSS, especially legacy system. Web service is going for international standards and other communication tools in .NET platform have many bridging products for interconnection.

-Low cost – This fact is really important. But we'll not mention on the cost problem in this paper.

Microsoft® .NET is a set of Microsoft software technologies for connecting information, people, systems, and devices. Many market research centers think it will dominate e-business application develop initiatives during the next five years with Java. In next sections, we will see .NET platform in detail and prove the capabilities of .NET with an OSS prototype. And then conclude if Microsoft® .NET has enough capabilities to satisfy OSS requirements or not.

.NET platform

- Web services
  - XML-based messaging as a fundamental means of data communication to bridge the differences that exist between systems
- ASP.NET
  - a programming framework built on the common language runtime that can be used to build powerful Web applications.
- ADO.NET
  - provides consistent, high-performance access to data, whether you're creating a front-end database client or middle-tier business object using an application, tool, language, or even an Internet browser.
- MSMQ (MS Message Queuing)
  - enables applications to communicate asynchronously across heterogeneous networks and systems that may be temporarily offline.
- COM+ (Component Object Model Plus)
  - a software architecture that allows applications to be built from binary software components.
- .NET Enterprise Servers
  - BizTalk server
    - A product for enterprise application integration (EAI) and business-to-business (B2B) integration.
  - SQL server
    - a relational database management and analysis system for e-commerce, line-of-business, and data warehousing solutions

XML Web services are the fundamental building blocks in the move to distributed computing on the Internet. Open standards and the focus on communication among applications have created an environment where XML Web services are becoming the platform for application integration. XML Web Services expose useful functionality to Web users through a standard Web protocol, SOAP (Simple Object Access Protocol). XML Web services provide a way to describe their interfaces called a WSDL (Web Services Description Language). XML Web services are registered with UDDI (Universal Discovery Description and Integration).

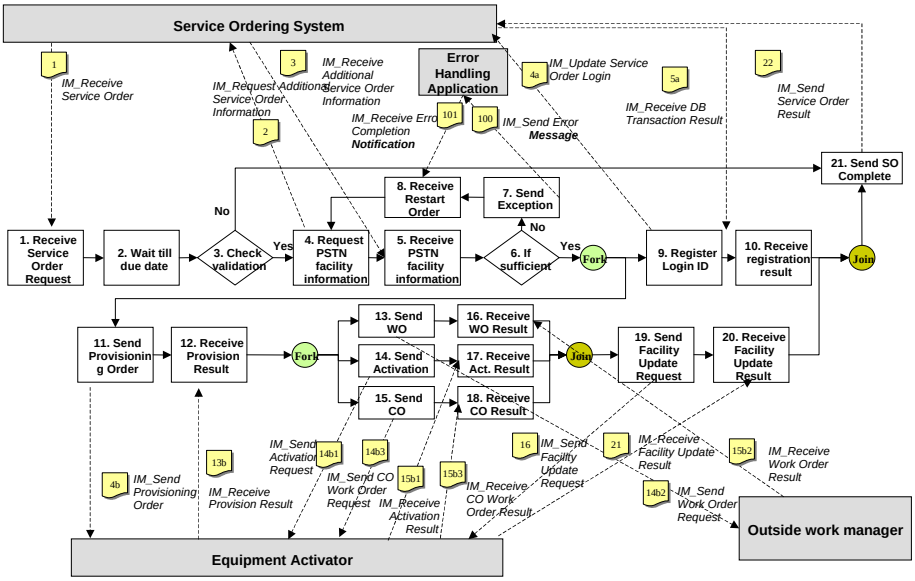
ASP.NET is a set of technologies for building Web applications and XML Web services. ASP.NET pages execute on the server and generate markup such as HTML, WML, or XML that is sent to a desktop or mobile browser. ASP.NET pages use a compiled, event-driven programming model that improves performance and enables the separation of application logic and user interface. ASP.NET pages and ASP.NET XML Web services files contain server-side logic written in any .NET-compatible language.

ADO.NET provides consistent, high-performance access to data, whether you're creating a front-end database client or middle-tier business object using an application, tool, language, or even an Internet browser.

MSMQ provides a messaging infrastructure and development tools for distributed messaging applications that enable applications on different systems to communicate with each other, even if systems and networks occasionally fail. Message Queuing provides guaranteed message delivery, efficient routing, security, support for sending messages within transactions, and priority-based messaging. COM+ is a software architecture that allows applications to be built from binary software components.

BizTalk Server provides a central, data-driven, integration server and a set of advanced productivity tools and services that enable you to build and deploy business processes for performing enterprise application integration (EAI) and business-to-business (B2B) transactions. With support for multiple transports and protocols, BizTalk Server helps you enable existing IT investments for XML and incorporate them into new e-commerce technologies.

Real Service Order Scenario



There are a lot of telecommunications services and lots of corresponding ways to manage them. But generally OSS business processes can be categorized as five major processes. Addition, deletion and modification of services, problem handling and monitoring of network devices. Addition is a process of creating a new customer entry in database and setting all the required network equipments to provide the requested service. Deletion is a process of canceling the service, and modification is a service change. Problem handling includes customer data checking, testing the equipment, outside worker dispatch and etc. Monitoring is active surveillance process for network resources. Works in those five categories are not very different especially in technology viewpoint. So, we choose ‘ADSL Addition’ scenario as a prototype scenario. Addition process includes service order validation, due date waiting, registering customer information including login ID, activation of network equipment and order dispatch for office and outside workers.

We’ll simplify this process but maintain core portions for the prototype especially in technology aspect at the next section.

Simplified Service Order Scenario

- Logical Design
  - A simplified business process of service order management for ADSL service
- Descriptions
  1. Receive Service Order request
  2. Check order type and save it in the order database
  3. Select a free port from facility database
  4. Activate previously selected port. [This step will not be tested in this paper.]
  5. Update the status of the selected port in facility database
  6. Insert or update customer information in customer database
  7. Update the status of the order in order database
  - ❖ If error occurs at every step, Error management routine will take care of that.

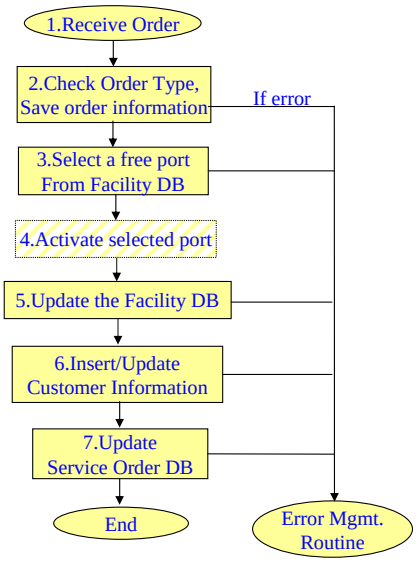


Fig.1 Service Order

This is a simplified Addition service order scenario. Complex algorithms and repetitive logics will be simplified and omitted. The descriptions of the Addition order process are as follows:

1. Receive Service Order request from the BSS (Business Support System).
2. Check order information and validate the service data. And save it in the order database.
3. Select a free port from facility database. Usually network resource selection has a very difficult algorithm, but it will be omitted for the convenience of this prototyping.
4. Activate previously selected port. At this step network activation and settings in the office and outdoors occurs. This step will not be tested in this paper.
5. Update the status of the selected port in facility database after the field work completion.
6. Insert or update customer information in customer database
7. Update the status of the order in order database and report the order completion to BSS.

Technical Architecture

- Message Oriented 3 tier Architecture
- Technologies adapted
  - 1. ASP.NET web application
    - Operators enter the orders via web application
  - 2. ADO.NET DLL
    - Connect database
  - Web services
    - Database APIs of order, facility and customer DB are exposed via web services
  - MSMQ
    - transfer order data from web server to application(biztalk) server
  - 3.4.5.6.7.BizTalk server
    - Receive order from MSMQ
    - Workflow control SQL server
  - SQL server
    - Store order, facility and customer information
  - COM+
    - BizTalk orchestration calls COM+ for database manipulation. COM+ calls web service of DB APIs

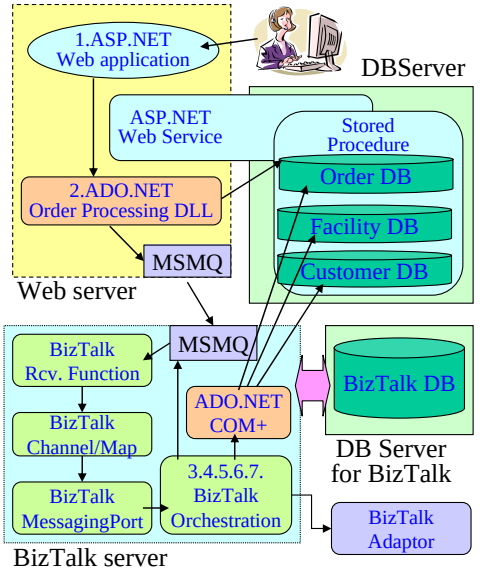


Fig.2 Technical Architecture

Web servers have ASP.NET web applications which enable the operators to enter service orders. ASP.NET server controls enable an HTML-like style of declarative programming that let us build great ordering pages. Displaying customer data, validating operator input, and uploading files are all easy. Best of all, ASP.NET pages work in all browsers including Netscape, Opera, AOL, and Internet Explorer. ADO.NET order processing DLL (Dynamic Link Library) connects the order database and save the service order input data.

We could build robust 3-tier applications using MOM(Message Oriented Middleware), MSMQ(Microsoft Message Queuing). MSMQ enables web servers to send messages with delivery guarantees that can be applied on a message-by-message basis. When networks go down, biztalk servers are offline, or computers containing message queues fail, MSMQ will ensure that messages get delivered as soon as connections are restored or applications and machines are restarted. MSMQ implements these guarantees using disk-based storage mechanisms and log-based recovery techniques. MSMQ messages can contain data in any format that is understood by both the sender and the receiver. The web servers send the orders in XML format.

Workflows and EAI functions are conducted by BizTalk server. The COM Component shape enables us to use preexisting components to perform database interactions within an XLANG schedule. Because COM technology is synchronous, there is always a bidirectional flow of messages when an action is performed. In contrast, the flow of messages for an asynchronous technology is in one direction. Details on biztalk will be explained later.

There are two kinds of databases. One is user data such as order data, facility data and customer data. These are exposed using web services. Also, we can directly manage the data using stored procedure. The other is for biztalk. BizTalk use its own SQL database to manage the states of workflow.

This architecture can be applied to the real order scenario as well as the simplified order scenario.

Web tier

- ASP.NET
  - Can be developed using Visual Studio .NET
  - ASP.NET Web Application : use template for creating a Web site with static or dynamic HTML pages as the user interface
  - ASP.NET Web Service : use template for creating Web Services that can be called through XML SOAP interfaces.
- Class Library
  - Order Processing DLL : use template for creating classes that will be used in other applications. Similar to a DLL
- XML message
  - Text data are converted to XML message in web tier
  - BizTalk Editor
    - A tool with which we can create, edit and manage XML specifications. With this we can create a specification based on a specification template, and existing schema, certain types of document instances, or a blank specification.

Fig.3 VS.NET Web Application

Fig.4 BizTalk Editor(XML message)

We used ASP.NET web application to receive service orders from BSS. It uses VS.NET template for creating a Web site with static or dynamic HTML pages as the user interface as shown in Fig.3. Web application take the HTTP request and deliver it to order processing DLL. Order Processing DLL is developed as class library in VS.NET. Class library template is for creating classes that will be used in other applications. This DLL save the service order information to order database and convert the order information to XML message format. XML specification is made by BizTalk Editor as shown in Fig.4. BizTalk Editor is a tool with which we can create, edit and manage XML specifications. With this we can create a specification based on a specification template, and existing schema, certain types of document instances, or a blank specification. ASP.NET Web Service uses a template in VS.NET for creating Web Services that can be called through XML SOAP interfaces. In this implementation, database entries can be accessed via stored procedure and web service API (Application Programming Interface). Web service is a good API for database because it can be easily accessed regardless of the platform of the clients.

Application tier (BizTalk)

- BizTalk Receive Function
  - monitor documents posted to specified locations (directory, queue or by web site).
- BizTalk Mapper
  - Build document maps that allow applications and business partners who use different document definitions
- BizTalk Messaging Manager
  - Use this wizard-based tool to rapidly define trading partner relationships.
- BizTalk Orchestration Designer
  - Visually define and build robust, distributed business processes.
- Adapter
  - integration with popular choices such as SAP, products from Siebel, Onyx and more.
  - integration with common transports such as J2EE, MQSeries and CICS, FTP and more.

Fig.5 XLANG schedule – Business process

Fig.6 BizTalk orchestration – Data page

In Fig.2 we can see various parts of biztalk server. When we need to receive service orders from a receive location and submit them to BizTalk Server, we configure receive functions to process the data. BizTalk Server 2002 supports three types of receive functions: File, Message Queuing and HTTP. We used MSMQ receive functions. After receiving, a channel takes the order. A channel is a set of properties that designates the source of documents or messages and defines specific processing steps that are performed by biztalk messaging services before a document is delivered to the destination designated by the messaging port or distribution list with which the channel is associated. We can adapt map which is made by biztalk mapper. BizTalk Mapper is a highly graphical tool that lets us define transformations by simply drawing lines between records, fields, and functoids. Messaging port gets the service order from the channel and deliver it to biztalk orchestration or workflow. A messaging port is a set of rules that trading partner organizations accept for sending documents to one another. In this case, the port guide the messages to go to the workflow object.

An XLANG schedule describes the business process and the binding of that process to implementation technologies. Also, this is called as biztalk orchestration. The process of service order is drawn by biztalk orchestration designer and compiled to XLANG language. It is activated by messaging port. Every XLANG schedule drawing has a data page. The Data page displays: One Message shape for every message in the XLANG schedule. One Constants message. One Port References message containing a port field for each port within the XLANG schedule drawing. Diagrammatic connections showing the flow of data between the message fields.

We can use adapters for integration with popular choices such as SAP, products from Siebel, Onyx and etc and integration with common transports such as J2EE, IBM MQSeries, FTP and more. We have lots of EMS(Element Management Systems) implemented with Java or other technologies based on Unix or linux. When we communicate with these modules, we have to use adapters.

But BizTalk 2002 orchestration designer has a fatal defect. Fig.5 and Fig.6 shows the biztalk orchestration. When the business process becomes more complex, the drawing

Data tier

- These are simplified database schema just for this prototype
  - Order database
    - SA_ID : Service Agreement ID – unique ID for a service contract
    - ORD_ID : Unique service order ID
    - ORDER_TYPE : New, Delete, Modify,…
    - SVC_TYPE : ADSL, PSTN, …
    - STATUS : status of order
  - Customer database
    - CUST_NAME : name of a customer
  - Facility database
- DataBase APIs
  - Stored Procedure
  - Web Service

NO	Column	Type	Description	Value
1	SA_ID	string	Service Contract ID	(unique, random)
2	ORD_ID	string	Service Order ID	(unique, seq)
3	REQ_DATE	DATETIME	Requested Time	
4	ORDER_TYPE	string	Type Of SO	NEW
5	SVC_TYPE	string	Kind of Service	PSTN, ADSL
6	CUST_NAME	string	Customer's Name	(unique, random)
7	PORT_ID	string	Facility ID Selected	(unique, random)
8	COMP_DATE	DATETIME	Finished Time	
9	STATUS	string	Status of SO	REQ, DONE

Table 1. Order DB schema

NO	Column	Type	Description	Value
1	CUST_NAME	string	Customer's Name	(unique, random)
2	SA_ID	string	Service Contract ID	(unique, random)
3	CREATE_DATE	DATETIME	Created Time	
4	SVC_TYPE	string	Kind of Service	PSTN, ADSL
5	PORT_ID	string	Facility ID Selected	(unique, random)
6	STATUS	string	Status of Customer	NORMAL

Table 2. Customer DB schema

NO	Column	Type	Description	Value
1	PORT_ID	string	Equipment (PORT) ID	(unique, random)
2	PORT_TYPE	string	Type of the PORT	ADSL, PSTN
3	STATUS	string	Status of the PORT	FREE, OCCUPIED

Table 3. Facility DB schema

These are simplified database schema just for this prototype. Some descriptions are as follows :

•SA_ID : Service Agreement ID – unique ID for a service contract

•ORD_ID : Unique service order ID

•ORDER_TYPE : New, Delete, Modify,…

•SVC_TYPE : ADSL, PSTN, …

•STATUS : status of order

•CUST_NAME : name of a customer

Database APIs are implemented with stored procedure and web services. Stored procedures can make managing our database and displaying information about that database much easier. We can save precompiled collections of SQL statements and optional control-of-flow statements stored under a name and processed as a unit. Stored procedures are stored within a database and can be executed with one call from an application; and allow user-declared variables, conditional execution, and other powerful programming features. The advantages of using stored procedures are :

•We can execute a series of SQL statements in a single stored procedure.

•We can reference other stored procedures from within our stored procedure, which can simplify a series of complex statements.

•The stored procedure is compiled on the server when it is created, so it executes faster than individual SQL statements.

Test Results (1/2)

- H/W Conditions
  - BizTalk, Web server : Intel DP 2.4GHz 2-CPU, 2GB RAM
  - Database server : Intel MP 2.0GHz 4-CPU, 4GB RAM
- S/W Conditions
  - OS (Windows 2003 server RC2, .NET Framework v1.1), IDE (Visual Studio .NET 2003 Beta)
  - EAI/Workflow engine (BizTalk 2002), Database (SQL server 2000)
- Throughput
  - There are pre-built 100,000 entries in Facility table.
  - Average order processing rate is over 13.4 orders/sec as shown in Table 4.
  - When DB access components registered in GAC not COM+, performances degraded by 17%
- Response Time
  - Average Response Time : 1.105 sec
  - Maximum Response Time : 2.100 sec (At the first time only : 6.190 sec)
  - Minimum Response Time : 0.143 sec
  - Standard deviation : distributed evenly from min. value to max. value

No. of Pool	Avg. completed schedule	Min. completed schedule	Max. completed schedule	Concurrent access	Failure
25	13.478/sec	11.0	15.0	50	0
50	13.443/sec	2.0	23.0	50	0
75	13.402/sec	11.0	14.0	50	0

Table 4. Throughput

Tests are repetitively conducted to confirm the stability of all the systems, tools and logics. There are pre-built 100,000 entries in facility database table. And other databases are empty at the start, but when every order comes in, corresponding customer information is automatically generated. Average input XML message size is about 250 byte. Some results on throughput are shown here:

•Average order processing rate is about 13.4 orders/sec

•DB access components can be registered in Global assembly rather than as COM+, but performances are degraded by about 17%.

Some results on response time (Elapsed time for an order completion) are shown :

•Average Response Time : 1.105 sec

•Maximum Response Time : 2.100 sec

•Minimum Response Time : 0.143 sec

•Standard deviation : distributed evenly from min. value to max. value

And we could find that there must be some considerations for BizTalk server. The number of input orders should be regulated using object pooling or other customized logic, if not, processing rates become lower and failure cases occur. BizTalk Server takes advantage of COM object pooling, and enables us to restrict the number of active schedule instances. We can set a maximum number of schedule instances for a pool. When this maximum has been reached, no additional instances of schedules that use that pool are created until a schedule either completes or is dehydrated. The default maximum is 25 instances. We can also set a timeout period on a pool. This is the length of time a schedule instance pending activation will wait for a slot to become available within its pool. We can improve performance by setting a minimum pool size. The minimum size determines how many pooling objects will be maintained in memory, so that they do not have to be created on demand each time a schedule instance is activated in that pool. The default minimum is one object.

Test Results (2/2)

- BizTalk
  - Completed Schedule Instances/sec (Red Line) : The rate at which schedule instances successfully complete
  - Running Schedule Instance (Blue line) : The current number of running schedule instances
  - Failed Schedule Instances/sec (Pink) : The rate at which schedule instances fail
- Memory
  - Process : Private Bytes (Brown line) : We can detect memory leaks (Results: Acceptable)
  - Available Bytes(Sky Blue line) : We can detect usable memory amounts. This value must be maintained above 4MB. If not, events which means system virtual memory lack will occurs. Available bytes must be maintained about 10% of real memory. (Results: Acceptable)
- System
  - Processor queue length : If this value is high, we can get more performance with faster CPU. One system has only one processor queue even in multi-processor system. 10 ready threads per processor are acceptable. (Results: Acceptable)
  - %Processor time : It is calculated by monitoring the time that the service is inactive, and subtracting that value from 100%. 0% means this server stops and 100% means this server is tremendously loaded. (Results: Acceptable)

Fig 7. Performance monitor - biztalk

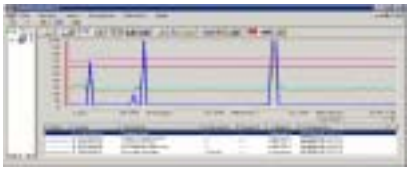


Fig 8. Performance monitor - memory

Fig 9. Performance monitor - system

The testing tools are as follows :

•VS.NET ACT (Application Center Test) : designed to stress test web servers and analyze performance and scalability problems with web applications, including Active Server Pages (ASP) and the components they use. It generates new orders automatically.

•SQL query analyzer : a graphical user interface for designing and testing Transact-SQL statements, batches, and scripts interactively.

•XLANG Monitor : monitor XLANG schedule events and see the progress of the schedule instances.

•Performance Monitor : Included in Windows OS. This is a tool for monitoring the performance of Windows servers. It uses a series of counters to track data, such as the number of processes waiting for disk time, the number of network packets transmitted per second, the percentage of processor utilization and biztalk and sql server performance counter. Some of the performance counters are explained above

We can have some lessons from this prototype. It takes very long time to conduct tests and make results. So, before testing, we must decide test items, perfect test methods/routines/plan and set up an exact configuration. In some cases, we have to make complex test scripts, so we have to think in advance and pump up our skills. We have to set appropriate number of maximum activated schedule instances in a pool considering XLANG scheduler throughput. If too many instances are initiated, their waiting time can be too long to be done well. Maintain pool limitation reasonably above the average completed instances/sec. If COM component port implementations in BizTalk Orchestration are registered in COM component services, we can get more performance than in the case of being registered in Assembly (GAC).

Conclusion & Future Work

- Possibilities for .NET based OSS are shown
  - With .NET platform, developers can work in a very efficient way
  - Test results show .NET platform can cover OSS business process
  - BizTalk 2002 is not good for complex workflows
- Future works
  - Test various implementations of BizTalk (e.g. Transaction shape, etc.)
  - BizTalk adaptor test
    - Lots of existing Unix-based activator/tester can be connected
    - But if used too many adaptors, performance degradation occurs
  - Adaptability of BizTalk 2004 test
    - BizTalk 2004 Beta released May 2003
    - Test new and enhanced features such as workflow functionalities

Conclusion and Future work

We implemented a simplified ADSL Addition process using .NET platform which is based on web service, ASP.NET, ADO.NET, CLR (Common Language Runtime) and so many libraries and enterprise .NET servers such as SQL server 2000, BizTalk server 2002. In developing phase, .NET platform made the developers work in very efficient way, and the test result showed .NET platform can satisfy the OSS requirements well. But this is a small project to see the 'possibilities' of .NET platform for OSS, we must proceed to test various things.

References

1. Jae-Oh Lee, "Enabling Network Management Using Java Technologies", IEEE Commun. Mag., Jan. 2000.
2. Steve Vinoski, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments", IEEE Commun. Mag., Feb. 1997.
3. Seong Ik Hong, Mun Jo Jung, "A Study on CORBA-based ATM.ADSL Network Resource Management Systems Design and Construction Project using Object-oriented Network Resource Modeling Technique", KICS Mag. (Korean), Jun. 2001.
4. "OSS Java Community Process Program", http://jcp.org/jsr/tech/oss.jsp
5. "OSS Working Group", http://www.microsoft.com/SERVICEPROVIDERS/ossbss/osswg.asp
6. "Microsoft Developer Network", http://msdn.microsoft.com
7. http://www.microsoft.com/biztalk
8. http://www.microsoft.com/sql
9. "Microsoft .NET vs Java", Mark Driver, Gartner Symposium Itxpo 2002
10. "The Essential Client/Server Survival Guide", 2nd edition, Robert Orfali, Dan Harkey, Jeri Edwards, Wiley