Improving Web Server Performance by Distributing Web Applications

Mital Kakaiya[1], Sushant Goel[1], Hema Sharda[1], David Taniar[2]

[1] RMIT University, School of Electrical and Computer Engineering, Vic-3000, Australia
hemas@rmit.edu.au
[2] Monash University, School of Business Systems, Vic 3800, Australia
David.Taniar@infotech.monash.edu.au

Abstract. Distributed interNet Application (DNA) covers a wide range of topics. DNA is a methodology that specifies how to distribute Internet application on various Web servers. DNA helps to generate scalable, reliable enterprise applications. It provides load-balancing techniques to distribute load on multiple Web servers. This paper describes DNA methodology for a distributed application, which enables better performance, availability and service to clients. This paper also provides comparison of application's performance and scalability between DNA and non-DNA application. The comparison clearly indicates web server performance improvement using DNA methodology. CPU usage improvement statistics are also provided in this paper. Choosing optimized technology is one of the major criteria in a distributed system to achieve best result. Current major industries are moving towards distributed Internet application solution for global market strategies.

1 Introduction

Distributed computing is a technique, which converts a huge software problem into smaller parts and distributes the smaller segments among several computers. It is a complicated job to develop a large application, which is distributed among several servers. Distributed interNet Application (DNA) provides a methodology for such applications, which are easy to understand and implement onto multiple servers. DNA architecture is not a solution – but rather a methodology to solve a complex distributed problems. In other words, DNA is just an abstract pattern. It is a software application engineering design, which generates a solution to a set of common generic problems.

The 2-tier architecture works well up to a medium size application requirement. If application is huge, the single server cannot process all user requests. Some application might require lots of memory and processing work. The server needs to process lots of instructions and data to generate final required output for clients. In major cases, increase in hardware speed is not a solution for application's performance, scalability and reliability. To overcome these difficulties, the single server's load could be distributed among multiple servers. In distributed computing, multiple servers are connected together to perform a specific task in a distributed

T. Böhme, G. Heyer, H. Unger (Eds.): IICS 2003, LNCS 2877, pp. 156-167, 2003.

environment. The multiple servers can be connected in horizontal as well as vertical hierarchy. DNA helps to develop enterprise applications as a scalable, secure, robust and reliable manners [1]. The goal of the distributed architecture is to distribute the processing load across as many resources as necessary; it doesn't mean to distribute the data within the system [1]. DNA is an abstract idea, which helps to understand design of multi-tier client/server application. There are no coding practices, special notation or even restrictions on the technologies to use. Developer can develop and deploy DNA architecture applications without any restriction of using DNA methodology.

Due to distributed development and communication demand, new network-based technologies were invented, which enabled faster, secure and reliable communication protocols and standards between computers within a network. The Internet and web based applications with open Internet standards have great ability to communicate across machine boundaries and provide information in a reliable, secure and efficient way.

This paper discusses how to distribute an Internet application on multiple web servers, which provides scalability, reliability, availability and better performance. This paper also describes performance tests between websites based on DNA methodology and without DNA methodology. It provides clear understanding of a performance improvement and necessary web server load balancing.

2 Distributed Component Technology: A Background

Traditional applications are hardly distributed among various servers due to limited resources, difficulty in developing and managing. Development of distributed application cost is very high and also contains high risk [9]. There are several technologies available to develop robust and reliable distributed development environment, like Component Object Model, Distributed COM, Transaction servers etc.

2.1 Component Object Model (COM)

The traditional applications are made of single monolithic binary file. Once the application file is compiled and published, it does not change until next version of the application is developed and shipped. If there are any changes into the application customers have to wait for the next rebuild. To find out bugs into monolithic application is critical, because incorrect functionality at one point might affect other parts of the application's functionality. It is difficult to find out exact incorrect part in a huge application.

COM [11] is a specification, which specifies binary standard. COM is a platform independent, distributed; object-oriented system for creating binary software that can interact with applications. It defines a standard for component's interoperability and it is available on multiple platforms like Windows, Macintosh, and Unix. Virtually, any programming language can be used to develop a component. This standard is helpful

when different people at different locations develop different parts of the application. COM is easily extendable and contains robust architecture. COM consists of a binary code, which is distributed as a dynamic link library (DLL) or an executable (EXE). The COM is not only specification. The COM has the COM library, which called "COM API". It provides components management services that are useful for all components. The COM component provides various advantages to an application like: Dynamic linking, increased performance, scalability, language independence, version compatibility etc. COM+ is an extension of COM and it provides additional helpful services like manage transaction, Just-In-Time (JIT) activation, advanced security, object pooling, queued components, loosely coupled events, basic interception services, deployment and administration. COM+ services help to develop fast, powerful and robust component for an enterprise application [10].

2.2 Distributed COM

The Distributed Component Object Model (DCOM) is an extension of COM with additional functionality of communication across machine boundaries. This protocol enables software components to communicate directly over a network in a reliable, secure and efficient manner. The concept of DCOM was introduced in 1992 by developing Dynamic Data Exchange (DDE) protocol. The DCOM also helps to reschedule one machine's components to another machine's components.

2.3 Load Balancing

A distributed application design consists of various components, which interact with each other and provides reliable required output. The component distribution task requires careful planning and analysis to distribute application on web servers. Load balancing, performance and scalability become key aspects of the design process in a distributed application [7]. Component's runtime load, architecture including logical packaging, physical deployment, remote server workload analysis, and available network bandwidth needs to be considered [3].

Web server receives requests from clients randomly. Server needs to respond to the client's request, so it creates many instances of component within distributed architecture. Due to uncertain request interval, some servers are heavily loaded, while others are lightly loaded [5]. Uneven distribution of the load disturbs performance of the distributed application. Load balancing algorithms helps to reduce uneven load, which improves performance by distributing the component more evenly on various servers. The performance of web server directly reflects with load balance. The over loaded server or unbalanced system provides poor performance result.

The number of variety of applications using the distributed architecture is increasing and the expectancy of customers is also increasing. As the number of users of any application increases the response time also increases [6]. The over-loaded server may not response to all clients, which can result in the timeout of the request. There are two ways to meet the ever-increasing demand of the Internet server performance.

The Modern Approach: The better way of solving the problem is to use a cluster of servers serving clients with one and the same service using synchronized contents. When the requests for the Internet service increases new servers are added to the cluster to meet the increased traffic requirements. The traffic is distributed among the individual servers to balance the load on each server.

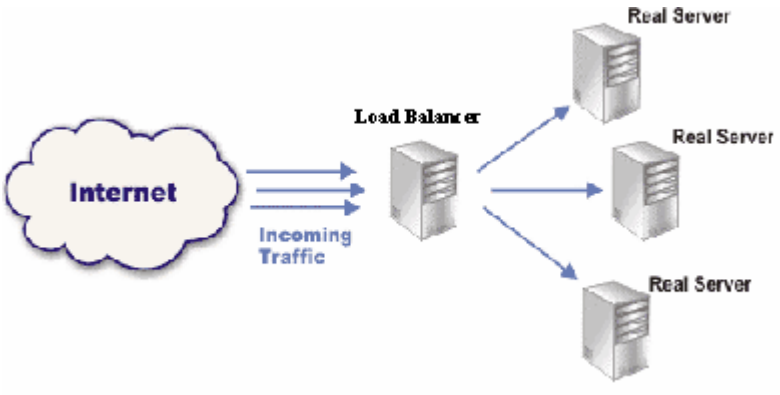


Fig. 1. Modern approach of load balancing

The Traditional Approach: The first way is the single server solution in which the server is upgraded to a higher performance. There is a problem in this approach that soon this server can be overloaded again and a next upgrade will be required. The whole process of upgrading is complex, time consuming and expensive.

2.4 Load Balancing Algorithms

The load balance algorithms are helpful to share load on various servers. Major load balancing algorithms add load state information to existing client requests. There are two types of useful load balancing techniques, which are static and dynamic algorithm [2]. The static algorithm does not distribute request based on current load on web server. The dynamic load-balancing algorithm calculates current load on web servers and forwards request to minimum load server [4].

Round-Robin Algorithm: Round-Robin algorithm is the simplest form of load balancing algorithm. The round-robin scheduling algorithm sends each incoming request to the next server in it's list. Thus in a three server cluster (servers A, B and C) request 1 would go to server A, request 2 would go to server B, request 3 would go to server C, and request 4 would go to server A, thus completing the cycling or 'round-robin' of servers.

Weighted Round-Robin Algorithm: The weighted round-robin scheduling is better than the round-robin scheduling, when the processing capacity of real servers are different. The weighted round-robin algorithm assigns each server hidden weight load based on processing power. However, it may lead to dynamic load imbalance among the real servers if the load of the requests varies highly.

Least-Connection Algorithm: The least-connection scheduling algorithm directs network connections to the server with the least number of established connections. This is one of the dynamic scheduling algorithms because it needs to count live connections for each server dynamically. The least-connection scheduling cannot get load well balanced among servers with various processing capacities. The faster server can process thousands of requests and keep them in the TCP's TIME_WAIT state.

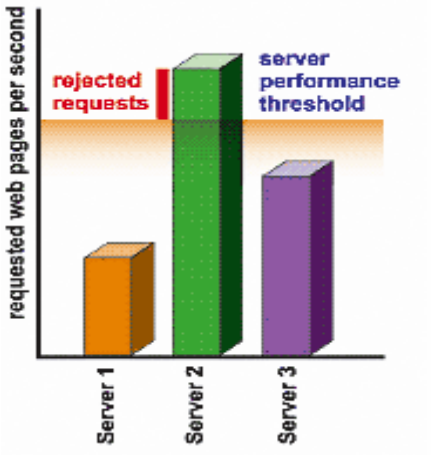
3 Performance Enhancement Using DNA

With the single server Internet application the chance of the server being unavailable is high. Adding new servers can increase availability significantly. If one server has 95-percent availability that would mean it's not available for an average of 1.2 hours a day. The probability of the server failing at a given moment is 0.1. Adding one additional server decreases the probability that both servers will fail at once to 0.1*0.1= 0.01. The likelihood of one of the servers being available is increased to a much-improved 99 percent. The algorithms discussed in the previous section helps to distribute the load to different servers. But the load on the servers is not evenly distributed, as the work to be allocated to the servers is not determined dynamically.

Optimized Weighted Round-Robin Algorithm: This algorithm is similar with the Weighted Round Robin algorithm. But the server can be dynamically assigned a weight depending on its current availability and current load.

Weighted Least Connections: As we know from the previous section, least connection algorithm cannot balance the load effectively among the servers if the processing capacity of the servers is different. In weighted least connection algorithm a performance weight is assigned to each real server. Larger percentage of live connections is assigned to the server with the higher weight value. If number of connections are `C1, C2, C3,...,Cn` and the performance weight assigned to the servers are `W1, W2, W3,...,Wn,` then as per the weighted least connection any new connection will be assigned to the server with minimum `Ci/Wi` value (where i `= 1,2,3,...,n`). The advantage of using this algorithm is any new connection will be allocated to the least loaded server.

Any cluster of servers or a server farm not properly balanced; may reject client requests because some of the servers may be at their performance level threshold but others may be still well under the threshold. But using the proper load-balancing algorithm like the “Weighted Least Connection” may distribute the load evenly in all the servers in the server farm. The following figures (2 & 3) show the comparison between the two:



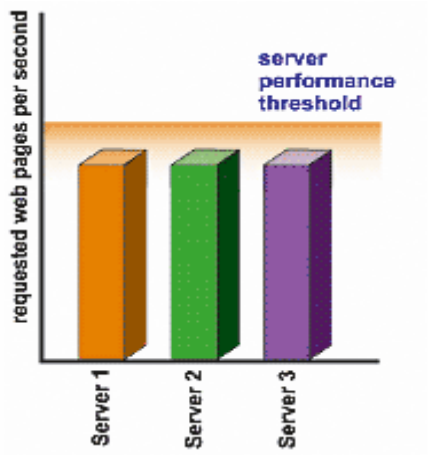


Fig. 2. Improperly balanced server-farm **Fig. 3.** Properly balanced server-farm

3.1 Distributed interNet Application

Distributed interNet Application (DNA) describes architecture for building multi-tier distributed computing solutions. The major tier of the DNA is Presentation tier, Business tier and Data tier. Each tier represents own services to the application. Each layer or tier usually resides on a different virtual machine. The presentation layer only communicates with the application or middle layer, which contains the business objects. The middle layer handles the applications processing logic and in turn communicates with the data access layer, such as SQL server. A three-tier application allows the implementation of thin client and is much more flexible and easy to maintain than a two or one-tier. For example, the data storage layer can be substituted completely without having to change any code at the presentation layer.

There could be more than one web server and application logic server depending on the traffic. Hence there could be a **Load-Balancing Layer** in addition to the Presentation-Layer, Business-Layer and Data-Layer. Load-balancing solutions present a single system image to clients in the form of a virtual host name, and distribute client requests across multiple application servers.

Presentation-Layer: The presentation layer handles the basic user input and output. It is responsible for providing the graphical user interface. This layer collects input from client and sends user input to business services for further processing. Some of the presentation layer tools are: DHTML, VBScript, Jscript, Browsers, activeX controls.

Business-layer: Business layer is the core of the application. The business service receives user input from presentation service tier, performs business operation automated by an application, interfaces with data service as necessary, and returns result to presentation service. Some business layer tools are: COM+, IIS Server, ASP, ADO.

Data-Layer: Data service receives requests from business service, retrieve data from database, and check data integrity and returns result to business services. Some Data layer tools are: Exchange server, OLE DB providers, SQL Servers and other DB servers.

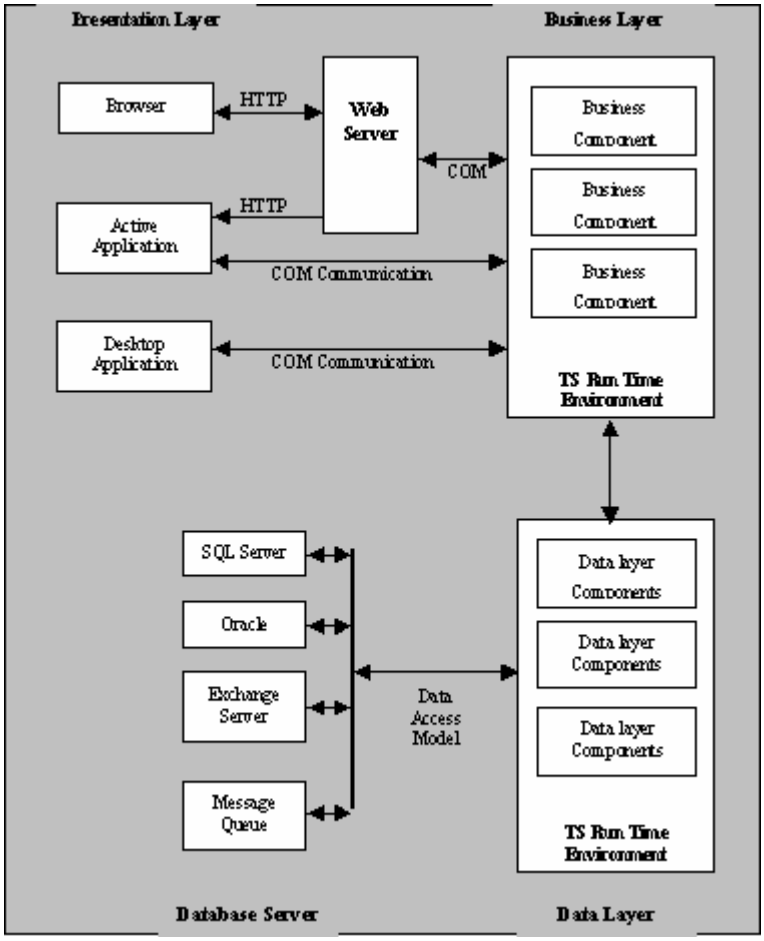


Fig. 4. DNA architecture overview

4 Performance Evaluation

To study the behavior of the Distributed interNet application and the application which is not distributed we performed the test on the following test bed:

- Processor: Intel P-III, 850 MHz.
- SD-RAM: 128 MB

- Hard Disk: 5 GB
- Network Card: Intel 8255X based PCI Ethernet Adapter (10/100)
- Windows 2000 server
- IIS 5.0
- MS-SQL Server 2000
- MS- Web Application stress (WAS) tool Stress Version: 1.1.293.1

The WAS tool generates arbitrary requests and is used to measure the performance level of the server. The test is done over two terminals. One terminal is dedicated to run the application, the other terminal is used to run the WAS tool and any activity other than running the application server. The test is conducted over two terminals to ensure that the developed application gets full attention of the processor in which the application is running so that the results are correct.

The tested application is a very standard e-commerce web site with member's login page, new client's registration page, product display page, shopping cart page, payment page etc. the WAS tool generates random requests for a specific time and measures the response from the server. The testing is done for two different cases:

I. The application is developed using Distributed interNet Application technology, three separate layers are created as discussed in the previous section.
II. The application is developed without using the component technology.

4.1 Sample Code at Different Layer (Using DNA Technology)

The **Presentation Layer** for the application consists of HTML and DHTML coding. The HTML coding is an important interface for users to communicate with website. The following lines show an example of user interface for web browser:

```
<html>
<head> <title>Home Page</title> ..... </head>
<body> ..... </body>
</html>
```

For presentation of a website, It needs information from database and various resources. The application is not able to communicate directly with database. It needs to pass through business layer component to maintain application's security.

The presentation layer creates an instance of a business layer component. The business layer component is registered into MTS environment. It will create object in MTS runtime boundary. The following lines shows create instance of `BusinessCOM.cbCart`, which is defined as an `objCart` Object.

```
Dim objCart
'Create instance of a component
Set objCart = Server.CreateObject ("BusinessCOM.cbCart")
```

Business Layer shows component based development code using Microsoft Visual Basic development language. Class initialize and terminate events are defined which is called automatically based on object creation and deletion.

```
Private Sub Class_Initialize()
  ' Component Initialized variables here
End Sub

Private Sub Class_Terminate()
  ' Component Terminate variables here
End Sub
```

All required variable declaration has been defined inside InitializeVariables and TerminateVariables functions. These functions contain general required initialization and termination declaration for components, so it needs to call from all components. It saves time and cost of development process. Modification at one function reflects changes into all components. The business component defined in this application has a `Get property` that returns the value of specified variable. Some of the methods defined in the business Layer are `AddCart(adds new product in the shopping cart), UpdateQty, ClearCart, TotalProductPrice, IsValidUser` etc. Following is the sample code for `ClearCart` function.

```
' Name of function: ClearCart
' Purpose          : Clear all products from a Cart.
' Returns        : true indicates Successful.

Public Function ClearCart() As Varient

' Assume successfully not clear products from cart

  ClearCart = False
  Dim intCountCart As Integer

   'loop until all products are clear.

  For intCountCart = 0 To MAX_CART
    TCart(intCountCart).m_intProductID = 0
    TCart(intCountCart).m_strProductDesc = ""
    TCart(intCountCart).m_intProductQty = 0
    TCart(intCountCart).m_curProductPrice = 0
  Next intCountCart

  'Successfully clear products from cart

  ClearCart = True
 End Function
```

The cart retains in memory up to user's session time. It is automatically destroyed when user's session is destroyed. It does not need to connect with database and store values. This component is not required to connect with data layer component for shopping cart operation. This component has no data layer functionality. It still uses data from database by accessing other data layer components. If username and password is provided non-zero length and valid values, the business layer component creates an object of a data layer component's customer class.

```
'Variable Declaration
Dim objCustomerData As DataCOM.cdCustomer
‘Create Object of Business Layer Customer class
SetobjCustomerData = CtxCreateObject
                                (TProgID.Data_cdCustomer)
```

The object is created using `CtxCreateObject` method. This method enables to create a new instance inside MTS runtime. The MTS handles to minimize impact of memory allocation and resources. This helps to improve overall performance.

Data layer is useful to communicate with database. The data layer is developed on Microsoft Visual Basic environment. The data layer always check the data validity before modify any permanent changes into a database. If data validity is correct and it could not destroy any current information from database, it sends login request to database objects. The data layer does not directly communicate with database table objects. The database forwards request to stored procedure of SQL Server and passes all required information to it. The following code creates ActiveX Data Object (ADO) Connection and Command objects. The object is running inside MTS environment by calling CtxCreateObject method.

```
 Dim objCmd As ADODB.Command
‘Create ADO Command Object inside MTS
 SetobjADOCommand=CtxCreateObject
                            (TProgID.Data_clsADOCommandC)
'open the connection object
 Set objConn = objADOCommand.OpenDB()
'Command Object calls Stored Procedure of a database
 Set objCmd = objADOCommand.LoadProc(objConn,
             TDB.TStoredProcName.prc_tblCustomer_login)
```

The Command object executes stored procedure, which returns a result based on arguments. The store procedure helps to execute faster query and data access.

Test Results

Two website test is taken using same hardware and test configuration: DNA and NODNA. The “DNA” indicates test data, based on DNA methodology web application. The “NODNA” indicates test data information using non-DNA methodology. The paper describes both website test reports and performance comparison. The DNA website is developed based on DNA methodology and sample DNA code described in the above section. NODNA website does not follow DNA methodology rules. NODNA website connects directly to database without interfere with business layer or data layer. NODNA website has no business layer or data layer.

Test results for DNA application:

```
DNA Overview
=========================================
Report name                          : DNA
Run length                           : 00:01:00
Web Application Stress Tool Version: 1.1.293.1
Number of test clients               : 1
Number of hits                       : 3989
Requests per Second                  : 66.39
%processor time                      : 83 %
Connection attempts/Sec.             : 71
Request handeled/Sec.                : 43

Socket Statistics
-----------------------------------------
Socket Connects                       : 4594
Total Bytes Sent (in KB)              : 1673.44
Bytes Sent Rate (in KB/s)             : 27.85
Total Bytes Recv (in KB)              : 23171.77
Bytes Recv Rate (in KB/s)             : 385.63
```

Test result for NONDNA application:

```
NODNA Overview
=========================================
Report name                          : NODNA
Run length                           : 00:01:00
Web Application Stress Tool Version:1.1.293.1
Number of test clients               : 1
Number of hits                       : 1222
Requests per Second                  : 20.36
%processor time                      : 90 %
Connection attempts/Sec.             : 20
Request handeled/Sec.                : 13

Socket Statistics
-----------------------------------------
Socket Connects                      : 1367
Total Bytes Sent (in KB)             : 604.34
Bytes Sent Rate (in KB/s)            : 10.07
Total Bytes Recv (in KB)             : 20758.11
Bytes Recv Rate (in KB/s)            : 345.89
```

4.2 Result Discussion

From the above mentioned test results it is very obvious that the number of requests handled by the DNA application is much higher than the NONDNA application. The "%processor time" is an important criterion during performance test. If %processor time goes above 90% of total time, it may result in delay in response. The incoming

request might wait into a request queue. The usage of processor time of the DNA application is less compared to NODNA web application. It increases performance of the website by reducing processor time and increasing response time. The performance test shows clearly that DNA website provides high performance with low CPU usage. It helps to handle more requests with less processing power because of caching various business layer and data layer objects into memory.

5 Conclusions and Future Work

As a result of growing interest and need for more powerful enterprise solutions, vast amount of research is undertaken in this field. This field depends on major research areas such as networking, programming language improvements, hardware and software technology changes, and database speed improvements. The improvement of any area may provide significant change in overall performance of the application. The DNA is not restricted with any programming language. It helps to develop Internet based applications by choosing any programming language. This provides benefits of using latest programming techniques and language such as Microsoft .Net framework, Microsoft asp, Sun jsp, php etc to develop a web application.

The test result of a real time application is also important during comparison between test results. The real time configuration and real time application output indicates exact performance capability of the web servers. The real time application response is an important factor to achieve better performance result.

References

1 Joseph M., *Enterprise Application Architecture With VB, ASP and MTS*, Wrox Press Ltd. 1999,

2 Colajanni M., Yu P., Diad D., "Analysis of task assignment policies in scalable distributed wed-server systems", *IEEE Trans. Parallel And Distributed systems,* 9(6):585-600, June 1998

3 Ezhilchelvan, Palmer D., Khayyambashi R., Morgan G. "Measuring the Cost of Scalability and Reliability for Internet-based, server-centered applications", 6th *Intl. Workshop on Object-oriented Real-time Dependable Systems (WORDS01)*, Rome, '01.

4 Petra B., Tom F., Ernst W. "Parallel Continuos RandomizedLoad Balancing". *Proc. of the 10th Annual ACP Symposium on Parallel Algorithms and Arch.*", pp. 192–201, '98.

5 Thomas L., Jon G. "Analysis of Three Dynamic Distributed Load-Balancing Strategies with Varying Global Info. Requirements". *IEEE Computer*, pp 185–192, August 87.

6 Xu C. and Lau F., *Load Balancing in Parallel Computers,* Kluwer AcademicPublishers, Boston,MA, 1997.

7 Andresen D.,Yang T., Ibarra O. H., Smith T. R., "Scalability issues for high performance digital libraries on the world wide web", *Proceedings of IEEE ADL, Forum on research and technology advances in digital libraries*, Washingoton, 1996.

8 Clements, Paul, "From Subroutines to Subsystems: Component-Based Software Development", *The American Programmer*, Vol 8, No 11, Nov 1995

9 Clements P. E., "Requirements for software systems which enable the flexible enterprise", 1997, *MSI Research Institute*.

10 "Microsoft COM Technologies" http://www.microsoft.com/com/