Security Engineering and Reengineering on Windows 2008 Server Based Distributed Systems

Cosmin TOMOZEI

Department of Computer Science, University of Bacău
Bacău, ROMANIA
cosmin.tomozei@ub.ro

Abstract: The objective of this paper is to reflect on the processes of Security Engineering and Reengineering in Distributed Systems, focused being also about the Windows 2008 Servers. It is very important to provide security and integrity to software applications, hardware and data. We will have in consideration the ways of making reengineering process efficient, including optimization of the encryption and authentication stages.

Keywords: Security Engineering; Reengineering; Distributed Systems; Data Encryption; Authentication.

1. Preliminary aspects regarding distributed systems

Software Engineering and Reengineering are considered to be actual and important in the industry, an increased number of specialists being involved in researching about the theme. The results of research are both theoretical and methodological. Practical work from software companies, research laboratories and universities is also not to be ignored and provides valuable scientific results.

Our research is focused mainly in describing the way in which we reengineer distributed systems, consisting of hardware, software and communication resources in order to obtain a higher level of efficiency.

Due to the higher and higher level of complexity of software systems and applications, it is practically impossible to start the development process from green field over and over again. Fast changing of the objectives which have to be accomplished by computer programs and IT systems in general presume that *maintainability* must become the most significant metric. In the same time, when talking about dimensions, volume of operations, or necessity of computing large volumes of information, the *distribution* of software, hardware and data come as a logical consequence. Maintaining an adequate level of integrity and security involve distribution as well, hence being subjected to reengineering.

In [1] there are presented some aspects regarding distributed systems that we mention in the following:

- *multiple nodes*, connected in a computer network; distributed systems suppose that multiple computers are connected and share tasks in order to realize the objectives; parallel processing is not to be confused with distributed computing; while parallel processing involves many processors on the same machine, distribution means at the outset task sharing;
- *concurrency*; each nod has independent functionalities apart from the other nodes and operates concurrently with the other nodes; it is likely to exist many processes on each node, and on each process there are multiple threads;
- *message passing* through protocols, such as TCP/IP over modems or Ethernet;
- *heterogeneity* of nodes; each node being dissimilar regarding hardware and software;
- *multiple protocols*, mainly asynchronous;
- *openness;* in comparison to sequential programs, which are mainly closed and do not change their configuration during execution, in distributed systems we can add nodes while the whole ensemble is functioning;

This is a post conference paper. Parts of this paper have been published in the Proceedings of the SECITC 2009 Conference (printed version).

ISSN 2067 – 4074

the openness presumes that each node satisfies a set of conditions and protocols to ensure *interoperability* with the components added or modified;

- *fault tolerance and transparency* allow users to cooperate with the software system without knowing if there are components that don't work in a certain moment of time; this fact should not affect the general functioning of the system;
- *persistence;* data is stored in a persistent, non volatile environment, such as databases, data warehouses and storage servers;
- *security;* each user should interact with the system according to the rights he has; elements such as *authentication servers*, *firewalls*, *antivirus* technologies are to be used;
- there is *no central server,* but there are multiple servers which cooperate in order to achieve nonstop, uninterrupted operation of the system;

Distributed applications consist of the software components that run on the distributed systems, and have the following characteristics:

- they are built in distinct development environments, they operate in diverse environments, on different operating systems, on platforms that are connected in a computer network;
- they are built on two tiers (client - server), three tiers (client - middleware - server) or multitier (client - multiple middleware - multiple servers);

In [2] we made the following remarks regarding distributed applications as well.

2. Software engineering and reengineering premises

If software engineering is defined by the amount of techniques, methods and procedures to create and maintain software applications, software reengineering will presume the defining of a new objective which have to be realized in an amount of time, but does not diverge significantly from the old one. Still, the new objective has to prove that an increase regarding quality is being achieved.

It is essential as well to have an existing software application which will be subjected to reengineering. Elements regarding the connection of the modules [3], [4] that remain unchanged are to be found, in order to integrate the new modules and the entire application to become a powerfully connected system. There are parts that remain in the structure and parts to be modified or deleted. In the next formula, this is formalized:

$$Ir = \bigcup_{i=1}^{n} Funct_{i_o} + \bigcup_{i=1}^{n} Funct_{i_1} - \bigcup_{i=1}^{n} DelFunct_i \quad (1)$$

where:

- Ir represents the indicator of reengineering;
- $Funct_{i_o}$ is the functionality in the initial moment;
- $Funct_{i_1}$ is the functionality in the present moment added due to reengineering;
- $DelFunct_i$ is the functionality eliminated by reengineering;
- n represents the number of functionalities;

The choice of the right reengineering strategy is another important premise for having a good and efficient result. A good strategy involves well defined objectives, good methodology, reliable software development platforms and an appropriate working team.

On each stage of development, it is compulsory to measure the level of achievement of each objective. If the preliminary results obtained in the reengineering process are differing significantly from the predicted results, modifications and updates should be made.

The two concepts regarding reengineering, *necessity* and *opportunity,* clarify in a

scientific manner the premises of software reengineering.

The necessity concept is defined regarding the development team or company, which is entitled to create a new software project for a beneficiary company. The necessity comes as a consequence, due to the need of realizing the project in a reasonable period of time and with reasonable costs. As a matter of fact, it would be compulsory to *reuse* the existing amount of libraries, procedures, architectures, diagrams and previous experience. Consequently the results are to be more efficient.

When talking about reengineering opportunity, we have into consideration a *mirror image* of the *necessity* concept, this time from the beneficiary company perspective. We must say that the opportunity of reengineering comes as a result of the need of a company to adopt the software they already have and use from a long period of time, to the new conditions of the market. The existing software cannot deal anymore with the present business environment due to the following premises:

- the fast evolution of the demand on the markets; the appearing of new demands for new goods or services, which have to be offer by the company;
- the growing diversity of goods or services that the company is offering on the market, or just producing;
- the business orientation to the electronic business environment, consequently to the proportion that the IT&C platforms grow and become more complex in this environment;
- the growing speed of delivery the ordered goods or services;
- the need of integrating the existing hardware, middleware and software platforms in the new business strategies.

The above mention premises allow us to state that in these cases the opportunity of software evolution and reengineering comes to light. The concepts of necessity and opportunity place the reengineering concept in relation to the both sides, the development company and the beneficiary.

Reengineering necessity and opportunity concepts are being defined according to the software evolution laws. The first law is *the law of continuous change* [5] and *law of increasing complexity* [5].

The statement of the law of continuous change is:

Software which is used in the real world must change and evolve, in a contrary case becoming less and less useful in the environment in which is used.

The statement of the law of increasing complexity is:

Once software is evolving, it tends to become more and more complex and needs additional resources for the reduction of complexity and preservation of architecture.

Software evolution [6] confirms that life of particular software does not stop once a new demand has appeared. The software entity has to be maintained in order to fulfil the new demand and keep the same level of quality in results.

3. Distributed applications reengineering representations

A legacy distributed application is a software application inherited from the previous stages of business in the company. Along with the years passed, the software application proved its value, along with the development of business. Consequently, on each stage of business environment evolution, new functionalities have been added, consisting usually in classes, methods or relational tables. All of these updates and modification had negative impacts on complexity. Furthermore, any other future modification or update will just complicate the structure of the application from both, architectural and code points of view.

Reducing complexity of distributed application is a matter of *robustness and security. S*ecured distributed applications need clear architectures, distinct and robust authentication and protection systems.

When talking about distributed application created in Object Oriented Environments, it is also compulsory to think about initial examinations of correspondences between class diagrams and the actually implemented classes and also between the

ISSN 2067 – 4074

sequence diagrams and the way classes interact each other.
It is emblematic for Object Oriented distributed applications to have incorrect relations between modules, high degrees of redundancy and violation of encapsulation. In C++, C# or Java application one of the most common ways of violating encapsulation is by placing public labels instead of private and use of friend methods. In figure 1 there is described the functional architecture of reengineering process.

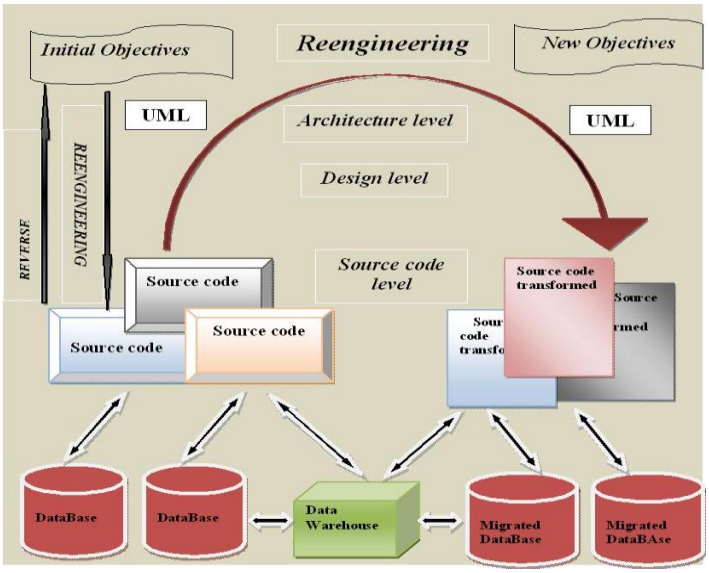

Fig.1 Functional architecture of the reengineering process

The functional architecture of this process points out the elements which are integrated in the activity of software reengineering. *Reverse engineering* and *reengineering* are constantly being mentioned simultaneously and sometimes it may appear some confusion between them.
Reverse engineering consists of the process in which a subject is analyzed and its components and relations between components are identified. The result of reverse engineering is just a representation, in which the reality is described in a new way, often more abstract. Reverse engineering tries to offer an image about how the system is working.
In comparison with reverse engineering, reengineering is the process, in which the software system is studied, developed and transformed in order to rebuild and implement it. Reengineering restructures the software system at all its levels, in order to repair the broken things and fulfil the new objectives determined by the beneficiary and the development team.
Forward engineering presumes going from the levels of architecture to the physical layers of a system. There are continuous debates about how the process of forward engineering should be developed, but the grand majority of specialists agree that it should be iterative and in conformity to the Barry Boehm's Spiral Model.
If forward engineering presumes going from the abstract models and architecture, to the concrete implementation of classes, packages or modules, reverse engineering is just the opposite, the way from concrete, implemented and existing software packages, classes or modules to the architectural level. We state, according to [5] that software reengineering inherits and integrates both forward and reverse engineering.
In addition to the model described in [5], we also declare as compulsory to integrate the database Tier in the functional architecture of reengineering. The reengineering process does not imply just transforming source code, but transforming a software system at all its

levels or tiers, including the database tier as one of them.
Transformations are made in the database tier as in all the other tiers, data migration being important and common when changing the software versions or architectures. The transformations are made as in the case of classes, packages or modules, transforming relational tables, studying and updating the relationships between them. Data from the warehouses are particularly less often transformed, because they are de-normalized and historical.
As the process of reengineering is progressing, more refined models and structures are to be obtained, based on the answers of the initial questions, and a more clearly technical documentation is elaborated.
The reasons for reengineering a software application may be distinctive, in [5] there are some common opinions for that:

- insufficient documentation; inexistent or not reflecting reality;
- inadequate distribution of code, classes, methods and packages which generates problems of maintenance and portability;
- lack of modularity; strong connection between modules implies a hard to obtain evolution of software;
- duplicated source code, which generates problems in maintenance;
- redundancy of functionalities;

The development team has to transform the whole system. They have to create and run tests for every component that have to modified, transformed or replaced. The most important thing is to evaluate the entire system and too look for a clear general image, about what has to be transformed.

4. Windows 2008 Server nodes security reengineering

In this section, we will focus on reengineering the roles and security policies of Windows 2008 servers in distributed systems. In [6], we have some aspects regarding virtualization, role based security, Active Directory and data encryption. It is also mentioned that Windows 2008 Server is the most secure server ever produced, which integrates Network Access Protection, Federated Rights Management and Read - Only Domain Controller. Data theft is prevented also, by the help of Rights Management Service and BitLocker [6].
Network Access Protocol (NAP) is a technology which prevents non compliant computers from accessing and compromising an organization network. It is compulsory for the client computers to pass some tests regarding their health and compliance, before accessing the organizational network. NAP helps them to maintain and improve their health and compliance. Companies which do not use Microsoft technologies are able to cooperate with the organization network which runs NAP technology by the NAP APIs.
Many organizations implement protocols regarding the offering of access to their networks from the exterior. For example [6] a client has to have antivirus technologies installed and the updates must be done regularly. As well, firewall technologies have to be used in order to connect the clients to the organization networks. Health policies are very important and prevent the company of being subjected to threats, attacks or theft of information.
Often, health policies must be updated and from time to time included in the reengineering process. Network administrators have to create and transform health policies in order to keep secure the organization distributed system. If we take into consideration the company's computer network and the software that is installed on the computers, as well as the communication technologies implemented, we may think the entire system as a distributed one.
Health policies prevent viruses, worms, Trojans and phishing adverts to spread in the computer network and put the business process under unwanted risk. NAP provides as well the possibility of updating automatically the client computer software with the necessary amount of data in order to pass the health security

ISSN 2067 – 4074

test. The NAP architecture is described in Figure 2.

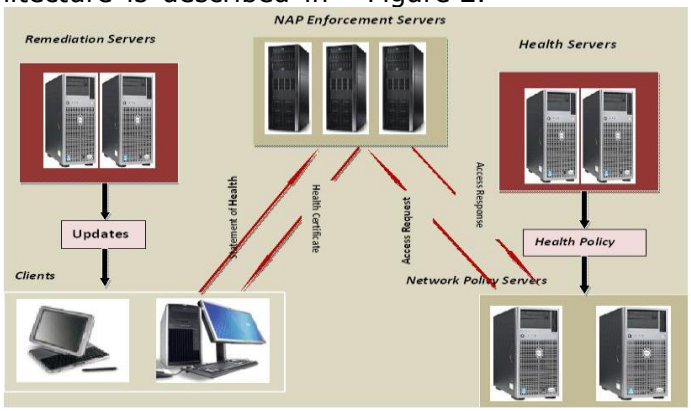

Fig. 2 Architecture of Network Access Protocol [6]

In the up - mentioned architecture, we can distinguish the functional elements of the Network Access Protocol. Remediation Servers offer updates to the clients. The clients send messages to the Enforcement Servers. The Enforcement servers communicate with the network policy servers, which provide access to the clients, after auditing the health certificates. Each component must be aware of the necessary updates and from time to time must be reengineered.

BitLocker Drive Encryption is a security feature in Windows Server 2008 that helps protecting servers which are placed in remote locations. This technology is also available for mobile users and other categories of operating systems, besides the server ones. The entire content of a disk drive is encrypted, preventing other users with dissimilar operating systems from breaking the protections and accessing the information stored on the hard drive.

Authentication and management of user identities [7] are guaranteed due to IDA (Identity and Access) which helps organizations to manage user's identities and associate privileges to them.

Reengineering may also bring new ways in which identities are associated with privileges and rights, improving the authentication process. The following lines reflect how the authentication process works under our distributed system, where usernames and passwords are store in relational databases. The connection strings needed for the database access are stored in .config files which cannot be accessed from the outside, due to the server restrictions. It is possible to store unencrypted data as well, such as usernames and passwords, or data regarding privileges, but it is advisable to encrypt it. Forms authentication offers the possibility to encrypt data in order to prevent the information theft, which would cause serious problems to the organization.

```
protected void Login2_Authenticate(object sender, AuthenticateEventArgs e)
  {
     SqlConnection sq1 = new SqlConnection();
     sq1.ConnectionString                                                    =
    ConfigurationManager.ConnectionStrings["ConnectionString1"].ConnectionString;
     SqlDataAdapter ad1 = new SqlDataAdapter();
     DataSet ds = new DataSet();
     SqlCommand sqli1 = new SqlCommand("select id_utiliz,nume,prenume,parola,categorie from
    utiliz2", sq1);
     ad1.SelectCommand = sqli1;
     ds.Tables.Add("utiliz2");
     sq1.Open();
     ad1.Fill(ds, "utiliz2");
     foreach (DataRow dr in ds.Tables["utiliz2"].Rows)
```

```
        if (((string)dr["nume"] == Login2.UserName.ToString()) && ((string)dr["parola"] ==
    Login2.Password.ToString()))
        {
            Session["idutilizator2"] = Convert.ToInt32(dr["id_utiliz"]);
            Session["nume2"] = Login2.UserName.ToUpper().TrimEnd();
            Session["prenume2"] = dr["prenume"].ToString().ToUpper().TrimEnd();
            Session["categorie2"]= dr["categorie"];
Response.Redirect("~/pagina_personala/pagina_personala2.aspx");
        }
        else
        {
            Login2.FailureText = "Nume sau parola gresite \n";
            Login2.FailureText += "Utilizati doar numele de familie";
        }
     sq1.Close();
  }
```

Fig.3 Authentication process, based on C# and ADO.NET

The management of user identities [6] is a high priority for many businesses and organizations nowadays. Employees or just people need to access the company's multiple systems and resources on the corporate networks are using different types of devices, which are based on multiple operating systems. Because many of these systems cannot communicate with each other, it's not uncommon to have multiple identities for the same person. As a result, managing these types of redundant identities is very complex, needs a lot of time resources and increases security risks due to errors and poorly constructed user password management.

```
protected void Button_Parola_Click(object sender, EventArgs e)
  {
     SqlConnection sq1 = new
    SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString1"].ConnectionString);
     SqlDataSource sds1 = new SqlDataSource();
     SqlParameter p1 = new SqlParameter("@parola", SqlDbType.VarChar,20);
     SqlCommand sqc1 = new SqlCommand("update utiliz set parola=@parola where
    id_utiliz=@id_utiliz",sq1);
     p1.Value = textnouaparola.Text.TrimEnd();
     p1.Direction = ParameterDirection.Input;
     sqc1.Parameters.Add(p1);
     SqlParameter p2 = new SqlParameter("@id_utiliz",SqlDbType.Int);
     p2.Value = Session ["idutilizator"];
     p2.Direction = ParameterDirection.Input;
     sqc1.Parameters.Add(p2);
     try
     {
        if (String.Compare(textnouaparola.Text,textreintroduceti.Text)==0)
        {
           sq1.Open();
           sqc1.ExecuteNonQuery();
           labelmesajmod.Text = "The password has been modified!";
        }
        else { labelmesajmod.Text ="You have entered the wrong password";
            }
     }
     catch (Exception ex)
```

ISSN 2067 – 4074

```
    {
        Response.Write(ex.Message);
    }
}
```

Fig.4 Password update process in distributed systems, based on C# and ADO.ET

Some considerations are made as well in [6] about the advantages brought by Windows 2008 Server by the following components:

- Directory Services
  - Read-Only Domain Controller (RODC)
  - Active Directory Federation Services (AD FS)
  - Directory Service Auditing
  - Service-based Active Directory Domain Services (AD DS)
- Information Protection
  - Federated collaboration
  - BitLocker
  - Federated Rights Management
- Strong Authentication
  - Cryptography API
  - V3 certificate templates
  - Public Key Infrastructure(PKI)

5. Conclusions

The paper presented some valid considerations regarding software engineering and reengineering in distributed systems, where Windows 2008 Server nodes are being in use. Software reengineering techniques are very important also regarding the security aspects. Considerations regarding the characteristics of distributed systems and distributed applications have been also made, in order to create an appropriate context for our discussion. The formal indicator of reengineering has been identified, based on distributed application development process. Graphic representations of the processes and activities presumed by the reengineering process were also described, with the purpose of underlining the functional architecture of reengineering.

In the last section, we mentioned aspects regarding Windows 2008 server security protocols, such as Network Access Protection, Federated Rights Management, Read - Only Domain Controller Rights Management Service and Bit Locker. Consequently, an example of authentication software application and update of passwords have also been described.

References

[1] E. D. REILLY – *Concise Encyclopedia of Computer Science,* Wiley, 2008, ISBN 0470090952

[2] Cosmin Tomozei – *N-Tier Distributed Applications Dependable Construction*, Journal of Information Technology & Communication Security, SECITC, November 2008, pag.65-71, ISBN 978-606-505-137-9

[3] Ion IVAN, Marius POPA, Cosmin Tomozei- *Reingineria entităţilor text*, Revista Română de Informatică şi Automatică, vol.15 nr.II 2005 ISSN 1220-1758 pag.15 – 28

[4] Marius VETRICI, Cosmin TOMOZEI - *Distributed Collaborative Software Development Process Improvement Using Criticality Analysis,* Economy Informatics, Vol 9/No. 1/2009,pag. 71-78, ISSN 1582-7941

[5] Serge DEMEYER, Stephan DUCASSE, Oscar NIERSTRASZ - *Object – Oriented Reengineering Patterns,* Square Bracket Associates, 2008, ISBN**:** 978-3-9523341-2-6

[6] Windows 2008 Server Technical Report, www.microsoft.com

[7] Windows 2008 Server Reviewer's Guide, www.microsoft.com