TESTING AND SECURITY IN DISTRIBUTED ECONOMETRIC APPLICATIONS REENGINEERING VIA SOFTWARE EVOLUTION

Cosmin TOMOZEI[1]
Assistant-Lecturer, PhD – C.
"Vasile Alecsandri" University of Bacău, Romania
Department of Mathematics and Computer Science

E-mail: cosmin.tomozei@ub.ro

Bogdan PATRUT[2]
Associate Professor, PhD
"Vasile Alecsandri" University of Bacău, Romania
Department of Mathematics and Computer Science

E-mail: bogdan@edusoft.ro

Abstract: *The objective of this paper is to present the results gathered in research, regarding the testing and the security assurance as key components of distributed econometric applications reengineering. Consequently, the testing and security procedures will be formally shown in order to bring into light the robustness and stability of econometric applications reengineering. The* ***W*** *model for software testing in reengineering will be as well exposed as one of the contributions. Agility and iterative features of software reengineering are to be mentioned, with the intention of revealing their use during the lifecycle.*

Keywords: *security reengineering; W testing model for reengineering; distributed applications.*

1. Distributed Applications Requirements and Dynamics through Reengineering

Dynamics, as an important characteristic of distributed applications is brought into light by the software evolution. Distributed applications are constantly being subjected to several types of transformations. They consist of the updates of their structures, such as the addition of new modules or modifications in other modules, so as the software applications does conform to the new objectives they have to achieve. The evolution in objectives has a perpetual correspondent in the evolution of requirements.

Evolution of software is viewed in literature as the other the types of evolution, such as human evolution, social evolution or economic development. The desiderata of stability and concordance have to exist between changes in business processes and changes in software applications, done by means of software maintenance or by reengineering.

The concept of evolution is defined in *Encyclopaedia Britannica*, from a biological perspective. Then customizations are made to present this concept in other ways and for other sciences. In our point of view, software evolution is considered to be the starting point in the analysis of reengineering, particularly when talking about distributed software or about econometric applications.

Definition 1[1]: *Evolution is the biological theory that plants and animals have their origins in pre-existing species and the differences that distinguish between them are due to changes that took place in successive generations.*

In [3] evolution is defined as a creative concept, an alternative to Charles Darwin's, by Henri Bergson in the work, Creative Evolution, published in 1907. According to this theory, the evolution is achieved due to the natural creative impulse and human motivation.

Definition 2 [2]: *Software evolution is the sub domain of the software engineering discipline that investigates ways to adapt software to the ever-changing user requirements and operating environment.*

The spiral software development model with agile elements that we used in our reengineering projects is observed and it extracts the trends of future iterations. In this way, reengineering and maintenance become predictive processes, more easily to be understood and continued by development teams.

Thus, certain aspects are provided in advance. Therefore, the activity of creating new software becomes an evolutionary process through successive stages of the projects evolution. It becomes straightforward then, to determine correlations between the social and economic developments and their coverage in distributed software applications.

Definition 3: *The evolution of software development is a scientific and technical set of creative phenomena. Creativity and willingness to develop new applications give impulses to specialists to create significant and valuable contributions in the development of new software systems, corresponding to the rapid and continuous changes in the real world.*

Regarding the dynamics of distributed systems and consequently of distributed applications, their evolutionary trend should be noted, while systems configuration changes over time. The successive stages of maintenance and reengineering transform the software entity, in such way that after a certain number of iterations, software applications are becoming very different in comparison to their initial configurations.

The spiral development model with agile elements shows that the integration elements of agility in application development cycle, software configurations are becoming more dynamic, so that the updates in real-world business processes are to be reflected in the structure and objectives of distributed applications in a shorter period of time.

We believe that the software development cycle should be defined correspondingly to the reengineering cycle, so that each stage of the development cycle should have a counterpart in the implementation of the reengineering process. This will offer to software applications higher levels of maintainability and adaptability in the future.

Distributed applications reengineering is an evolutionary and adaptive approach, which incorporates elements from existing IT systems and applications that proved to be valuable for the organization over time. These elements will then be integrated and exploited in the new stage of the system or distributed application, by transformation.

In [4] the text entities reengineering is taken into discussion. By analogy, we present the concept distributed applications reengineering with the involvement of the following aspects:

- the definition of a new objective that has to be achieved, which does not diverge significantly from the initial aim of the distributed application, but reflects a qualitative leap in the level of the outputs;

- the existence of a distributed applications that is to be subject to the transformation process; consequently, the components which remain in the structure and the new modules that should be introduced in the structure should be determined, so that the results should identify themselves within a well-coagulated software; some classes and objects remain in the structure, others being eliminated and others being modified in order to achieve new goals; furthermore, we formalize this by the reengineering function [5] which is defined on the basis of the development function relationship (4) by updating the modular structure of applications which is subjected to reengineering;
- the choice of appropriate technologies so as to reengineer the entire software system; it consists of the architecture, diagrams, classes of objects, databases, relational tables, all included in the new structure of software;
- the establishing of practical quantitative methods and indicators for a precise measuring the quality of each iteration of the process such as engineering, reverse engineering and reengineering; afterwards the software system reaches its final form by being iteratively transformed;
- the accurate description of the evaluation stages of the reengineering life cycle for the efficient management of the budget, people involved in the project and time resources;
- the management of risks related to software reengineering process; this stage is important for the security and integrity of the reengineering projects and contributes to the quality assessment of the reengineering process;
- the team composition**,** each team member will have specific tasks to be completed within the project; each member of the reengineering team becomes familiar with the applications state-of-art and current technologies.

Distributed applications reengineering is a complex process that presumes transformations and updates to all the levels of software such as client, business logic and database tier. The projects of reengineering assume higher costs and long development durations. Still, reengineering saves many financial resources and besides that, it reduces development costs and keeps the valuable modules of software for a longer period of time.

In distributed applications reengineering, the most common cases are made of migrations to new versions. The needs of such kind of projects are due to the following premises:

- the emergence of new fields of activity, which are to be reflected in the software application, such as economic growth, legislative or tax changes or structural changes;
- the appearance of new categories of goods and services, that are needed to be offered to the customers; depending on the operating activities of the company, it will comply with consumer demands by offering new products involving a high degree of innovation and quality;
- the materialization of new ways to deduct business expenses, due to legal developments and changes in financial accounting;
- new regulations relating to fixed capital depreciation, which update the categories of goods that are repaid, duration and the methods of calculation;
- the need to meet customers' demand in a shorter time and with greater suitability; increasing speed of response to the demand appeared on the market is essential to ensure competitiveness;

- organizational and structural changes are compulsory to be reflected in the computer system of the company;

The key element in software reengineering takes for granted the existence of the original software, which has proved its value in use over time. Due to numerous updates and phases of maintenance, there is a sufficient likelihood that soon the software will no longer achieve the necessary tasks, or his errors will exceed the established level of significance.

Unlike software development process that starts at the green field, with the original vision of the development team, interviewing beneficiaries, determining objectives, determining the preliminary architecture of the system by the time he entered into service, the reengineering project assumes as a starting point the analysis of the existing system in the organization and detection of the problems it faces.

As a result, the reengineering team will decide which of the parts of the initial structure are going to be kept in the new application and which of them are going to be updated or deleted. The key point of reengineering is to reuse the valuable components or modules of the application.

Another key issue in the process of software reengineering consists of identifying the optimal timing of application, based on formula (1). For each of the iterations in the development cycle a time resource *Ti* and a level of total cost CT_i are allotted. The inequalities given in the relations (1) and (2) present these desiderata. But for the maintenance process to be efficient, the cost and duration for the following iterations of adaptive maintenance have to be less or equal to the time and cost of the previous iterations.

$$T_1>=T_2>=\ldots T_i\ldots>=T_{n-1}>=T_n \qquad (1)$$

$$CT_1>=CT_2>=\ldots CT_i\ldots>=CT_{n-1}>=CT_n \qquad (2)$$

Each one of the iterations within the spiral reengineering model with agile elements reflects on the achievement of a new objective. The new objective is reflected in the requirements identified by the software development process management, due to the interactions with the target group of the IT application. This is shown also in the algorithm in Figure 1.

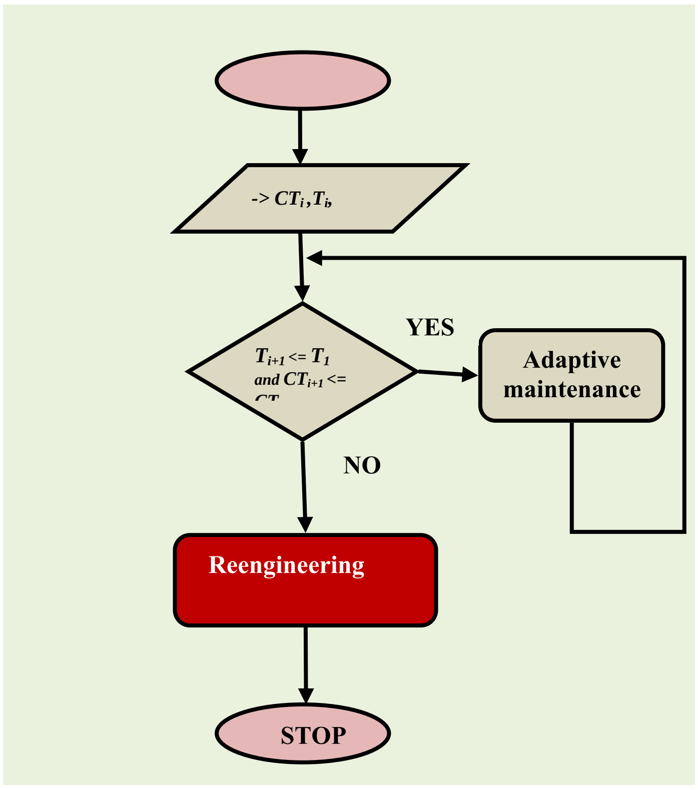

Figure 1. The choice between adaptive maintenance and software reengineering

In figure 1 it is presented how the management of the IT project decides between the continuation of the process of adaptive maintenance for the distributed application and the starting of a new project, which consists of software reengineering.

The idea of reengineering is brought into light because of the growing durations and costs of each adaptive maintenance process, correspondingly to the new objectives appeared in user requirements. Therefore, after several stages of adaptive maintenance, the complexity of the software application [13] is significantly growing and each of the new users' requirements engages additional costs and more and more time resources.

2. Software components reuse in the process of reengineering

The reuse of components constitutes the core element of the reengineering process. The components already existing in the system have to coexist with new elements added to the structure of the software entity.

The reuse of components in reengineering is carried out mostly by the upgrade of their existing structure and behaviour. It is a lot more productive to update the components through the process of software refactoring [10], in comparison to completely abandoning them and then to start the development process from the green field.

In software developers' communities, it is often understood by reuse of components the meaning of source code reuse. This point of view is partially correct, but in a simplistic manner, because distributed applications presume the existence of many other components, with the same importance and need for reuse in reengineering. The reuse of components consequently implies transformations and updates to the level of requirements, to the architectural level, to the level of detailed design, to the level of unit testing and the database tier, as well as in the source code.

The refactoring process takes into account the existing methods in the classes of objects to give them a high level of quality, by identifying and removing duplicate sequences of code, so - called software clones. Refactoring has the role to provide a higher degree modularity and reusability in the following iterations of software reengineering.

When talking about the life cycle distributed applications reengineering, it is necessary to detect code duplication and subject the software entity to refactoring, because even from earlier stages of the adaptive or corrective maintenance, redundant elements of various types, such as classes of objects, methods, tables, or attributes, were added by method of copy / paste in order to meet the new requirements in a relatively short period of time. That aspect does greatly increase the complexity and the redundancy of software.

The reuse of components is based on the study and preliminary understanding of the basic elements existing in the application being reengineered. Each one of the data structures has to be identified and it should be determined how each data structure contributes to the objectives of the application. Just after that, decisions about the update or elimination of data structures are to be made. However, these decisions also take into account that all the work done by specialists is based on accurate requirements.

The duplication of code and of the databases structures in distributed applications condition the growing of redundancy. This phenomenon is identified and minimized through reengineering. Most of the reengineering process analysed in the community of specialists [6] reported that duplication of code has been detected from 7% to 23%, which in extreme cases went up to 59%. This is the reason why significant efforts have to be made for the detection and the minimization of the level of code duplication in distributed applications reengineering projects.

The process of code compaction is ensured by source code refactoring. Having a compact and stable code is important because this is minimizing the volume of operations performed and reduces the number of calls of the default method as well. This process has great importance in simplifying the class graph of the distributed application. Hence a much clearer picture of the class type entities and interactions between them is developed.

Because changes made in the code by refactoring process are reflected in the architecture and system design, and vice versa, it is desirable to always maintain the link between architecture and actual implementation.

The indicator of code duplication reflects the number of duplicated source code entities divided by the total number of entities from the source code and it is defined as:

$$Idc = \frac{\sum_{i=1}^{n} ClassDup_i}{\sum_{i=1}^{k} Class_i} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} MetDup_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{m} AttrDup_{ij}}{\sum_{i=1}^{k}\sum_{j=1}^{p} Met_{ij} + \sum_{i=1}^{k}\sum_{j=1}^{p} Attr_{ij}} \qquad (3)$$

where:

- I_{dc} is the indicator of code duplication
- $ClassDup_i$ represents classes are duplicated in the distributed computing application;
- $Class_i$ represent classes of objects within the distributed computing application

- $MetDup_{ij}$ is the duplicated method *j* from the class *i;*
- $AttrDup_{ij}$ is the duplicated attribute *j* from the class *i;*
- *n* is the number of duplicated classes of objects in distributed application;
- *m* is the number of class methods and attributes of objects duplicated;
- *k* represents the total number of classes in the distributed application;
- *t* is the number of duplicate methods within the classes.

This indicator has been taken into account in the process of reengineering the two-stage least squares distributed econometric application for regression analysis. In consequence, the number of code lines has been reduced and several methods were correctly parameterized by refactoring. However, in econometric applications development the correctness of econometric algorithms comes on the first place, and just after the obtaining of correct coefficients of linear models developers have to pass through the source code optimization by refactoring.

The transition of the econometric application from the initial phase to a distributed environment is described in (4), concerned being on the evolution in objectives and modules.

$$Dev\left(\bigcup_{i=1}^{n} Mo_{i1}\right) = \sum_{i=1}^{n} Obj_1^i = Obj^1 \xrightarrow{[\mathrm{Re}\,engineering]} Dev\left(\bigcup_{i=1}^{n} Mo_{i1} + \bigcup_{i=1}^{m} Mo_{i2}\right) = \sum_{i=1}^{n+m} Obj^i{}_2 = Obj^2 \quad (4)$$

where:

- Mo_{i1} are the distributed application modules after the first iteration of process reengineering
- *Dev* is the development function of the spiral model with agile elements;
- *n* is the number of modules of the application after the first iteration of reengineering;
- *m* is the number of application modules, introduced during the second iteration of the reengineering process;
- Mo_{i2} application modules are distributed result after the second iteration of the reengineering process;
- $Obj^i{}_2$ targets are operational, distributed as a result of the application process reengineering software that are integrated into the ultimate objective of the application;
- Obj^2 is the ultimate aim of application in stage 2 as an arithmetic sum of other objectives.

Quality assurance strategies of distributed applications are very important reengineering. Each one of the updates are being inspected and certified through testing. In the next section, accordingly, the W testing model for reengineering will be defined and described.

3. The W Testing Model for Reengineering

If the process of reengineering starts in a mostly theoretical way, by the determination of requirements and demands, suddenly aspects regarding the study and testing of the existing software application are brought into the play. Initial predictive test are made, for the team to efficiently understand the behaviour and the structure of the application. In addition to this, behavioural and structural tests are also being defined by the reengineering team.

The initial tests are carried out by the team of specialists, while analyzing the existing information system or software application for the determination of the systems behaviour based on the established factors from requirements.

It is also necessary that in the steps of reengineering discussions and interviews take place between the reengineering team and the target group of users. The ideas and objectives determined through the interviews and the preliminary discussions are the basis of the black box testing phase.

Black-box testing consists of a collection of predictive methods, techniques and procedures used in the analysis of existing software, which is to be subjected to the reengineering process in order to determine the behaviour and utilization patterns. These tests provide an objective view on the IT application.

In [7] there is mentioned that black-box tests identify the applications behaviour, and black – box tests are used in order to determine bugs in the high-level operations as well as the overall functionality of the tested software.

Testing the behaviour of the software application requires an understanding of the domain, and how software contributes and models the activities within the organization. The higher level of design is understood by the team of IT project and black-box test cases are identified. Every test case is placed in correspondence with the activities of the target group of users.

In the case of distributed econometric applications for two-stage least squares regression analysis, it is shown that the development team as well as the teams of maintenance and reengineering must have solid knowledge of statistics and econometric methods and the way to implement them in programming languages. Later in the project, aspects about how to integrate the modules in distributed applications are to be carried out, by means of reengineering.

For distributed econometric applications discussions and interviews are carried out as in the general case of software reengineering. There are group discussions and interviews with the users in order to determine the essential elements of their interaction with the software. These include the following tasks to be achieved by the reengineering team:

- the interviewing of users about the types of problems they solve with the help of the existing econometric application; they describe the elements of statistics and econometrics consisting of the defining of problems which have to be solved and the way in which the econometric problems are integrated into the existing application;
- to get the users description of the types of data sets, here the test team checks the general types of data sets, the maximum number of data series, the maximum number of exogenous variables included in the regression model, the validation of the data sets;
- the way in which the target group of user are dealing with the functionalities of the existing application; the reengineering team follows the human – application interactions, and the way in which each member of the target group reaches his econometric or statistical objectives;
- what the users opinions about the application which is subjected to the analysis are ; if they are pleased regarding the duration of the operations, and how the users perceive the ways of improving of the existing application;

- the unusual aspects determined in the applications history; the target group of users tell to the reengineering team about the problems had in the past, in the use of the software application, and also about the ways and durations of solving of the identified problems;
- get the users opinions about the quality of the application; the target group is formed of specialists in statistics and econometrics, and they have the knowledge to share with the development team, regarding which other statistical or econometric algorithms are needed to be implemented in the application;

In figure 2 there are presented the stages of the distributed econometric application testing during the process of reengineering, integrated in the *W* testing model based on the V model presented in [7].

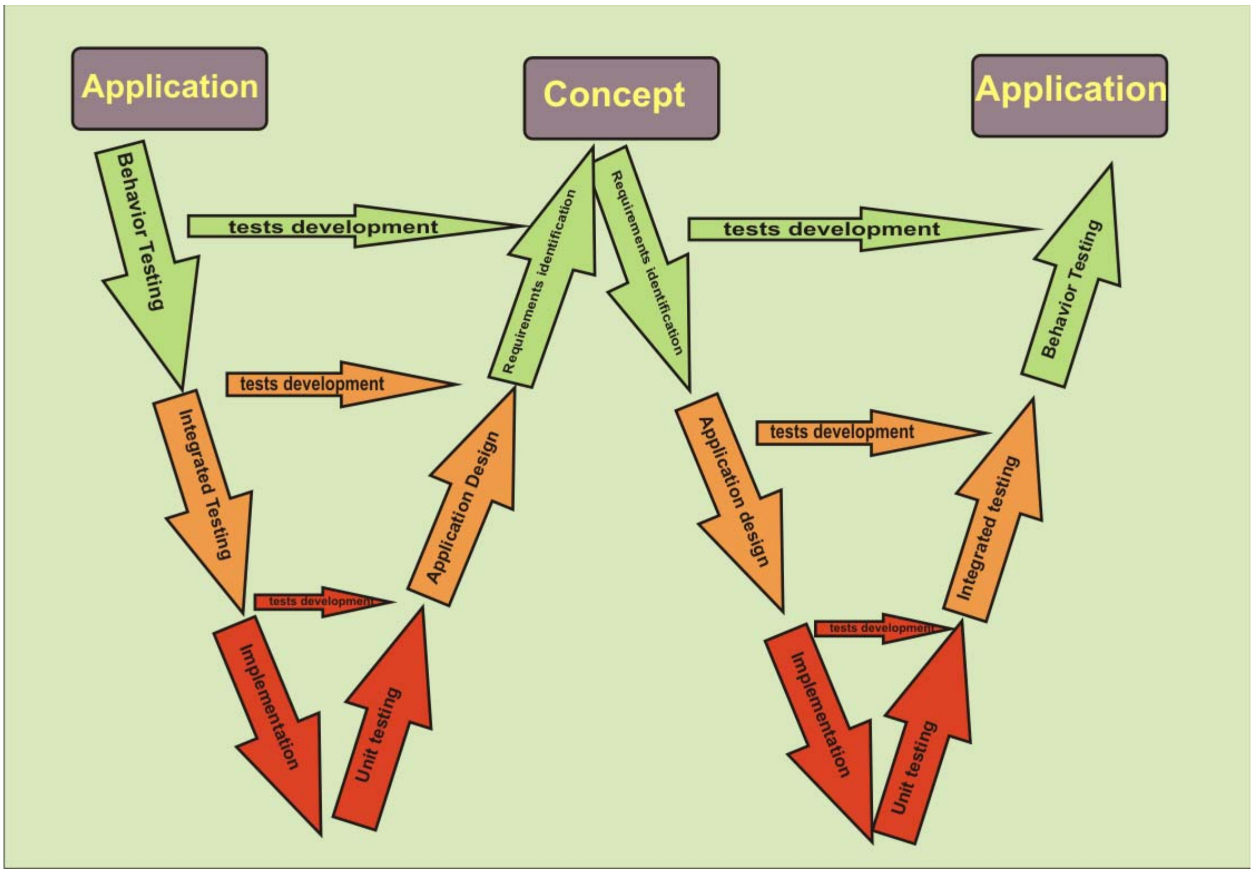

Figure 2. W testing model for software reengineering

In the distributed econometric application for two-stage least squares regression analysis, there are some important aspects, both for black-box, beravior, testing and for the white-box, structural testing.

First of all, the testers have to define the econometric problem, that has to be solved through the software. The defining of the problem presumes the description of the data entered in the regression model, the number of exogenous variables, the number of data series. They have the possibility to enter the data by hand, by completing the tables appeared on the web page or by reading it from binary or XML files. Sometimes, generators linear models can also be used for defining the series of data more rapidly.

Secondly, the testing of correctness of the coefficients determined from the model is prepared, by using the two-stage least squares method, and consequently the two-stage least squares algorithm. The testers are to determine whether the entered data is correctly instantiated in large scale matrices and if the calculations are made correctly. If there are any errors of reading the data from the files, they are recognized and exemplified by the testers.

Another important aspect is the determination of the exact code sequences that generated the results of the calculation of coefficients. The testers introduce several series of data repetitively in order to see whether the application is stable and whether it reflects the modifications made in the series of data entered again.

The computation of the error [11], [12] from the regression model and the analysis of the error are tested, in order to determine whether the error is set between the acceptable limits. In (5) and (6) the formulae for the computing of the error and its limits are presented.

$$Err_i = y_i - \hat{y}_i \qquad (5)$$

where:

- Err_i represents the error term from the series *i* of data from the regression model;
- y_i represents the endogenous, dependent variable of the series *i* of data from the regression model;
- $\hat{y}_i$ represents the estimated endogenous variable;

$$LimInf <= Err_i <= LimSup \qquad (6)$$

where:

- $LimInf$ represents the lower limit of the error;
- $LimSup$ the upper limit of the error;

The testing procedure continues with the computing of the residual sum of squares [11], from (7) which represents the unexplained variance of the model;

$$SSR = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad (7)$$

where:

- SSR represents the residual sum of squares;
- $\hat{y}_i$ represents the fitted value for each observation;
- *n* represents the number of data series from the model;

After the computation of the residual sum of squares, the testing procedure goes on to the computing of the total sum of squares, as total variance of the dependent variable, presented in (8), [11] and (9).

$$\overline{y} = n^{-1} \sum_{i=1}^{n} y_i \qquad (8)$$

$$SST = \sum_{i=1}^{n} \left(y_i - \overline{y}\right)^2 \qquad (9)$$

where:

- SST is the total sum of squares;
- $\overline{y}$ is the average of the dependent variable;

The explained sum of squares is defined as the explained variance of the model and reflects the sum of the differences between the fitted values $\hat{y}_i$ and the average of the dependent variable and is presented in (10)

$$SSE = \sum_{i=1}^{n}\left(\hat{y}_i - \overline{y}\right)^2 \qquad (10)$$

The test case will continue by checking whether the total sum of squares is equal to the sum of the explained sum of squares and the residual sum of squares, shown in the relation (11);

$$SST = SSE + SSR; \qquad (11)$$

The next step consists of the computing of the coefficient of determination R^2 presented in (12) and whether it belongs to the interval [0,1].

$$R^2 = SSE / SST = 1 - SSR / SST \qquad (12)$$

Additional aspects related to how the results are saved in binary files, XML files or in databases are also tested, in order that the data should be used again in the future. The testing team checks if the data is correctly saved and whether there are any conversion errors generated by the software during the serialization and deserialization procedures.

All of the relevant aspects determined in the testing stage are taken into account for the transformation process. The recognition and solving of computing issues ensures that the process of reengineering will gain in efficiency and will offer a higher level of quality to the software application.

The test cases are also reflecting upon the security aspects of the application. However, in the following section, several security issues in distributed applications development will be described.

4. Security assurance in distributed econometric applications development

In distributed applications reengineering the security assurance is a very important element, which has always been seriously considered. The work in distributed environments such as computer networks and the sharing of information through the Internet has to notify the presence of threats and dangers.

In our case, due to the technology implemented in the two-stage least squares distributed application for econometrics, it is compulsory to assure that the data and the software are protected in an appropriate manner.

When users want to access the websites of the distributed econometric application they have to pass through the authentication procedure. They have the possibility to create a user account, and after that to login for being redirected the main page.

Asp.NET forms authentication is used for reaching the authentication process, with all the elements provided by .NET Framework [9] and Windows Server 2008. Before the reengineering process, the application did not have any authentication procedures defined, since it hadn't been projected for the distributed environment. The process of reengineering transformed it radically into a distributed application with special needs, such as concurrent access, authentication, authorization and cryptography. Consequently, custom methods for defining the process of forms authentication have been initialized, due to the target groups' requirements.

The encryption of configuration files has also been achieved for the distributed econometric application, and it was straightforwardly done by means of the .NET Framework, just through the addition of special directives for security.

Before the encryption, the connection strings section from the configuration files looked like in figure 3. The evolution of requirements brought an evolution in the applications configuration security, so as the configuration files and the password table to become encrypted.

```
<connectionStrings>
   <add name="conectionutiliz" connectionString="Data Source=.\SQLEXPRESS;Initial
Catalog=phdresearch2;Integrated Security=True;Pooling=False" />
   <add name="phdresearch2ConnectionString" connectionString="Data
Source=PROGRAMARE03-PC\SQLEXPRESS;Initial Catalog=phdresearch3;Integrated
Security=True"
     providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Figure 3. Web.config connection strings section

After the writing of the encryption method presented in figure 4 and [8], transformations are suddenly made for obtaining the results shown in figure 5. The connection strings section from the web configuration file specifies that there are two databases integrated in the application. The reasons for the integration of two databases are the fact that the econometric application uses a database distributed by replication.

```
    private void EncryptConnectionString()
    {
        System.Configuration.Configuration config =
WebConfigurationManager.OpenWebConfiguration("~");
        ConfigurationSection ConnectConfigSection = config.GetSection("connectionStrings");

        if (ConnectConfigSection != null
            && !ConnectConfigSection.IsReadOnly()
            && !ConnectConfigSection.SectionInformation.IsProtected
            && !ConnectConfigSection.SectionInformation.IsLocked
            )
                {
ConnectConfigSection.SectionInformation.ProtectSection("DataProtectionConfigurationProvid
er");
                    ConnectConfigSection.SectionInformation.ForceSave = true;
                    config.Save(ConfigurationSaveMode.Full);
                }
        else
                {
                    ConnectConfigSection.SectionInformation.UnprotectSection();
                    ConnectConfigSection.SectionInformation.ForceSave = true;
                    config.Save(ConfigurationSaveMode.Full);
                }
    }
```
Figure 4. Asp.Net C# encryption method

The method form figure 4 uses a configuration object defined in the ***System.Configuration*** namespace and in the ***System.Configuration.Configuration*** class. In the method, there is also specified that the section from the web configuration file which will be encrypted is the connection strings section. The security provider for doing the encryption is ***DataProtectionConfigurationProvider*** and it implements Windows Data Protection Api. The key idea is that [8] the Windows Data Protection Api does not allow the exporting of encryption keys, and consequently the application has to run on just a single server, which is good for our case.

If it is necessary to work on multiple application servers the ***DataProtectionConfigurationProvider*** has to be replaced by ***RSAProtectedConfigurationProvider*** which use a RSA publik key cryptography algorithm.

```xml
<connectionStrings configProtectionProvider="DataProtectionConfigurationProvider">
    <EncryptedData>
      <CipherData>
<CipherValue>AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAWqz2nwxtwUucSibHUAfMTwQA
AAACAAAAAAAQZgAAAAEAACAAAACU+WAl43JVod9aEOcmhjHJ6DOKgX6vlsQTzj1dF+Vr
nwAAAAAOgAAAAAIAACAAAABInEtiJ4PXYzffxzzi4FzJoRIJnJhsoe5XvOgEmM57QiAFAACygLU
+4nki+6k7WjtQe7Q2FzEXon5Or8A0s7LMQpHxQg1Nh/r0EVDHu4etudMBkGNgrwbC7nkLu
rkFr7kst3AFoZO85Bb3xDc0fkMdLTVD995MgaUIWkVh4f4KDgtikqU4U/OKpBmGciRKhD6uV
mfi30jCiNVtITepmqWDM1RC7ZIOTjgT0J5Asx+Da8Tc0OjkVSGpHS8rYrG+NnAnUyUn7EJj3b
MPTxNCHZGFx/vgG/qcsfXvRThsdzGRkDL8VRFJF+v0hIdqIGrg2g+VFQkbIQ7aHD/Kwk21xEpv
Lv7ZComc+FZuF58c4fB6ZJEAg9VI5/ybBjClis7aOsIHBX2+EYMUiDuZ1qlRh0bStwjWInbGG7G
FrzWRHnUAMx/Xj3YpK3bRIqmuw8+6/ixXvxngNgNeTQVdiusUj6PksHqbFv4hdcPSdFPuCG6li
Fq9+IkkmCkYMvZ+ODg3GtcEtbr4W4EHz2kHRP2CDmqoNChntYq5FnrxlgXfXuB44X4+6PBes
wsgwrN0dl9wFAFrrnNbnBzqRUcweZqE/MUfvX4x5JOiqJ6dQrAvVSBBAcCFyjBhvKYi0qp0MTE7
DiaQtWFKaIjqiT3H45I56t58NNLC2HDSR9QcZYxLncOyBR/MOAmQQqQPIGkvGkM2nsnFeY
32+5s4WlJrGimw8e/dB+fBGw6R0rcBBw9vNNmyYJJFoUMXh+QLOei6U4ZlWvWml8h02RlQ
1ofpxpkcF+Y/DZ2ZnYXYeViaUYwxupjCrKdIf99vDGqTixv6CP4N7W0nAX1UAqnes4es/TSOSl
DZA5HAHcDKrDcD2LTU/XEOGjtHgiYgcGkUMLdrC5dn7DxE/a2bjcIVY9shyjd8I+4Dm4raiPQ
CtjCb5IdLml3fxEjGnnQBJRYEi7I2EN+Zz3mVTFEGBVbDeHjgSSAtfSg+JMzxj6A5UbUWmqXy7
K5UV1TeQThQu+McpwElJrKafkP41LQtkOeqXzpQBjqehgZByf1lkpqkS5BGi1eYqzH1K/urfB0G
KQE7iY4jraakBMNpJKOrLTnLjnNNEaLeu9GbVmZBhUHLboUVZvGJhWX7OievT3oQD+cDsl5
uMxW+uQvTc+rs5lLlzadHtmhVuEopa12CoygedvsKc2OQB6HKeXPjX0sxwXr2YbuGnmoBI6jJ
6lgZBzhKObCBtW9ypBwOBovVIpLZ8WZjGiWuXHxIqPW4IlTP2QWT1iBxJxn9x0SUvN/zsr6ejWf
0Ow1ZlMvP+9xTHVmQSBfwDOjSLjrAmypx9ajMhR60VcvgDhTWbwHoHgWwq4QNx2eB+Bw
+U3n31asYnQxFGi8l9L+uoinNc46OZpn5ASUHFgli1fL+kvCGdgIhwUdj8eRX/D2qGljLYAEE5
wXlIQ2uEXwaN56IcT+ycWicXAxSx1S4Il1VlVSWniboe+cY0Z/abfs/UXwuM4B8uBy4FzrrAsVxyij
jO/TUAcVi58ecTLq+VowYOTEY/mwT+bYg/a34DuqJX8cilHnSYjm+UCNYaYXAIR5bHvaolecL
66HtSqENLiTXjD3eWQuDHbUFErgCYwg0fiIs93OtzPtA2MLS4yzR8l8jaBJhdCuhgf8gdsDvLdYZ
VfbEJKhP6VRZel0g4IF5rkwsq/SelQ+AaEzoJhWVCB/t/FHLoCzIGamI7cZxB7VY+yXmBXgMKU
ZdnP8e5fCAFDdotVPhgacJOP91Q3iWiuyJ5ihMz8ElJYNz22pK6q9hSoQgMhqSQAAAAP4FsZ
NdxtrdnAZUjvCX5ve49csGV1XuYHBLOO+MJKE/3J9aGVQnqkhYhT/SQtQLS6qECtD69BUjaJr
Z44LGSes=</CipherValue>
      </CipherData>
    </EncryptedData>
  </connectionStrings>
```

Figure 5. Web.config with encrypted connection strings section

In figure 5 the encrypted section of connection strings from the configuration file is presented. Potential hackers cannot connect to the databases and access the data, because of the encryption of the connection strings. The only machine which can decrypt the above code is the server on which the application runs on. Attackers may want to break in the server, but it is very difficult, because Windows Server 2008 is one of the most secure servers ever built.

Another very important thing in distributed applications development is the implementing of parameterized SQL Commands and stored procedures. In distributed econometric applications, the work with the SQL commands has to be done with maximum of care, because hackers may want to forge identities and delete the data from tables. Another important danger which has to be prevented is the SQL Injection Attack. Both of the dangers are prevented by solid authentication and authorization procedures and using correct parameters in SQL commands.

The next iteration of reengineering supposes moving to Windows Communication Foundation and to integrate the existing modules of the application in WCF Services. When talking about Windows Communication Foundation security aspects we have to take into account the transport level security and message level security.

Each one of the above mentioned security directions are important for the development of reliable econometric software as distributed applications. In this case data sent by messages between services in distributed environments is as well encrypted.

To conclude, we may say that security reengineering integrates itself in the IT reengineering project and offers a higher level of confidence in the use and administration of the distributed econometric application.

5. Conclusions

To sum up, this paper has described the objectives achieved by the authors in the field of distributed applications development for econometrics. This desideratum of the reengineering presumed the defining of new strategies of testing, which are different from the ones mentioned in the usual testing model, due to the starting point of the project, which consists of the active application, being subjected to reengineering.

We defined our own indicators and metrics so as to determine the general image of the existing software application, which has been subjected to reengineering. Consequently, the quality assessment became more efficient and the reengineering team had the opportunity to get a more accurate image about the work need to be done in the following stages of the IT project.

Security reengineering implies the study and optimization of the entire software entity at all levels. Furthermore, security reengineering does offer new security strategies concerning the software product, such as architecture, design, source code and database tier. Migrating to new technologies, such as WCF supposes that important changes are made in the security policies, on the basis of the integration of human readable applications into machine to machine communication over distributed systems.

6. References

[1] Encyclopaedia Britannica – online, 2010, www.britannica.com

[2] Mens T, Demeyer S. **Software Evolution**, Springer, 2008, ISBN 978-3-540-76439-7.

[3] Bergsion, H. **Creative Evolution**, Henry Holt and Company, New York, 1911, ISBN 0-486-40036-0.

[4] Ivan, I., Popa, M., Tomozei, C. **Reingineria entitaţilor text,** Revista Romana de Informatica şi Automatica, vol.15 nr.II, 2005 pp. 15 – 28, ISSN 1220-1758.

[5] Tomozei, C. **Hypertext Entities Semantic Web-Oriented Reengineering**, Journal of Applied Quantitative Methods, vol .III nr. 1, 2008, pp. 9- 19,ISSN 1842-4562.

[6] Koschke, R. **Identifying and Removing Software Clones**, Software Evolution, Springer, 2008, pp. 15 – 36, ISBN 978-3-540-76439-7.

[7] BLACK, R. **Critical Testing Processes: Plan, Prepare, Perform, Perfect**, Addison Wesley, 2003, ISBN 0-201-74868-1.

[8] Wildermuth, S., Blomsa, M., Wightman, J. **Microsoft.NET Framework 3.5 – ADO.NET Application Development**, Microsoft Press, 2009, ISBN 978-0-7356-2563-1.

[9] Northrup, N**. Microsoft.NET Framework 3.5 – Application Development Foundation**, Microsoft Press, 2009, ISBN 978-0-7356-2619-5.

[10] Fields, J., Harvie, S., Fowler, M., Black, K. **Refactoring, Ruby Edition**, Addison Wesley, 2010, ISBN-13: 978-0-321-60350-0

[11] Wooldridge, J.M., **Introductory Econometrics, A Modern Approach**, South Western Cengage Learning, 2009, pp. 506-546, ISBN-13: 9780324581621.

[12] Wooldridge, J.M **Econometric Analysis of Cross Section and Panel Data**, The MIT Press, 2002, ISBN 0-262-23219-7.

[13] Patrut, B., Pandele, I. **How to Compute the References Emergencies in a Hyper-encyclopedya**, Recent Advances in Systems Engineering and Applied Mathematics. S WSEAS conferences in Istanbul, Istanbul, 2008 p.72-75, ISBN 978-960-6766-91-6, ISSN 1790-2769.

---

[1] **Cosmin TOMOZEI** is University Assistant - Lecturer at the Mathematics and Computer Science Department from Faculty of Sciences of the "Vasile Alecsandri" University of Bacau. He is a PhD candidate from October 2007 at Economic Informatics Department from Academy of Economic Studies, Bucharest. He holds a Master in Science - Databases - Business Support from the Academy of Economic Studies, Bucharest. He graduated in Economic Informatics at Faculty of Economic Cybernetics, Statistics and Informatics in 2006. His main research areas are: object oriented programming, functional programming in Lisp and F#, software reengineering and distributed applications development. He is the author of 27 peer reviewed scientific papers.

[2] **Bogdan PATRUT** (b. June 16, 1969) received his BSc in Informatics (1994), MSc in Distributed Programming (1996), PhD in Accounting and Business Information Systems (2007) from "Al. I. Cuza" University of Iasi, Romania, and PhD in Informatics (2008) from "Babes-Bolyai" University of Cluj-Napoca. Now he is associate professor of informatics at Mathematics and Computer Science Department, Faculty of Sciences, "V. Alecsandri" University of Bacau, Romania. His current research interests include different aspects of Artificial Intelligence. He has (co-)authored 23 books and more than 20 papers, more than 10 conferences participation, member in International Program Committee of 4 conferences and workshops.