Reliability in Distributed Software Applications

Cătălin Alexandru TĂNASIE, Sorin VÎNTURIŞ, Adrian GRIGORIVICI
Academy of Economic Studies, Bucharest, Romania
catalin_tanasie@doesec.ase.ro, sorin.vinturis@ie.ase.ro,
adrian.grigorovici@doesec.ase.ro

Reliability is of vital importance for distributed software application and should be ensured in all stages of the development cycle. Ensuring a high level of reliability for distributed software applications leads to competitive applications which increase the level of user satisfaction. The aim of this paper is to present techniques and methods which ensure high level of reliability. A model for estimating the reliability through risk assessment is presented. Distributed software applications are composed of multiple components spread across multiple heterogeneous platforms and partial failures are inherent. To ensure high reliability is very important that the input data for distributed application components are correct and complete.
***Keywords**: Distributed Applications, Reliability Model, Risk Assessment, Data Acquisition*

1 Introduction

Reliability refers to the ability of a software application to maintain the performance level in a certain environment for a specified period of time. In [7] reliability represents the characteristic of the informatics application to run correctly and completely for all data sets inputted by the users. A reliable distributed application is defined in [2] as a system whose behavior is predictable, in spite of partial failures, asynchrony, and reconfiguration.

Reliability shows [4] the level in which a program conducts to correct and complete results. An estimated reliability is found which refers to the software product behavior who is in one of the phases of development and more than that the effective reliability, I_F, which is calculated as a report between the number of success running, N_S, and the total number of iterations, N_T, of the software product:

$$I_F = \frac{N_S}{N_T}. \quad (1)$$

During all stages of developing of a computer application, starting with product design and continuing through implementation, testing, product obsolescence, post-production, reliability is assured every step by choosing the right technologies and ensuring a proper technical training and expertise of people involved in the product development.

In Figure 1 is presented the reliability engineering with the major activities of the product development cycle:

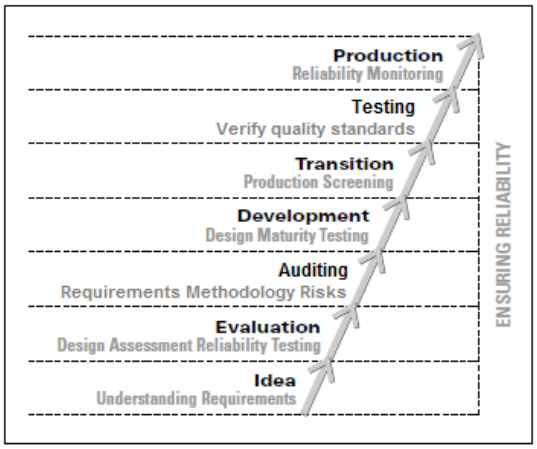

Fig. 1. Reliability engineering

Application users are interested in high reliability software products, since the purpose of reliability is [10]:

- to analyze the defects identified in the testing phase; learning the causes that lead to the arise of the problems, which were the processes involved, what methods of control must be applied to remove and prevent their future occurrence;
- to provide a quantitative understanding of the application behavior over time, depending on the internal and external influence factors, such as the maximum number X of server processes that can be used to achieve a number of Y transactions during Z period; when these limits are exceeded the performance significantly decreases;
- to establish methods and models for calculating and forecasting the reliability, by monitoring application behavior during a certain period of time in a real operating environment;
- to establish methods for maintaining and enhancing the reliability of future distributed applications;
- to establish methods for selecting and processing data on products reliability and to determine the optimal values of reliability indicators.

The effects that occur in the context of a lack of reliability of software products should not be neglected; depending on the application usage, they have very important economic implications, by producing additional maintenance costs and increasing the number of resources needed for solving the defects; they also determine failure to comply with deadlines, or may even endanger human lives, in case of defects appearing to the railway switchgear system.

Reliability of distributed applications requires both a qualitative and quantitative approach [10]:

- qualitative approach refers to the ability of the information system to meet the specified requirements, in the defined environmental conditions, and proper functioning in a preset time; important qualitative indicators are: the ability to resist failures, the ability to recover after failure due to a system crash or a power failure; this is achieved through very well designed error recovery or backup systems;
- quantitative approach means achieving the designed functions and delivery of all duties without fault and with a certain performance, in a given period of time; it is based on numerical indicators, obtained through comparison with other existing systems or forced by user needs.

Testing is very important to increase reliability. It can be applied at different phases of the application development, starting right from the design stage, being the most important way of verifying the distributed application.

The causes leading to low reliability [3] of the distributed applications are generated by:

- *weak design concepts* caused by immature or inferior design concepts;
- *escaping reliability problems* that impose higher service costs and design modifications;
- *constrained human resources* induced by inexperienced developing team;
- *constrained prototypes* caused by insufficient funding and development time; this is important to develop robustness and grow reliability and to obtain user feedback;
- *overloaded resources* leaving no reserve capacity for variations in demand;
- *lack of attention to risk management* when risks become actual problems to be solved, leading to delayed delivery dates;
- *lack of flexibility in design* that cause serious consequences for schedules and costs;
- *late changes in requirements* leading to either late-arriving feedback from users or new expectations set by competitive entries;
- *weak investment in early development* leading to numerous problems that can

force costly corrective actions in the later development phases.

The challenges are to identify the results that are not acceptable and to establish their priorities and their root causes.

Non-reliability is obtained by [5]:

- using inadequate content;
- making operations with particular values which position the condition indicators such that further processing is impossible;
- altering the content of control variables which determines referring memory zones with random content.

Distributed application developers must apply strict rules to eliminate the conditions for non-reliability, because most of the application functional and security defects are related to code quality which, if improved, will minimize undesired effects in reliability.

In complex distributed software applications reliability problems are observed by decreasing performance over time, and because robustness is the basis of reliability, a concentration of the team members involved in the development process, to achieve the reliability objectives, should be taken into account. Reliability for a distributed software application requires good performance during operation, and not just being functional [3].

There are no perfect systems, but the goal is to increase its reliability and finding ways to improve performance, reliability being an essential requirement for distributed software applications [6].

2 Reliability in data acquisition process

Data acquisition is an important activity, with major influence in ensuring the data quality. The risks of errors in data acquisition process are related to data collection method. The data acquisition is achieved through the following methods [8]:

- the input of data by operators using a keyboard;
- the taking over the data from various media, such as optical or magnetic;
- the taking over the static or moving image from the camera or video camera;
- the reading of data by scanning the text, the images, the barcodes, and the magnetic stripe cards;
- the taking over the data by capturing the sound, using the microphone, or the ultrasound;
- the analog data collecting using a combination of suitable devices.

In the processes of data acquisition and monitoring is necessary to use information systems and devices to achieve:

- elimination of errors caused by human factors, such as errors in reading measuring instruments, data transcription errors in primary documents, errors in reading of numbers and letters;
- human effort focusing on analysis and interpretation of data, which leads to increased efficiency of human resources.

The error represents the difference between actual and measured value of a certain characteristic. The errors that occur in data acquisition and data entry processes include:

- *filling errors* of primary documents consisting of careless transcription, incomplete data transcripts, characters reversals, replacing data with other, noncompliance of the measurement unit;
- *typing errors*, such as confusion of letters, numbers or characters; these errors are caused by inattention and mistakes of understanding;
- *time errors* are caused by delays occurred between the time of data acquisition and data correction; delays that generates errors occur between the following moments of time:
  - acquisition - summarization;
  - summarization - processing system input;
  - input - displayed results;
  - displayed results - stakeholder reporting;
  - decision acknowledgment - decision implementation;
- *errors that occur during the life cycle of data*, caused by a correct value of data that became invalid over time.

Errors in data production are caused mainly by poor planning and inadequate manage-

ment of data production processes, issues that determine their poor quality. Causes of errors in compiling data [8] are presented in Table 1.

Table 1. Causes of errors in the data production

Deficiency	**Causes**	**Fix**
different values for the same data	multiple sources, occurrence of spurious data on the chain of transmission	development of common definitions and consistent procedures, elimination of technical failures
data loss	systemic errors in data production	statistical control of the processes, improvement of the processes, behavior control and proper incentives
difficult data access in reasonable time	large amount of stored information	rewriting of the informatics applications using graphical user interface and the usage of the customer systems
definitions, formats and inconsistent values	heterogeneous distributed systems	data warehouses
change of useful data	changes in tasks of users and in organizational work environment	anticipation, changes of the users' tasks, review of processes and systems before the failures to determine the stop of informatics applications
limited data access	insufficient calculation resources, restrictive policy of informatics security	policies development of policies modernization, so that consumers know when to expect more resources

Measures [9] implemented to prevent and eliminate errors in data are presented in Table 2.

Table 2. Measures to prevent and eliminate data errors

During data collection	**During the input of data in the system of processing**	**After the input of data**
- design of primary documents - verification of completed documents - specialization and training of the staff - creation of the conditions for correct completion - ensure of optimum ratio between the volume of operations and the number of people participating at the collection and processing	- taking over the correct data from the document -elimination of intermediaries - double input - scanning	- validation program

Storing the same data in multiple locations is an error-prone source, because it is difficult to ensure a consistent update of all children, and should be avoided.

Elimination of intermediate links in the chain of data entry in processing systems means eliminating of potential sources of input errors. To achieve improved data quality, it is necessary to separate different aspects associated with the data, such as intrinsic properties, acquisition systems and data delivery.

Ways to increase the reliability of data acquisition process is achieved by applying the following factors:

- human operators, by
  - appropriate selection of staff;

- employees training;
- creating appropriate working conditions for data entry operators so that their attention is solely focused on their activity;
- operators activities checking;
- appropriate wage and salary correlation with the number of errors made by the operator;

- equipment, by:
- choice of input data devices , having performances appropriate to the corresponding needs;
- equipment installation according to the norms and providing equipment quality service;
- equipment calibration;
- verification of equipment operation in normal operation.

The complexity of software applications is given by a combination of factors whose influence is determined by the levels of quality characteristics. Complexity is studied in conjunction with reliability, maintenance and stability. A great importance in this analysis is given to data quality. Due to the costs they incur in an organization, and that the poor quality of data causes large additional costs of processing, data quality is a priority for any successful management.

3 Increasing system reliability through risk assessment

Distributed applications encompass multiple physical or logical components, united by a common architecture and communicating through an array of heterogeneous environments. In this context the manner in which the system reacts to scenarios like the loss of component functionality or security threats represents an important aspect in addressing the issue of global system reliability.

In order to provide an accurate model for predicting and improving a distributed application's threat response capabilities, the following factors are considered:

- *application scope*, or *target domain*, with effects like increased susceptibility to certain incidents or necessity for running and communicating under certain environments;
- *component and global redundancy*, the degree in which an incident affecting an individual node can prevent the proper functioning of another or of the whole;
- *application users*, specialized components that are accessed by trained professionals are more reliable than general purpose ones, as the users are less susceptible to being a security concern;
- *threat awareness in application design*, or the manner in which the architectural features and component role target threat analysis and prevention; error handling, secure access to core functionalities, specialized tools in dealing with concerning effects, incident logging capabilities ranging from simple files to a specialized database.

The current model considers a data acquisition module in the application, responsible for collecting raw data in various formats and storing it in both the initial form and in a refined one derived through performing a series of operations meant to bring this information to a common standard in order to help analysis. The application scope is ultimately testing, validating and refining risk assessment models in the distributed systems environment.

To achieve a high level of reliability in the development cycle of distributed software applications, the following aspects should be taken into account:

- *data source incompatibility* – in order to correctly assess the risks, the raw data used in testing the models must allow for collecting sufficient compatible information, as shown below – numerical values included for exemplification, alphabetical order not relevant:

Table 3. Raw data compatibility

Source (raw data type)	**Source A (audio)**	**Source B (images)**	**Source C (text)**	**...**	**Source K (type K)**	**...**	**Source N-1 (video)**	**Source N (video)**
Source A (audio)	—/1	0,1	0,01	...	comp(A,K)	...	0,5	0,5
Source B (images)	0,1	—/1	0,05	...	comp(B,K)	...	0,1	0,1
Source C (text)	0,01	0,05	—/1	...	comp(C,K)	...	0,001	0,001
...	...	...	...	...	...	...	...	...
Source K (type K)	comp(A,K)	comp(B,K)	comp(C,K)		—/1		comp(N-1,K)	comp(N,K)
...	...	...	...	...	...	...	...	...
Source N-1 (video)	0,5	0,1	0,001	...	comp(N-1,K)	...	—/1	1
Source N (video)	0,5	0,1	0,001	...	comp(N,K)	...	1	—/1

where:
$\mathrm{comp}(X, Y)$ – the compatibility degree between the X and Y data sources, allowing for the existence of identical data types – as is the case with sources N-1 and N, with the following properties:

$$\mathrm{comp}(X, Y) = \begin{cases} s, \ \forall X \neq Y, s \in R, s \in [0,1] \\ —/1, \ X = Y \end{cases} \quad (2)$$

$$\mathrm{comp}(X, Y) = \mathrm{comp}(Y, X) \quad (3)$$

and

$$W = X \iff \mathrm{comp}(X, Y) = \mathrm{comp}(W, Y), \forall Y \quad (4)$$

- *data quality inconsistence*, a variation in the mentioned aspect even when considering the same data type, relating to the hardware device used through-out the collecting phase; the problem does not arise from the technical aspect but from the possible assessment errors caused by comparing two same-type sources and analyzing a variable quality amount of identifiable elements; the (4) property mentioned above is considered;
- *unrepresentative model*, as risk assessment is by its nature a heterogeneous domain and there are no clear boundaries as to what the qualitative impact of a new model will be; in order to avoid this issue, clear information on the factors included in the model is required, as well as allowing for evolutionary components inside the model;
- *natural information loss* during the data collecting and refining stages.

In order to avoid quality-related issues relating to data and algorithms used in the risk assessment models the following steps are taken:

- ensuring that meta-information acquired through analysis on a given medium encompasses only exogenous evolution factors, in relation to technical or contextual aspects not included in the model;
- enlarging the raw information database, with effects in minimizing the error margins;
- the rigorous testing of the models;
- the development of an evolutionary algorithm in order to help increase application reliability and also serving as a risk assessment model; this approach is presented in the following sections.

4 Emphasizing compatibility factor in the model

A model for assessing the manner in which the system reacts or has reacted to individual incidents is composed of two logical areas:

- a *qualitative assessment* of component behavior facing an array of identified threats, together with and influenced by a *quantitative* statement of previous costs in removing the effects of incidents;
- an *evolutionary algorithm* based on previous threat response assessment, which uses the behavior of the existing version

both as the source information for a new one and also as the benchmark to which this improvement is subjected.

Considering the distributed system *S*, serving as a collection of *n* components, *c*, a number of *m* threats*, t,* composing the array *T* and an ordered array Q of individual component and threat-related *qualitative marks* corresponding to thresholds in the system owner's cost analysis, as well as *quantitative marks* corresponding to previous, calculated costs, brought down to scale, we compose the following:

Table 4. Component/threat correlation

S\T	t_1	t_2	…		t_m
c_1	q_{11}, h_{11}	q_{12}, h_{12}	…		q_{1m}, h_{1m}
c_2					q_{2m}, h_{2m}
…	…	…			…
c_n	q_{n1}, h_{n1}	q_{n2}, h_{n2}	…		q_{nm}, h_{nm}

where:

n – number of components in the system;

m – number of identified component-related threats;

q_{ij} – qualitative assessment mark corresponding to threat *j* for component *i*;

h_{ij} – quantitative assessment mark corresponding to threat *j* for component *i*;

The value interval is defined as:

$q_{ij} \in \{QM_1, QM_2, \dots, QM_k\}$,$h_{ij} \in [a, b]$, $a, b \in R_+$,

having

$QM_1, QM_2, \dots, QM_k$ – array of *k* qualitative marks; an example for k = 5 is {*Very Bad*, *Bad*, *Average*, *Good*, *Very Good* };

a, *b* – real, positive numbers defining costs broth to scale by reporting to a maximum considered loss – C_{max}.

Calculating the scale index starts from the maximum defined loss value of C_{max} and the top limit of the scale:

$$I_s = \frac{C_{max}}{b} \qquad (5)$$

$$C_{max} = \max\left(C^i_{j\,(e)}\right) \qquad (6)$$

described as:

I_s – *scale index* or, simply, system costs *scale*;

C_{max} – the maximum registered or estimated cost caused by treating the system losses caused by an incident;

e– the current stage or version in the evolution of the application reliability and risk assessment model;

$C^i_{j\,(e)}$ – the cost of removing the damage caused by a type *j* event in component *i*, considering recommendations or testing the model in the *e* stage.

Any $C^i_{j\,(e)}$ cost is represented on the given scale as:

$$\widehat{C^i_{j\,(e)}} = \frac{C^i_{j\,(e)}}{I_s} \qquad (7)$$

$$\widehat{C_{max}} = b \qquad (8)$$

The *qualitative coefficientsQM* are included in the model and relate to the estimated losses caused by the occurrence of a given type of incident. They are included for events on whose behavior little is known on stage *e* and their relation to the *quantitative coefficients* is the following:

$$QM_i \Leftrightarrow \begin{cases} \left[a, \frac{C_{max}}{k}\right), i = 1 \\ \left[\frac{(i-1)*C_{max}}{k}, \frac{i*C_{max}}{k}\right), i = \overline{2, k-1} \\ \left[\frac{i*C_{max}}{k}, b\right], i = k \end{cases} \qquad (9)$$

The relation between variables in *Table 2* and coefficients is the following:

$$q_{ij} = QM_{ij} \in QM_i, i = \overline{1,n}, j = \overline{1,m} \quad (10)$$

$$h_{ij} = \widehat{C_{J\,(e)}^{l}}, i = \overline{1,n}, j = \overline{1,m}, \forall e > 0 \quad (11)$$

Thus the problem of improving component or system reliability translates to the following two objectives:

- minimizing incident-related costs;
- algorithm reevaluation.

We consider $mod_{(\mathrm{e})} : \mathrm{T} \rightarrow \mathrm{E}$ a function describing the on cost values after applying the model in its *e* stage, otherwise formulated as envisioned by the model in its *e-1* stage, E being the cost assumption collection and T the identified risk collection.

$$mod_{(\mathrm{e})}\,(t_1 t_2, \dots t_m) = \mathrm{E}(\sum_{j=1}^{m} \widehat{C_{J\,(e)}^{1}}, \sum_{j=1}^{m} \widehat{C_{J\,(e)}^{2}}, \dots \sum_{j=1}^{m} \widehat{C_{J\,(e)}^{n}}) \quad (12)$$

or at component-level as:

$$mod_{(\mathrm{e})}(t_j) = \widehat{C_{J\,(e)}^{1}}, \forall j = \overline{1,m} \quad (13)$$

Increasing system reliability is, in the current scope, a consequence of an ever increasing efficiency in evaluating threat-related damage. The used algorithms require the component or system-wide minimization of the discrepancies between predicted and measured behavior.

The model's efficiency condition is given by the following inequalities – for a given *j* incident:

$$\widehat{C_{J\,(e)}^{l}} < \widehat{C_{J\,(e-1)}^{l}}, \forall e > 0 \quad (14)$$

$$\sum_{j=1}^{m} \widehat{C_{J\,(e)}^{l}} < m \sum_{j=1}^{n} \widehat{C_{J\,(e-1)}^{l}}, \forall e > 0 \quad (15)$$

or for the *i* component as:

$$\widehat{C_{J\,(e)}^{l}} < \widehat{C_{J\,(e-1)}^{l}}, \forall e > 0 \quad (16)$$

$$\sum_{i=1}^{n} \widehat{C_{J\,(e)}^{l}} < \sum_{i=1}^{n} \widehat{C_{J\,(e-1)}^{l}}, \forall e > 0 \quad (17)$$

and system-wide as:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \widehat{C_{J\,(e)}^{l}} < \sum_{i=1}^{n} \sum_{j=1}^{m} \widehat{C_{J\,(e-1)}^{l}}, \forall e > 0 \quad (18)$$

Estimating incident-generated losses is the result of applying the risk assessment models on primary data converted to an analysis compatible format. As the performance in analyzing behavior of the monitored process or component is partially dependent on aggregating this information, the source compatibility coefficient must be included in the model.

Component and threat-oriented cost assessment generates the need for correlating each incident type with the compatibility of its data sources. Based on the information presented in Tables 1 and 2 we identify this relationship as taking the form exemplified below. Determining the representativeness of the *K* source of a given *l* type incident is done encompassing the representativeness of this incident to the components using *K* as behavior data source.

Table 5. The component – data source and incident type – data source correlations

Component\data source	**Source A**	**Source B**	**Source C**	**...**	**Source K**	**...**	**Source N**
c_1	X	X					
c_2		X	X		X		
c_3		X					X
...	...	...	...	...	...	...	...
c_k	X	X	X				X
...	...	...	...	...	...	...	...
c_n		X					

Incident type\ data source	Source A	Source B	Source C	...	Source K	...	Source N
t_1	X	X					
t_2			X		X		
t_3							
...	...	...	...	...	...	...	...
t_l		X	X		X		
...	...	...	...	...	...	...	...
t_m							

Based on determining the component/incident type and data source correlations, the inter-source compatibility on a component or incident type axis, is introduced into the model.

Assessing costs including the compatibility component for the *i* node is given by:

$$\widehat{CMP^i}_{(e)} = \frac{CMP^i{}_{(e)}}{I_s}, \forall e \geq 0 \text{ (19)}$$

$$CMP^i{}_{(e)} = C^i_{j\,(e)} * COEF^i{}_{(e)}, \widehat{CMP^i}_{(e)} = \widehat{C^i_{j\,(e)}} * COEF^i{}_{(e)}, \forall e \geq 0 \text{(20)}$$

$$COEF^i{}_{(e)} = \prod_{k=1}^{v} \text{comp}_k\left(S_x, S_y\right) \in [0,1], \forall x, y, \forall e \geq 0 \text{(21)}$$

where:

S_x - the*x* numbered data source, relevant to component *i*;

$CMP^i{}_{(e)}$, $\widehat{CMP^i}_{(e)}$ – estimated costs for component *i* including the compatibility factor;

$COEF^i{}_{(e)}$ – the source compatibility coefficient for the *i* component, calculated as the product between individual coefficients for all uniquely associated, relevant data sources.

Assessing costs including the compatibility component for the *j* threat type or incident is given by:

$$\widehat{TIP_{j_{(e)}}} = \frac{TIP_{j_{(e)}}}{I_s}, \forall e \geq 0 \text{ (22)}$$

$$TIP_{j_{(e)}} = C^i_{j\,(e)} * COEF_{j_{(e)}}, \widehat{TIP_{j_{(e)}}} = \widehat{C^i_{j\,(e)}} * COEF_{j_{(e)}}, \forall e \geq 0 \text{(23)}$$

$$COEF_{j_{(e)_{(e)}}} = \prod_{k=1}^{v} \text{comp}_k\left(S_x, S_y\right) \in [0,1], \forall x, y, \forall e \geq 0 \text{(24)}$$

where:

S_x - the*x* numbered data source, relevant to incident type *j*;

$TIP_{j_{(e)}}$, $\widehat{TIP_{j_{(e)}}}$ – estimated costs for threat type *j* including the compatibility factor;

$COEF_{j_{(e)}}$– the source compatibility coefficient for the *j* threat type, calculated as the product between individual coefficients for all uniquely associated, relevant data sources.

The (14) – (17) relations including the source compatibility aspect in a given incident type j are:

$$\widehat{TIP_{j_{(e)}}} < \widehat{TIP_{j_{(e-1)}}}, \forall e > 0, \forall j = \overline{1,m} \text{(25)}$$

$$\sum_{j=1}^{m} \widehat{TIP_{j_{(e)}}} < \sum_{j=1}^{m} \widehat{TIP_{j_{(e-1)}}}, \forall e > 0 \text{(26)}$$

and for a given *i* component:

$$\widehat{CMP^{i}}_{(e)} < \widehat{CMP^{i}}_{(e-1)}, \forall e > 0, \forall i = \overline{1,n} \quad (27)$$

$$\sum_{j=1}^{m} \widehat{CMP^{i}}_{(e)} < \sum_{j=1}^{m} \widehat{CMP^{i}}_{(e-1)}, \forall e > 0 \quad (28)$$

The (18) criterion, used to calculate cost evolution on a global level, losses its correspondence. In considering system-wide effects similar to the ones described by this formula, the (27) and (28) equations are sufficient.

The model's performance is related to meeting simultaneously or selectively the properties described above:

Table 6. Efficiency criterion in risk assessment models based on increasing distributed applications reliability

Target criterion/Condition	(14)/(25)	(15)/(26)	(16)/(27)	(17)/(29)	(18)/(26, 28)
Increasing component *I* reliability			X		
Minimizing costs generated by incident type *j*	X				
Minimizing costs generated by incident type *j* in component *i*	X		X		
Increasing multiple component reliability		X			
Minimizing costs		X			
Model efficiency relating to application reliability					X

5 Conclusions

The presented model is relevant to the assessment of cost-measurable IT risks. It is extendable based on identifying additional factors such as recovery time or component grouping based on cascade vulnerability susceptibility. A supplementary advantage in extending the current model is the establishment of qualitative to quantitative coefficient weights. Aggregating information is made based on the inter-coefficient relationship (9).

Reliability in software development cycle has a very high impact in the maintenance process. If techniques to improve reliability are correctly applied, the resources needed in the maintenance process decreases and the need for refactoring, that is a lot more difficult than to create the product from scratch, vanishes.

Acknowledgements

This article is a result of the project „Doctorate in Economics at European standards of knowledge" (DoEsEC). This project is co-funded by European Social Fund through The Sectorial Operational Program for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies.

References

[1] C. A. Tănasie, "Open Source Medical Software – OpenMRS", *Open Source Science Journal*, Vol. 3, No. 1, 2011, pp. 5-20, ISSN 2066-740X.

[2] M. Cristescu. L. Ciovică, "Estimation of the Reliability of Distributed Applications", *Informatica Economică*, Vol. 14, No. 4, 2010, pp. 19-29, ISSN 1453-1305.

[3] J. P. King, W. S. Jewett, "Robustness development and reliability growth: Value-adding strategies for new products and processes", *Prentice Hall*, Michigan, USA, 2010, 603 pg., ISBN-13: 978-0-13-222551-9.

[4] M. Doinea, "Open Source Security – Quality Requests", *Open Source Science Journal*, Vol.1, No.1, 2009, pp. 126-135, ISSN 2066-740X.

[5] D. Palaghită, "Quality Characteristics of Open Source Components", *Open Source*

Science Journal, Vol.1, No. 1, 2009, pp. 38-57, ISSN 2066-740X.

[6] I. Ivan, M. Doinea, S. Pavel, S. Vînturis, "Security Management in Distributed IT Applications", The *Proceedings of the 9th International Conference on Informatics in Economy, EDUCATION, RESEARCH & BUSINESS TECHNOLOGIES*, May 7 - 8, 2009, Bucharest, ASE PUBLISHING HOUSE, 2009, pp.803 – 808.

[7] I. Ivan, B. Vintila, C. Ciurea, M. Doinea, "The Modern Development Cycle of Citizen Oriented Applications", *Studies in Informatics and Control*, Vol. 18, No. 3, 2009, ISSN 1220-1766.

[8] A. Grigorovici, I. Ivan, G. Noşca, "Calitatea datelor obţinute în procese de achiziţie", *Journal of Information Technology and Communication Security*, ASE Publishing House, Bucharest, 2008, pp. 127-136, ISBN 978-606-505-137-9.

[9] I. Ivan, G. Noşca, M. Popa, "Managementul calităţii aplicaţiilor informatice", *ASE Publishing House*, 502 pg., Bucharest, 2006, ISBN 9789735947436.

[10] "Reliability, performance and industrial risk" [Online], Available at: http://cfcem.ee.tuiasi.ro

Cătălin Alexandru TĂNASIE, born at 18.08.1984 in Pitesti, Arges, is a graduate of National College I. C. Bratianu of Pitesti and he graduated in 2007 the Faculty of Cybernetics, Statistics and Economic Informatics within the Academy of Economic Studies, the Economic Informatics specialization. Immediately after the faculty, he attended the Informatics Security Master in the same institution, and starting from 2009 is a PhD student at the Doctoral School within the Academy of Economic Studies. He has concerns in the field of distributed applications programming, evolutionary algorithms development, part of the field of artificial intelligence - neural and genetic programming. Currently he works as a information systems designer in a banking institution. He was involved in creating commercial applications using the development platform belonging to leaders in the field, companies like Microsoft and Oracle.

Sorin VÎNTURIŞ has a background in computer science and is interested in security systems related issues. He has graduated the Faculty of Automatics and Computer Science, from the Polytechnics University of Bucharest in June 2007. In March 2009 he has graduated the Informatics Project Management Master, program organized by the Academy of Economic Studies of Bucharest. Currently he is a PhD student at Doctoral School of Bucharest Academy of Economic Studies in the field of Economic Informatics. Other fields of interest include software performance evaluation, data structures, object oriented programming in C++ and Java.

Adrian GRIGOROVICI has a background in computer science and it is interested in data acquisitions. He has graduated the Faculty of Automatic from the Polytechnics University of Bucharest in 1985. In 2009 he has graduated the Informatics Security Master, program organized by the Academy of Economic Studies of Bucharest and, Implementation and Management of Informatics Networks Master, Technology Department of University of Bucharest. Currently he is a PhD student at Doctoral School of Bucharest Academy of Economic Studies in the field of Economic Informatics.