Case Study Analyses of Reliability of Software Application "ePasuria"

Dhuratë Hyseni [1], Betim Cico [2], Isak Shabani [3] ,Bekim Fetaji [4]

[#1, 2, 4] *South East European University, Contemporary Sciences and Technologies, Computer Science*
Ilindenska bb, 1200 Tetovo, FYROM

[#3] *University of Pristina, Faculty of Electrical and Computer Engineering, Pristina, Kosovo*

[1] dh11752@seeu.edu.mk
[2] b.cico@seeu.edu.mk
[3] isak.shabani@uni-pr.edu
[4] b.fetaji@seeu.edu.mk

Abstract— **The focus of the research study is set on analyzes of the reliability of software application, aiming to determine the ways of measurement and determine the parameters of a reliable software application through the case study realized. Measurements of software reliability are important because it can be used to plan and control resources while implementing the software application and offer reliability regarding the correctness of the developed software. Throughout the study we elaborate the analyses of different problems that are encountered in order to maintain a higher level of reliable software application, especially the systems that are more complex and the process of their implementation depends on sensitive data. Furthermore we elaborate ways of detailed analysis and studies in achieving the reliability of software application and researches on the assessment of reliability of the software, and the measurement of the level of the failures in order to realize the level of reliability of a software application.**

Keywords: reliability of software systems, testing methodology, testing failures.

I. INTRODUCTION

The purpose and objective in engineering reliable software applications is to increase the likelihood of the software system to work flawlessly without errors and to be acceptable and trustable by the end users. Therefore, measurements and determining the parameters of a reliable software application are very important.

Software does not change over time unless it is changed by request because of new requirements that impact the reliability of software application which are difficult to achieve due to the complexity of applications that have a tendency to a long period of usage, especially monolithic software. Software application with a high level of complexity that includes multiple users and data manipulation by a greater number of users makes it difficult to achieve a certain level of reliability.

According to [1] in order to increase the reliability of software applications, at each step the support of users should be present, because the reliability of software applications grows continuously. As a result of this we need to set levels of reliability in software applications.

With this we will benefit in higher level of reliability, but we also will not increase the cost for the software application. Removal of all these risks, leads to profitability, as well as numerous benefits which can be used in other similar systems in the market.

With the reliability of software applications we can understand the probability that a software application provide accurate and fast services, for a period of real time based on the environment defined.

II. 2. RELIABILITY MEASURES FOR SOFTWARE APPLICATIONS

In order to determine the exact reliability of a system, it should be based on concept related to the software architecture as defined by [6]. Based on what it needs to identify software applications, it should act in similar circumstances; operate at different points and different times, so failures can be described only in terms of probability.

This section details the analyses of the concept of software reliability.

The discussed concepts provide the basis for the reliability of a qualified software application enabling comparisons between systems that provide an accurate and logical basis for improving the rate of failure, which occurs during the life of that application software as defined by [4].

Specifically, reliability is the probability that a product or a part of that operating system based on the requirements specified for a certain period of time under design conditions (such as temperature, voltage, etc.) to work without failures. In other words, the reliability can be used as a measure of the success of the system to function properly. Reliability is one of the features that consumers demand quality products from manufacturers or rather a tool to assess the security of a software application.

In order to have the design and manufacture of a complete system reliably it is necessary to anticipate the MTTR (Main Time to Repair) as defined by [2] for different conditions that can occur when working with the system. This is generally based on past experiences of designers and experts that are available to handle the repair work. Repair time system consists of two separate intervals of time repair regarding the application software:

- passive time repair
- active time repair

Passive repair time is largely determined by the time taken by the service engineers to travel to work sites users. In many cases, the cost of travel time exceeds the actual cost of repair. Active timer repair of software application is directly affected by system design and is registered as follows according to [8]:

1. Time between the occurrence of a failure of the system and users to become aware of what has happened.
2. Time required to detect a fault and to determine the problem that takes place in this part.
3. Time necessary to replace the remaining components.
4. Time necessary to verify that the problem is corrected and the system is fully functional.

Active repair time can be improved by designing the system correctly so that errors can be detected and corrected quickly. More complex is the design of the application the more difficult will be to correct the mistakes as defined by [2] and [7]. Reliability is a measure that requires the system to a successful operation for a certain time and anticipates the failures or repair that should not be allowed.

Mathematically, reliability D (t) is the probability that a system to succeed in the interval: 0 - t:

$$\mathbf{D(t)} = \mathbf{1} - \mathbf{P(t)} \qquad (\mathbf{1})$$

$$\mathbf{P(t)} = \int_{\mathbf{t}}^{\infty} \frac{\mathbf{1}}{\boldsymbol{\theta}} \mathbf{e}^{-\frac{\mathbf{t}}{\boldsymbol{\theta}}} \mathbf{dt} \qquad \mathbf{t} > \mathbf{0} \qquad (\mathbf{2})$$

t - is a random variable that marks the time of failure.
P (t) - a measure of the uncertainty of failure which is defined as the probability that the system fails at time t.
Increasing the P(t) reduces the reliability of application software, given in the figure below.

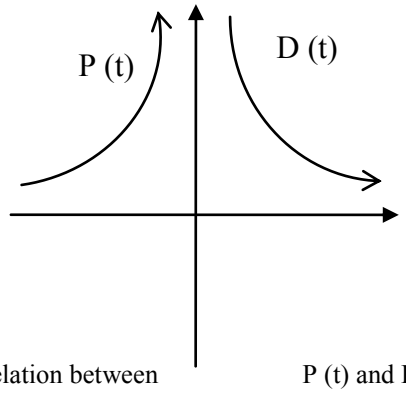

Figure 1. Relation between P (t) and D (t)

III. Reliability of software applications with multiple failures

According to [9] the reliability of software applications should be integrated in the whole software life cycle and should involve all the staff to participate in the project, and it cannot be considered only at the end.

With the classification process the process describes itself as well as its behavior and reduces the failure of software application. The whole process can be optimized by avoiding the introduction of errors in the application of our techniques to process aimed at finding and eliminating errors as it is not possible to implement software without errors, application errors tolerated if the nature of the system requires since it is not possible to ensure that there will be errors remaining in the software as suggested by [8].

The study focuses also in presenting some of the common distribution functions and some dangerous models in reliability engineering software applications. Binomial distribution is used in the dissolution of the reliability of software applications and controlling its effectiveness following the guidelines as suggested by [5] and [6]. It is applicable in reliability engineering, for example when we are dealing with a situation in which an event is present on a success or a failure. Distribution and reliability of software applications is discussed by [8]:

$$\mathbf{P(X = x)} = \binom{\mathbf{n}}{\mathbf{x}} \mathbf{p^x (1-p)^{n-x}} \qquad \mathbf{X = 0, 1, 2, \ldots\ldots\ldots\ldots,}$$

$$\mathbf{n}\binom{\mathbf{n}}{\mathbf{x}} = \frac{\mathbf{n!}}{\mathbf{x!\,(n-x)!}} \qquad (\mathbf{3})$$

n - the number of trials,
x - number of successes,
p - the probability of success trials only
k- reliability factor

Reliability of software applications as function R (k), is given below as defined by [8]:

$$\mathbf{R(k)} = \sum_{\mathbf{x=k}}^{\mathbf{n}} \binom{\mathbf{n}}{\mathbf{x}} \mathbf{p^x (1-p)^{n-x}} \qquad (\mathbf{4})$$

IV. Case study assessment of life cycle reliability for software application "e-Pasuria"

Although in theory the process of assessing the reliability does not seem too complicated, developing safe, complex, and accurate reliability assessment systems requires a lot of effort and research. This is due to the fact that every software product has its own specifics and features, such as the conception, realization and implementation.

After analysis of the case study application identified are several shortcomings. In this part shall jointly identify both problems, because these problems are inter-connected. An enquiry of data in the database is done through functions and further reports are filled with data from lists that are filled by those functions.

Based on work experience in developing software applications it is known that the user is interested in reports that appear as final products of the software applications.

The problem that is identified after the implementation of this application is shown in parts of inquiry. This application is in use by thousands of users simultaneously, it happens that in some institutions the information presented another institution that was the problem in the form of reports created by the "Crystal Reports". This form requires that each report be copied to the folder "Temp". Further, the user sent a copy of the report. Folder "Temp" is loaded at the same time by many reports, which can lead to blockage of the reports or data tampering as discussed in [11].

Below we focus on the module for amortization of the software application “e-Pasuria”, where we examined in details the methodology stages for assessing the reliability of the life cycle.

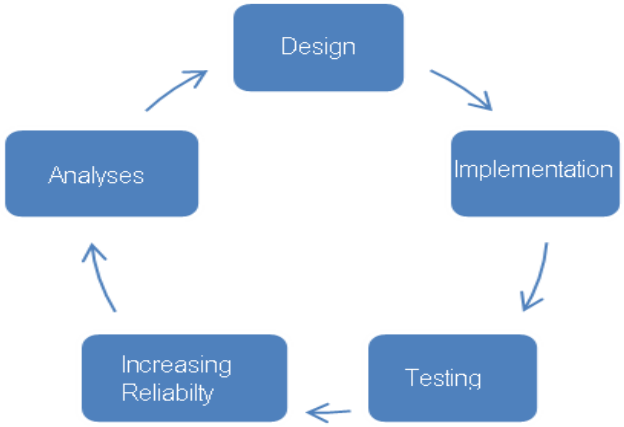

Figure. 2 Methodology for evaluating the reliability over the life cycle of software product

Below we present the steps of the life cycle of the software application (figure 2), and the problems identified:

1. Analysis - in this part of the identification of the problem: the presentation of data that is identified late after being introduced to thousands of data in the system. The other problem is in the form of submission of reports. At this stage it offers logical solution to these problems proposed by engineers.

2. Design - identified problems present the way that should pursue to solve these problems. After testing many functions and sub-functions that call in all the major functions that return slow down data. Decision was taken to establish a procedure that will call only area which is the logo of the institutions of the format "images", and all functions are turned on because this method of procedure that call data is much faster than the functions. This section is enabled by using the tool provided by SQL, "Query Execution Plan Estimated-Display" that allows to be announced for each selection that is located in the procedure, ie the time required by it. For the presentation of data in the report is that the decision to return all reports in the format "Report Viewer Control", which allows converting the content of the report in pdf and further makes sending the file to the user. Another problem is that the application of-Fortune used simultaneously by many users, who generate many reports at the same time, There emerges a lack of "Crystal Reports" that provide great efficiency, and confusion as to become data. There is also an omission in obtaining data from SQL, when using many variables, which are not managed properly by the implementers of application software. But the format "Report Viewer Control" report is sent to the user in PDF format and is unable confusion data for their appearance. For making these decisions is presented using the above method as a model for successful and sustainable applications, which have thousands of users simultaneously. At this stage have become more and testing different scenarios followed guidelines from [10]

3. Implementation - as has been designed, work rules are defined, to be implemented at this stage. This stage should not renege all claims made in the previous step and must be in compliance with all conditions and algorithms which are defined by the design phase. If you have any blockage should review the code and if necessary even re-implementing code.

4. Testing - this part should be realized in the environment provided in which the software application submitted by the developers. More specifically in this phase is done:
   - testing of all cases of research in this application software,
   - comparison of research data as well as many less data
   - accurately set the time which is needed for searching data
   - tested appearances of all reports for all groups of users

To determine the time of failure will base it on the formula (2), where is presented the time of failure of the application software, being carried on the upper step of the evaluation methodology based on reliability regarding the product life cycle software for problem laid down.

The first case is the case of failures and in that case the time of application software failure is greater. Measurement is performed for a time not very long three months analyze (t = 30+31+30 =91 days * 24 = 2184 hours).

There are some measurements made within this interval.

T- is a random variable that marks the time of failure.

Case I (1120 hours):

The formula (2) will have the following values:

$$\mathbf{P(0 < t \leq 2184) = \int_{0}^{2184} \frac{1}{1120} e^{-\frac{t}{1120}} dt \approx 0.86}$$

Then we will refer to the formula (1):

$$\mathbf{D(T) = 1 - 0.86 \approx 0.14}$$

Case II (1500 hours):

The formula (2) will have the following values:

$$\mathbf{P(0 < t \leq 2184) = \int_{0}^{2184} \frac{1}{1500} e^{-\frac{t}{1500}} dt \approx 0.76}$$

Then we will refer to the formula (1):

$$\mathbf{D(T) = 1 - 0.76 \approx 0.23}$$

Case III (2000 hours):

The formula (2) will have the following values:

$$\mathbf{P(0 < t \leq 2184) = \int_{0}^{2184} \frac{1}{2000} e^{-\frac{t}{2000}} dt \approx 0.67}$$

Then we will refer to the formula (1):

$$\mathbf{D(T) = 1 - 0.67 \approx 0.33}$$

In Table 1 and Figure 3 shows the results of the failure time and reliability for a certain time on the problem set out above.

TABLE I
The first case of failures and achievements - credibility with the initial version

Case	P(t)	D(t)
I (1120 hours)	0.86	0.14
II(1500 hours)	0.76	0.23
III(2000 hours)	0.67	0.33

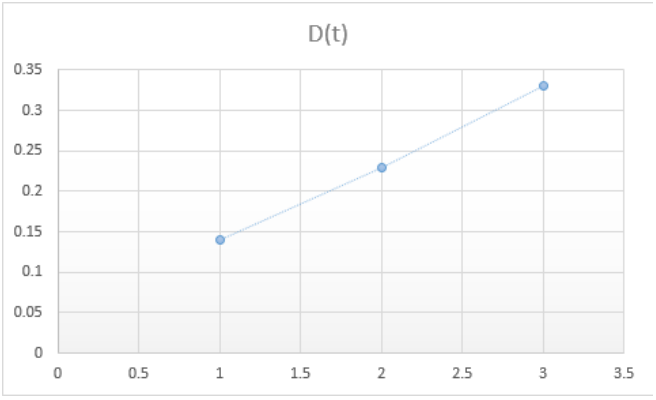

Figure. 3 The first case of failures and achievements - credibility with the original version

These measurements were made for 3 (three) months after implementation of the system with the necessary changes.
There were measurements made within this interval as in this Table I, Figure 3, but in these cases the analysis starts from the first three months and the days of the second three months are added for the analysis.
After implementing the changes version is observed that the credibility of the three cases has increased significantly, Table II:

Case I (3880 hours):

The formula (2) will have the following values:

$$\mathbf{P(2184 < t \leq 4380) = \int_{2184}^{4380} \frac{1}{3880} e^{-\frac{t}{3880}} dt \approx 0.25}$$

Then we will refer to the formula (1):

$$\mathbf{D(t) = 1 - 0.25 \approx 0.75}$$

Case II (4000 hours):

The formula (2) will have the following values:

$$\mathbf{P(2184 < t \leq 4380) = \int_{2184}^{4380} \frac{1}{4000} e^{-\frac{t}{4000}} dt \approx 0.23}$$

Then we will refer to the formula (1):

$$\mathbf{D(t) = 1 - 0.23 \approx 0.77}$$

Case III (4300 hours):

The formula (2) will have the following values:

$$\mathbf{P(2184 < t \leq 4380) = \int_{2184}^{4380} \frac{1}{4300} e^{-\frac{t}{4300}} dt \approx 0.2}$$

Then we will refer to the formula (1):

$$\mathbf{D(t) = 1 - 0.2 \approx 0.8}$$

After these results, we can say that this problem is associated with increased application reliability and user-Property to recent Figures. 4 and Table II

TABLE III
The first case of failures and achievements - confidence in the version amended

Case	P(t)	D(t)
I (3880 hours)	0.25	0.75
II (4000 hours)	0.23	0.77
III(4300 hours)	0.2	0.80

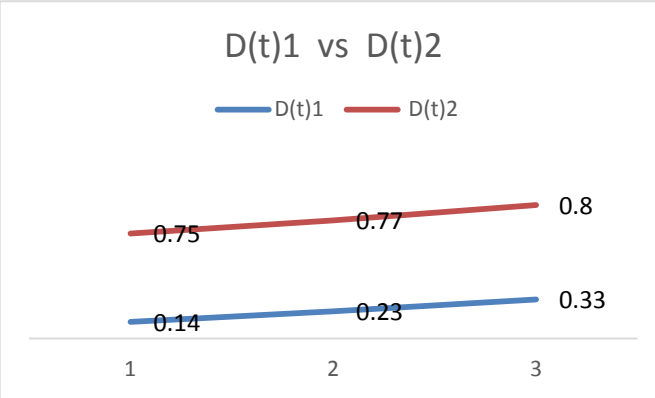

Figure 4. The first case of failures and achievements credibility by comparison analyses

.
This measurements for reliability of the software as a whole-property is roughly 0.80, which is a high value.

TABLE IIIII
Statistics from 01/01/2009 to 28/05/2013 in separate years

Year	Invoice	Entry	Out	Downloads	Transfers	Wealth	Users
2009	4877	310178	115723	231	81	27176	3254
2010	16272	1413902	1074904	1403	585	33278	7965
2011	40380	3555577	3693563	4709	23373	36149	5066
2012	56136	9054189	13699757	8729	44144	39740	3042
2013	13438	1530877	3590858	3757	7974	8848	462

.
The table presents all transactions carried out through the application of Real-years:
Based on Table 3. Can be concluded that transactions have increased over the years and in 2013 the statistics are valid only for the first five months of this year.
There also should be emphasized the registry of invoices where in institutions is made at the end of every three months and therefore the values are lower than the previous year, it means that these values apply only for the first three months because for the second three months the records are not completed yet.
This table can also be seen that the reliability of the application as the user only has increased thanks to the measures taken for the application. [10]

V. Conclusion

The study focuses on the reliability of software applications, and tries to offer insight regarding the analyses and measurement of reliability parameters of trustable software based on the described case study.

The concrete case study is focused on research in the field of management of specific cases in the life cycle of software application development and the course of reviewing these cases based on application software of “ePasuria”, as part of these systems, providing services to thousands of users simultaneously.

The paper provides insights and information for all those who are interested in obtaining more extensive knowledge about testing the reliability of software applications to end users, in determining the time of failure and reliability were measurements are performed for a period of 3 months. The study can also serve as starting point for further research in this field.

References

[1] [1] Bailey, D., E. Frank-Schultz, P. Lindeque & J. L. Temple III. “Three reliability engineering techniques and their application to evaluating the availability of IT systems: An introduction.”, 2008, pp. 577- 589
[2] MTBF, MTTR, MTTF & FIT Explanation of Terms, Susan Stanley
[3] Eduardo Valido-Cabrera, Software reliability methods, 2006
[4] Gokhale, S.S. & K.S. Trivedi. “Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework.”, 2006, pp 578-590
[5] Goševa-Popstojanova, K., M. Hamill & R. Perugupalli. Large empirical case study of architecture-based software reliability, 2005
[6] Holger Hellebro, Architecture-Based Reliability Modelling of Software Applications, 2009
[7] Hung-Hua Lo, Chin-Yu Huang, Sy-Yen Kuo, and Michael R. Lyu, Sensitivity Analysis of Software

Reliability for Component-Based Software Applications, 2008
[8] Hoang Pham, SystemSoftware Reliability, 2006, pp 15-32, pp 334-365
[9] John Musa, How testers can use a software reliability engineering maturity model, 1999
[10] e-Pasuria. Assets Management System. DOI=https://e-pasuria.rks-gov.net
[11] Sabedin A. Meha, Agni H. Dika, Isak R. Shabani, Improving performance of a web based software application, 2011