The Increase of the Web Application Reliability to the End Users

Isak Shabani
University of Prishtina,
St. Bregu i Diellit w/n
Pristina 10000, Kosovo
+377 (0) 44 979 318
isak.shabani@uni-pr.edu

Betim Çiço
Faculty of Contemporary Sciences and Technologies, South East European University,
St. Ilindenska w/n
Tetovo 1200,
FYR of Macedonia
+355 682 094 229
b.cico@seeu.edu.mk

Dhuratë Hyseni
Faculty of Contemporary Sciences and Technologies, South East European University,
St. Ilindenska w/n
Tetovo 1200,
FYR of Macedonia
+377 (0) 44 202 109
dh11752@seeu.edu.mk

Fatos Halilaj
University of Prishtina,
St. Bregu i Diellit w/n
Pristina 10000, Kosovo
+377 (0) 44 451 592
fatos.halilaj@uni-pr.edu

ABSTRACT
In the Kosovo institutions the e-Government is being implemented. Regarding it, a web based assets management system is developed and it is in use from 2009. This paper provides concrete results on the reliability of software applications. As a sample many public institutions are considered, where the assets management system is being used from the beginning of 2009 to the end of 2012. During the extractions of these results relating reliability, mathematical methods are used and for the increase of reliability, a synchronization algorithm is provided, which allow for the application to operate in the absence of the network. Provided methods have increased reliability of the software application to the end-users.

Keywords
Web Application, Software Reliability, e-Pasuria, e-Government

1. INTRODUCTION
Often, when multi-user systems are developed to be used nation-wide by the government and citizens, reliability of the software should be taken under high considerations. Involvement of the end-users in this process is very valuable. The aim of the reliability is to make a system reliable and consistent with no-faults to the end-users. Thus, measurements and information of a reliable software application are very important. For every complex system that includes multi-users and data-manipulations, it will be really hard to achieve a consistent level of reliability. Detailed study and analysis and research on software application reliability is provided in this paper.

2. SOFTWARE RELIABILITY MEASUREMENTS
Reliability $R(t)$ of a system at time t is the probability that the system operates without failure in the interval $[0, t]$, given that the system was performing correct at time 0 [7]. High reliability is required in situations when a system is expected to operate without interruptions.

Reliability is a function of time. The way in which time is specified varies considerably depending on the nature of the system under consideration. For example when a system is required to do some job in a short period of time, time is specified in units such as minutes/seconds/milliseconds.

To exactly define system reliability we should rely on the concepts related to software application. Based on these software application needs we act on the same way, by operating in different point and in different time; this way the failure can be expressed only in terms of probability. Thus, fundamental definition for the software reliability depends on concepts of the probability theory. These concepts, provide basics of a software reliability, allow comparisons among systems and they provide fundamental logic for improvements of the fail rates, that will be reached during the application life cycle.

In general, a system may be required to perform different functions, each of which can have different reliability level.

In addition, in different time, software application can have different probability to perform required functions from the user under declared conditions.

Reliability represents probability of a success or possibility, that the software application has to perform it's functionality for the sake of the project under certain limits. More specifically, reliability is a probability that the product or part of it that operates in basis of predefined requirements for a defined period of time, under projects' conditions (i.e. number of transactions, bandwidth, etc.) works with no failure. In other words, reliability can be used as a measurement of a system's success for it to work as required. Reliability is a quality characteristic that customers demand from the producer of the product or better said a tool to evaluate safety of a software application.

Mathematically, reliability $R(K)$ is the probability that a system be successful in the time interval: [0-k]:

$$R(K) = 1 - P(K) \quad (1)$$

$$R(t) = \int_t^{\infty} \frac{1}{\theta} e^{-\frac{s}{\theta}} ds = e^{-\frac{t}{\theta}} \qquad t >= 0 \qquad (2)$$

The increase of R(k) decreases software application reliability (see Figure 1).

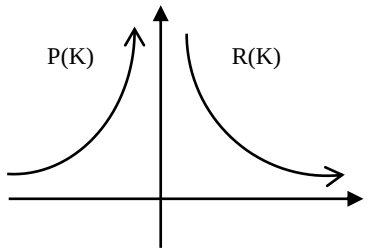

Figure 1. Report between P(K) and R(K)

In order to have a complete reliability in a system, the prediction of mean time to repair (MTTR) is required for different conditions that can occur during system use. This is based from the system designers' experience in the past and experts' disposal for repair handlings.

Time system repair is composed of two specific intervals:

- Passive time for software application repair
- Active time for software application repair

Passive repair time is defined mainly from the time taken from the engineers that travel to the users' locations. In many cases, traveling time cost is exceeded by actual repair time cost. Active time for software application repair is related directly with system projection and is defined as follows:
Time between occurrences of a failure and user of the system is notified about it:

- Time required to explore the failure
- Time required to change the components
- Time required to verify that the problem has been solved and the system is fully functional.

Active time or software application repair can be correctly improved if the system is designed in such a way that errors can be identified easily and be corrected. The more complex the design of the system is made, the harder to find the errors and improve things in a system. Reliability is a measure that requires success for the system for a particular period of time and such as that failures are not allowed.

Availability $A(t)$ of a system at time t is the probability that the system will be functioning at the instant of time t.

$A(t)$ is also referred as point availability, or instantaneous availability. It is often necessary to determine the interval or mission available.

$$A(T) = \frac{1}{T} \int_0^T A(t)dt \qquad (3)$$

$A(T)$ is the value of the point availability averaged over some interval of time T. This is the time required for the system to accomplish some task [7].

If a system cannot be repaired, the point availability $A(t)$ equals to the system's reliability, i.e. the probability that the system did not fail in between 0 and t. Thus, as T goes to infinity, the steady-state available of a non-repairable system goes to zero.

$$A(\infty) = 0$$

Steady state available is often specified in terms of downtime per year. Figure 2 shows the values for the availability and the corresponding downtime.

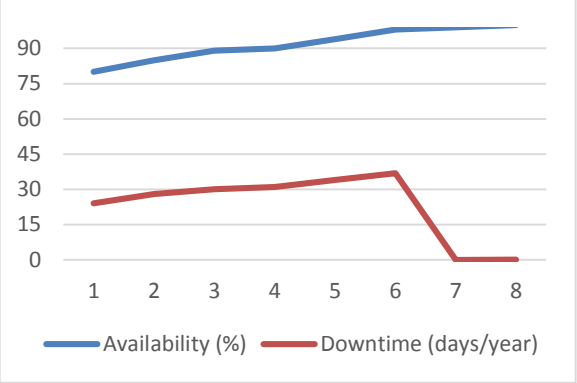

Figure 2. Availability and corresponding downtime values per year

Availability is typically used as a measure for systems where short interruptions can be tolerated. Surveys have shown that nearly 60 percent of web users say that they expect a website to load in less than 3 seconds.

3. SOFTWARE RELIABILITY WITH MANY FAILURES

Reliability should be integrated in the whole cycle, it should include all the staff that take part in the project. It cannot be considered only at the end, but technical state should be achieved to improve software reliability that are included. All of these techniques are classified in three different groups (see Figure 3).

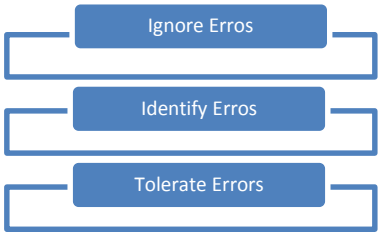

Figure 3. Group of failure's classification in software reliability

With classification a process can be described and with it we can lower the failures to the software as we suggested earlier. All of the process can be optimized from errors, the goal of the application of our technique for this process aims error identification and avoiding them. Since it is very hard to implement a software that is error free, application of tolerated errors if the system nature requires it since we are never sure that there will not be errors left in a software. This section represents some of the common distributions and some risky models in software reliability. Binominal distribution is mostly used in the case of reliability distribution and in the control of performance. It is applicable in software engineering, i.e. in a situation where an event is present in a success or failure. Software reliability distribution is given with the following formula:

3.1.1 $\mathbf{P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}} \qquad \mathbf{X = 0, 1, 2, \ldots}$

3.1.2 $\mathbf{n \binom{n}{x} = \frac{n!}{x!(n-x)!}} \qquad (4)$

Where, n – number of judgment, x – number of successes, p – probability of judgment of a single success.

Software reliability function R (k), (namely. at least k is given by the software and n are used software applications) given as [6]:

$$\mathbf{R(k) = \sum_{x=k}^{n} \left(\frac{n}{x}\right) p^x (1-p)^{n-x}} \qquad (5)$$

4. PRACTICAL EVALUATION OF RELIABILITY RELYING e-PASURIA

Even though theoretical process of reliability evaluation doesn't seem to be very complex, safe development, complex, and exact for the reliability evaluation systems requires a lot of effort and research. This for the fact that each product has its own properties, in concept, development and implementation. In the following we will examine the module of amortization in the system e-Pasuria, in details we will show the methodologies of reliability evaluation relating the application life-cycle. Mathematically, part of the failures are shown, all of these statistics will be presented in a tabular and graphical form.

Relating the formula (2) for the first three months (2190 days):

$$P(0 < T \leq 2190) = \int_0^{2190} \frac{1}{2022} e^{-\frac{t}{2022}} dt \approx 0.334$$

Then from the formula (1):

$$D(T) = 1 - P(T) = 1 - 0.334 = 0.67$$

Where D(T) is the reliability of the software, whereas P(T) is time of failure for the software which is determined based on methodologies of evaluation of reliability relating the products' life-cycle (for the problem of amortization which is very sensitive). Main goal is the reliability of amortization of inventory accurately for each equipment. First case is presented in table 1 (see Figure 4).

Table 1. First case with failure and reliability achievement against first version of the software

P(T)	D(T)
0.33	0.67
0.42	0.58
0.38	0.62

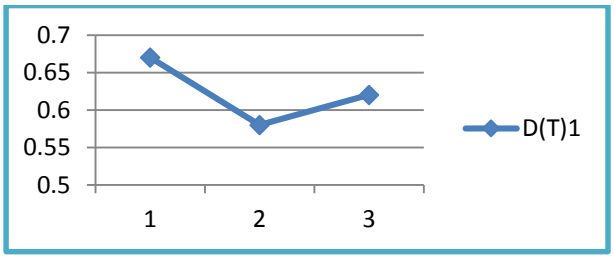

Figure 4. Case with failure and reliability achievement against first version of the software application.

After this, an upgrade (update) to the current version has been provided with amortization functionality provided allowing the user to automatically calculate the amount of amortization for the equipment based on categories provided in the system. Before the update, user could calculate the amount of amortization on monthly bases, and this was not accepted very well from officials/users because it did not fulfill the needs required by audits. Audit, required that the calculation of amortization be rectilinear, and should be provided in days, months, and years, which is now provided with the update. Initial results for the achieved reliability are too low.

Now we present the case when equipment amortization is well accepted by users and it is closely or better said very likely with rectilinear method. Now, we give statistics which are taken for the period of six months, after the release of the update to the software.

$$\mathbf{P(2190 < T \leq 4380) = \int_{2190}^{4380} \frac{1}{2211} e^{-\frac{t}{2211}} dt \approx 0.10}$$

4.1.1 D (T) = 1 - 0.1 = 0.90

Below, calculation of the reliability of the system after update are provided, Table 2, Figure 5:

Table 2. First case with failure and reliability achievement after the updates

P(T)	D(T)
0.10	0.90
0.04	0.96
0.01	0.99

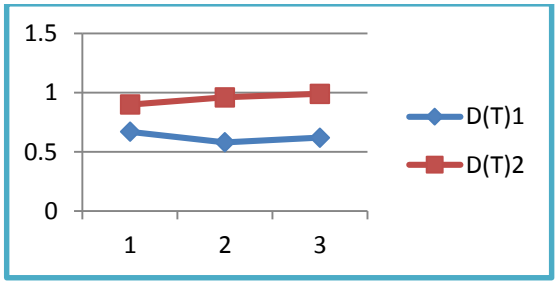

Figure 5. Case with failures and increase of reliability by comparison

As is seen for the graphics (see Figure 5) with the changes made on the update release, based on the methodologies for reliability evaluation of software life-cycle, reliability has been achieved [6].

In the following, we provide software reliability of the sysetem e-Pasuria, relying the formula (4):

- Regular time for functioning is taken as 90% of full time
- Successfull time is evaluated as an average of all values from different modules
- An average of 95% out of 100% resulted successful in e-Pasuria software:

$$\mathbf{R(90) = \sum_{90}^{100} \left(\frac{100}{90}\right) (0.91)^{91} (0.05)^{91-5} \approx 0.877}$$

It can be said that reliability of e-Pasuria software in general is aproximately ~0.90, which indicates a high value of reliability.

5. DATA SYNCHRONIZATION

Optimistic replication strategies are attractive in a growing range of settings where weak consistency guarantees can be accepted in return for higher availability and the ability to update data while disconnected [1]. These uncoordinated updates must later be synchronized (or reconciled) by automatically combining non conflict updates, while detecting and reporting conflict updates. The ability to support mobile and remote workers is becoming more and more important for organizations every day. It is critical that organizations ensure users have access to the same information they have when they are in the office. In most cases, these workers will have some sort of laptop, office desktop, Smartphone, or PDA. From these devices, users may be able to access their data directly through VPN connections, Web Servers, or some other connectivity method into the corporate networks. Synchronization gained great importance in modern applications and allows mobility in the context of information technology. Users are not limited to one computer any more, but can take their data with them on a laptop.

5.1 Data Synchronization Algorithm

In Kosovo there are still problems with infrastructure reconstruction. Main problems are: non-regular power supply, connection drops, server drops, which present a big problem in software applications. Such problems cause activity interruption at work and inability to do the service on time, which reflects the service to the students. Situations of this nature; cause skepticism to users, personnel and management in use of IT services. If there's connection, e-Pasuria (assets management system) works parallel online and offline, which means the data are transferred in both local and central databases. If the connection is lost, e-Pasuria works with local server, which means that new data are being saved in local server which are not in the central server and in this case the synchronization component should synchronize the data with the data center when connection is present. Web Service has the information on how data should be synchronized through the columns in the tables.

Considering those aspects and for the continuity of this project, a solution to minimize the problems should be found. For this purpose an offline mode is used, realized with the design of the algorithm used for data synchronization based on Web Services [5], as shown in Figure 4.

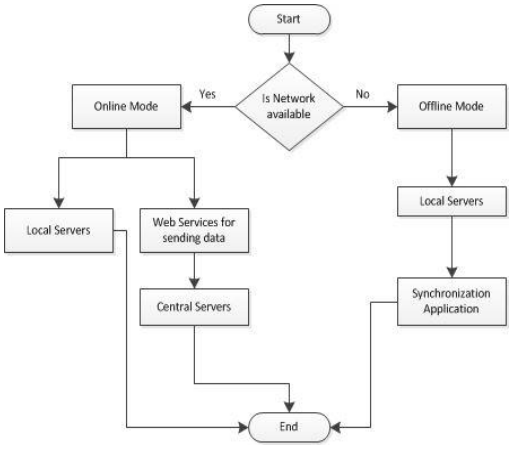

Figure 6. Algorithm for data synchronization in e-Pasuria

6. ASSETS MANAGEMENT SYSTEM

6.1 The Concept of the System

e-Pasuria is an Asset Management System which is in used by all Ministries, Municipalities, Agencies in Kosovo. Through this application monitoring and controlling from the auditing agency is done for all integrated institutions in Kosovo. e-Pasuria provides the management of assets, their usage, expandible materials, equipment barcoding, amortization of equipments based on amortization percentage provided with categories, stock management, and online requests for officials when they require the usage of assets (see Figure 7).

Figure 7. Online request in e-Pasuria

Application also supports internal and external transfers. By internal transactions, assets are transferred internally to the employees inside the institutions, and by external it is possible to transfer assets from one institution to another (i.e. Ministry of Public Administration (MAP), donates 20 Computers to the Ministry of Science and Education (MSE), so assets from MAP warehouse are transferred to MSE warehouse). With the use of this integrated software it is possible to track government assets on all stages of their life-cycle (i.e. from the day you buy them, to the last day of their usage).

As this is a role-based system, it provides the following user roles:

- Administrator – *top level administrator*
- SubAdministrator – *administrator for a particular institution*
- Warehouse official – *In charge for warehouse operations*
- Logistics official – *in charge for internal transactions, and internal asset tracking*
- *Property official* – in charge for request approval, external transfers from one institution to another, equipment amortization and financial reporting
- *Personnel official* - Human resource management operations
- *Ordinary personnel* – every employee, can make request for assets, request tracking, reports on their load of assets.
- *Auditor* - full audits of assets for a particular institution
- *General Auditor* – full auditing for all institutions that use e-Pasuria.

6.2 Statistics of Data Transfer in e-Pasuria

Statistics with real data usage for the transactions made by users in e-Pasuria are given as follows, Figure 8:

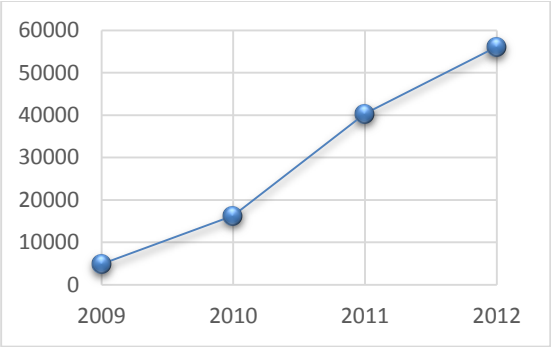

Figure 8. Number of total invoices registered in e-Pasuria

In the following (see Figure 9), we provide number of transactions per year (2009 – 2012):

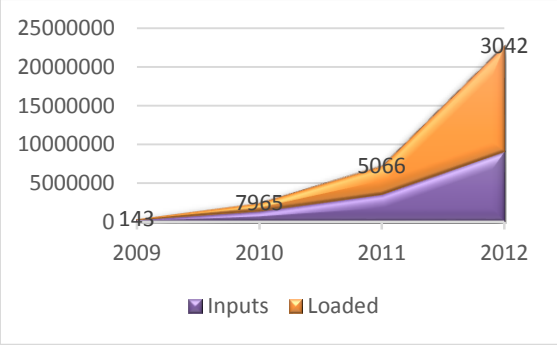

Figure 9. User transactions per year in e-Pasuria

As seen in Figure 9, number of transaction is greatly increased from 2009 – 2012. With increased reliability of such a scale in the system, number of integrated institutions is also increased [8].

As number of users is of a great value in the usage of the system, we provide statistics on number of users per year (see Figure 8):

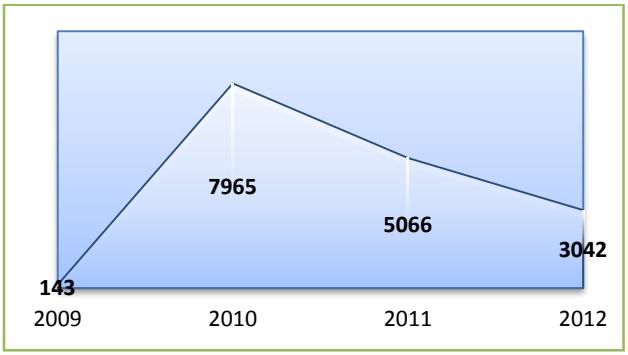

Figure 10. Number of registered users per year in e-Pasuria.

As can be seen on Figure 10, number of new users registered in e-Pasuria has begun to lower, this is because of the total number of integrated institutions from 2009 to 2011, as current number of unintegrated institutions is less than 10 (this includes only agencies, and some municipalities that are not yet integrated because of their infrastructure and network which makes it impossible for the moment).

6.3 Technologies used to develop e-Pasuria

Application is built using Microsoft Technologies using Microsoft Visual Studio as a Development Environment, Microsoft ASP.NET, SQL Server 2008, ADO.NET, and Communication with other applications using Web Services and XML.

7. CONCLUSION

It is very hard to calculate the exact reliability of software, but to come to those results we have included a bunch of factors involved with this process, such as availability of the software, number of transactions, users, downtime and we have shown mathematical methods on how to increase reliability on the software application. As we were unable to show in details the reliability provided in each module for the Assets Management System (e-Pasuria) we leave a lot uncovered and for further researches in the future.

8. ACKNOWLEDGMENTS

We are grateful to all the people working at Kosovo Government that supported us on gathering of information and measurements on e-Pasuria software to make this research possible.

9. REFERENCES

[1] Shabani, I., Çiço B., and Dika A. 2012. Solving Problems in Software Applications through Data Synchronization in Case of Absence of the Network. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 3, January 2012, ISSN (Online): 1694-0814. Port-Louis, Mauritius.

[2] Mauher, M., and Smokvina, V. 2008. Municipal Asset and Property Management System for the Web Collaborative Environment. Multidimensional Management Consulting, Ltd. Zagreb Vinogradi 36 C, Zagreb, Croatia & City of RijekaKorzo 16, Rijeka, Croatia

DOI=http://www.majorcities.eu/generaldocuments/pdf/municipal_asset_and_property_management_system_for_the_web_collaborative_environment.pdf

[3] Çiço, B., Hajdini, A,. Sadiku, S., Meha, S., and Shabani, I. 2011.The Increase of the Speed of Integration of Online Services for Citizens Through Standardization of Municipality Portals". 15th International research/expert conference) Czech, TMT.

[4] Kreutzkamp, J. Hagge, L. Deffur, E. Gellrich, A., and Schulz. B. 2003. Experience with an IT Asset Management System. DESY, Hamburg, Germany (Computing in High Energy and Nuclear Physics, 24-28 March 2003, La Jolla, California). DOI=http://www.slac.stanford.edu/econf/C0303241/proc/papers/TUDT001.PDF

[5] Shabani, I., Çiço B., and Dika A. 2011.Web services oriented approach for data synchronization. 6th South East European Doctoral Student Conference, Greece.

[6] Bailey, D. Frank-Schultz, P. Lindeque, J. Temple III. Three reliability engineering techniques and their application to evaluating the availability of IT systems: An introduction.

[7] Dubrova, E., 2007.Fault Tolerant Design. An Introduction Department of Microelectronics and Information Technology Royal Institute of Technology. Stockholm, Sweden.

[8] e-Pasuria. Assets Management System. DOI=https://e-pasuria.rks-gov.net