

Quality Related Costs of e-Business Systems

Paul POCATILU

Academy of Economic Studies, Bucharest, Romania
ppaul@ase.ro

Abstract: *Developing e-business applications involve many costs that have to be known before the beginning of this process. Software reliability is an important characteristic of the software quality. Software reliability is clearly shown after a software application has been released. The paper presents the role of software testing in increasing the reliability of the software applications, starting with identifying the main causes of non-reliability. In this paper are presented the main categories of costs identified during the development of e-business applications. Those costs depend on many factors that have to be identified too. Removing errors from programs - errors identified through an adequate testing – is leading to an increased reliability for software.*

Keywords: *e-business systems, collaborative systems, quality costs, faults, failure, software testing, reliability.*

1. Introduction

One of the most important characteristics of software quality is reliability. Software reliability is the probability that a software application will work without failure as is provided in specifications.

As is presented in [1], failure is the nonconformance with specifications. Failures are generated by faults that exist in programs.

Software testing, the process of finding errors in programs, is a very important activity in the software development cycle. The cost of software testing is very high; sometime it could be 30% from the cost of the project [1], [2], [7].

Software development cycle, environment, and hardware issues are the main causes for unreliable software. Software development cycle issues are the most important cause of the unreliability. Poor specifications, inadequate activities of design, implementation, testing and debugging can lead to unreliable software applications.

Environment issues are generated by software incompatibilities between the real environment where the application is running, and the environment that was described by specifications. Hardware issues can be the result of incompatibilities between different hardware components of the computer on which the application is used.

Today there are many companies that use the e-commerce in various fields like direct marketing, selling, clients services, banking services etc. E-business software includes components for safely payments using credit cards, components for transactions security, components for the presentation level and other components. Depending on the nature of the application, some components of the architecture might not be present, or some specific component can be added (like payment server).

The reliability of the Internet applications should be very high considering the great number of users that access them. The development life cycle of e-business applications is almost the same as the classical applications. The most important phases are analysis, design, coding, testing and debugging and the implementation and maintenance.

Today, the development of applications based on Internet need to be made rapidly, so the time allocated for the testing process is shortened. This could lead to poor quality of the e-business applications, but using automated testing tools and other verification activities, the testing process will succeed.

2. E-business applications as collaborative systems

The collaborative systems are developed based on a set of specifications that were defined in the analysis stage in order to establish the goals for the development process. The system must behave and must offer the results that the agents want and that they have established at the start.

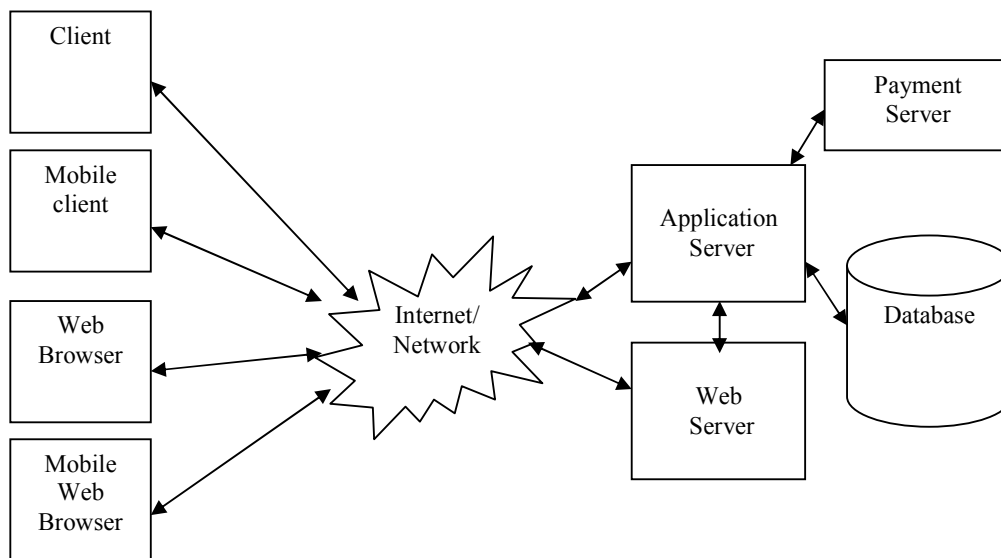


Fig. 1. Architecture of a collaborative system [3]

Collaborative software architecture is based on distributed systems. That includes a server application, client application and a database server. The client application could be Web based or a rich client, a mobile or desktop client. Figure 1 depicts a general architecture of a collaborative system. It includes all types of clients and servers. The network includes wired and wireless transmission medium.

E-business systems are a particular case of collaborative systems. They are based on Web architecture that confer them a high reliability, scalability and flexibility. The Web architecture is different from the n-tier architecture by two aspects:

- the client applications has a reduced complexity (it is a simple Web browser)
- the e-business applications rules level is based on components, and there is not a single system that implants the whole logic.

The client components are graphical user interface and runs in Internet browsers. The server components that run into an application server provide the business logic. The e-business applications architecture has the following components (figure 2):

- web server;
- clients (browsers);
- database server;
- network connections;
- application server (optional);

- payment server (optional).

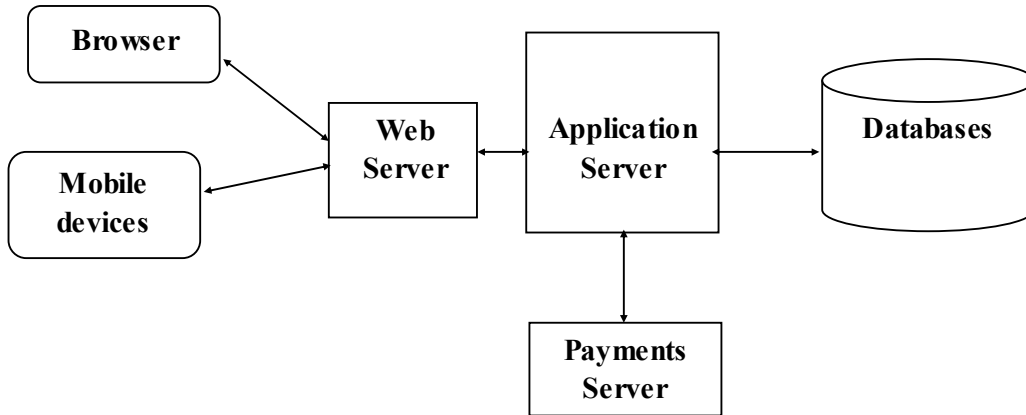


Fig. 2. The e-business systems architecture

E-commerce represents all software and commercial processes required for business processes to run using only or mainly digitally data streams. E-commerce involves using the Internet, digital communications and software applications in the buying-selling processes, it being a subset of e-business process [5].

In order to maintain the Internet applications, specialized personnel is required like Web, database and payment servers administrators, along to personnel involved in the development process (analysts, programmers, testers).

Technological standards and also standards for e-business processes are used for the development of e-business applications.

3. Software testing and reliability

Every program has an associated tree structure, with an initial node, intermediate nodes and terminal nodes or leaves. One node from the graph corresponds to a statement, to a sequence of statements or a function.

As an example, considering the function *Maximum* that has the following code:

```

int Maximum (int a, int b, int c)
{
    int max;
    max = a;

    if (max < b)
        max = b;

    if(max < c)
        max = c;

    return max;
}
  
```

The associated tree structure of the *Maximum* function is given in figure 3 [6]. Complete testing involves exercising all leaves of the tree. If there are k leaves untested, the probability to have program failures depends on the value of this number.

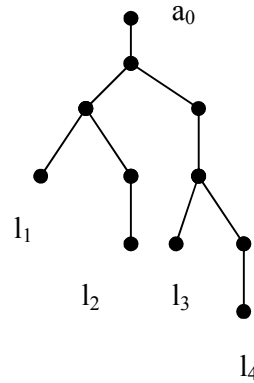


Fig. 3. Tree structure associated to the *Maximum* function

Suppose that there is a fault on path $a_0 - l_2$ that had not been tested and the execution of this path will lead to program failure. When the user introduces data that exercise this path, the program will fail. The number of errors discovered during the testing process is recorded as in table 1. After each period of time, corrections are made.

Table 1. Error logging

Time	Number of identified errors
t_0	k_0
t_1	k_1
...	...
t_n	k_n

There are techniques to estimate the number of errors remaining in programs. One of them is *error seeding*. Error seeding involves insertion of faults in programs in order to estimate how many errors would remain undiscovered after testing process. This technique is helpful in estimating how many faults still exist in programs after the testing is done.

The reliability of a program is computed with the following formula:

$$R = \frac{N_S}{N_T} = 1 - \frac{N_U}{N_T},$$

where:

N_S – number of successful runs

N_U – number of unsuccessful runs

N_T – total number of runs.

Depending on the number of errors remained in programs the number of successful runs can be high or low. If there are many undiscovered errors, the probability to activate a fault is high.

In figure 4 is shown how some user actions and inputs can cause activation of existing faults in programs. This will lead to a failure of the programs, and to an unsuccessful run.

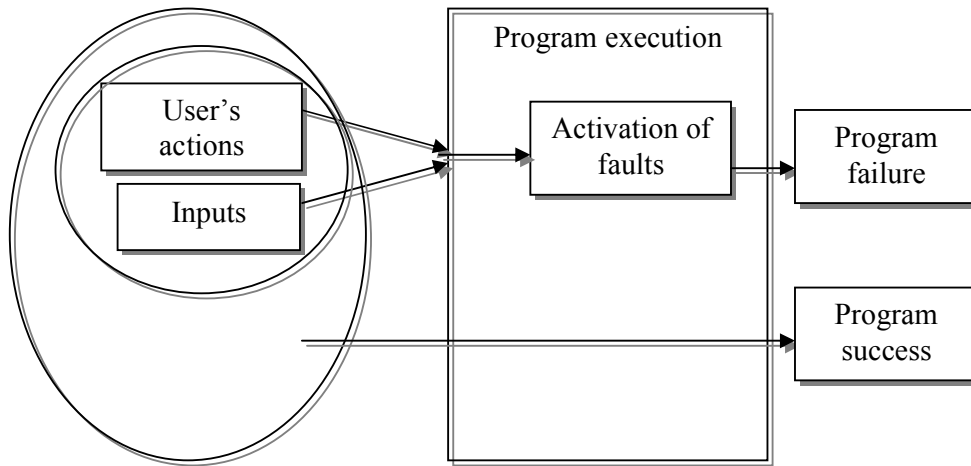


Fig. 4. Results of programs execution

The result of a program's execution (success, failure) depends mainly on the number of errors that still exist in the program and on the user actions and inputs that are given.

4. Cost analysis

Important questions that arise are: “when to stop testing?” and “when to release the software?” [2]

It is very important to choose an optimum release date for the software, because the testing is very expensive after the software has been released. The costs of testing might be multiplied hundreds of times after the release of the software than in earlier phases of development cycle. As the testing advances, the cost of releasing unreliable software decreases. This is depicted in figure 5.

Reliability depends on testing. After a certain level of reliability is reached, continuing testing increases the reliability only with a small percentage.

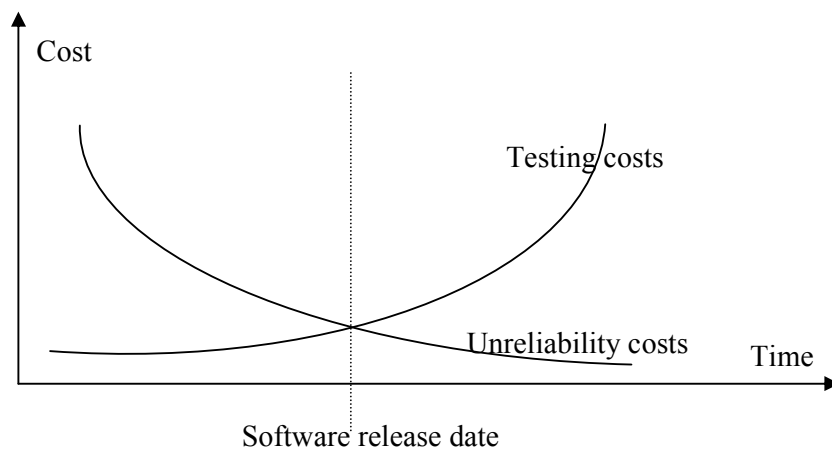


Fig. 5. The costs of software testing and reliability

In order to improve the testing process and decrease the costs associated with it, the automation of testing is an efficient solution even it is almost impossible to automate the entire testing process.

The main factors that influence the e-business applications development are:

- application complexity;
- application size;
- development technologies used;
- personnel qualifications and experience;
- the certification level of the company;
- number of users expected.

The *application complexity* is one of the most important factors that influence the costs. As the application complexity is higher, the required manpower is increasing. There are many ways to compute the complexity of an application. The most used is the McCabe complexity metric based on cyclomatic number, computed as the number of decision blocks plus one. If the McCabe complexity is over 10, the complexity can be reduced by splitting the program in two or more parts with a lower complexity.

Another complexity metric is the Halstead complexity, based on the number of operand operators from a program.

The *size of an application* is given by the LOC (Lines of Code) or by the PF (Point Functions). LOC does not count comments or documentation lines. This measure is used as KLOC (Kilo LOC), thousand of source code lines. Measuring the software size based on the LOC is dependent of the implementation language. In [1] is shown the correspondence between the number of source code lines of different programming languages, table 2.

Table 2. LOC correspondence for different programming languages

Programming language	Source lines
Assembler	320
C	128
COBOL	106
Pascal	90
C++/Java	64
Visual Basic	32
SQL	12

The function points are an indirect measure of the software and its development process. Function points are focused on the functionality of the utility of the application and are independent of the programming language used to develop the software.

The *development technologies* that are used to build the application influence its costs. Modern tools and integrated development environments allow designing and building user interfaces very fast by using wizards and templates through the code reusing. There are many technologies that can be used, based on various platforms, starting with the analysis and ending with the implementation.

Personnel qualifications and experience and the *certification level of the company* have a big impact on the costs of the applications.

The *number of user* expected influences the way how the application is build. When are expected many user, the development of the application needs more effort, especially on the verification and quality side.

5. Cost models

There are many models that can be used to compute the e-business applications costs. The models can be linear and nonlinear and they can take into account many factors that contribute to the final cost of the applications.

The cost model presented here is based on the effort used to develop an application during the development cycle. In order to compute the cost of the e-business applications, for each application are recorded data into a matrix, as given in table 3.

Table 3. The effort distribution during the development phases

Development phase \ Effort	Analysis	Design	Coding	Testing and debugging	Other activities	Total
Manpower						<i>MC</i>
Software						<i>SC</i>
Hardware						<i>HC</i>
Other						<i>OC</i>
<i>Total</i>	<i>AC</i>	<i>DC</i>	<i>CC</i>	<i>TDC</i>	<i>OAC</i>	<i>CEB</i>

In table 2 are identified the following costs:

- CEB – the cost of the e-business application
- MC – manpower costs
- SC – software costs
- HC – hardware costs
- OC – other costs
- AC – analysis costs
- DC – design costs
- CC – coding costs
- TDC – testing and debugging costs
- OAC – other activities costs

The *manpower cost* is the most important [4], having the main contribution to the final cost, and it includes salaries and other expenses related to salaries. *Software and hardware costs* usually are fixed costs, and include the costs of renting or buying software and hardware systems used to develop and to test the applications. *Other costs* are composed by variables and fixed costs that are not related to manpower, software and hardware costs, and they are included in the final price of the application.

The *cost of analysis, requirements, design and coding* is mostly compounded by the manpower costs and is the effort recorded during the correspondent development phases of the application.

Software testing is a very expensive process. The *cost of testing* e-business applications is bigger than the cost of testing classical software. It requires a supplementary effort due to:

- testing the application server;
- testing the Web server;
- testing the transactions.

Integration testing costs are also bigger than the integration costs for the classical applications. For the Internet applications there are many combinations of elements that have to be integrated and tested.

The total cost of testing Internet applications is given by the sum of all testing activities. There are also some overhead costs. The main cost category is the personnel's

salaries. Other costs include the costs of the tools and hardware used in testing. The main costs of software testing are described in [4].

Based on the existent technologies on the market, the cost of testing Internet application can be reduced by automation of this process. Another way to reduce the costs of software testing is to use verification activities like inspections and technical reviews earlier in software development cycle, before the testing process. This will decrease the testing effort.

Other activities costs include various costs, like:

- the manpower required to configure and administrate specific components
- the effort required to write the technical reports and product documentation and guides, either in electronic or paper format.

The cost of an e-business application can be computed using the data recorded in table 3 with the formula:

$$CEB = AC + DC + CC + TDC + OAC$$

or

$$CEB = MC + SC + HC + OC$$

If CA_i is considered the cost of activity i , where the activities are analysis, design, coding, testing, debugging and other activities, and h_i is the total number of hours required for the activity i , the total cost of manpower for an e-business application is [9]:

$$MC = \sum_{i=1}^n CA_i \times h_i$$

This decomposition can be used for every cost component.

6. Conclusions

Developing e-business applications usually is an expensive process. Recording the costs on various phases of the development process for different resources for many software projects, could help the managers to estimate and evaluate the costs of future e-business applications.

The development costs can be reduced by using COTS (Components on the Shelf), by reusing software and by using automated tools in different stages of the software development cycle.

The process of identifying errors in a program through an adequate testing leads to an increased reliability for software. Building reliable software is conditioned by making good testing. As software testing process is good, as customer's costs are lower. This happens also to the producer.

Using verification and validation testing methods in all software development phases (analysis, design and implementation) will increase the reliability of software applications. In order to be efficient, the testing process should be followed by good activities of correction and debugging. This will assure that all discovered errors will be corrected and new errors will not be introduced.

References

- [1] R. S. Pressman, *Software Engineering – A Practitioner’s Approach, An European Adaptation, Fifth Edition*, McGraw-Hill, 2000
- [2] J. F. Peters, W. Pedrycz, *Software Engineering – An Engineering Approach*, John Wiley & Sons, Inc, 2000
- [3] P. Pocatilu and C. Ciurea, "Collaborative Systems Testing," *Journal of Applied Quantitative Methods*, Vol. 4, No. 3, 2009, pp. 394-405
- [4] P. Pocatilu, *Costurile testării software*, Editura ASE, 2004.
- [5] A. Samaroo, S. Allot and B. Hambling, *E-effective testing for E-commerce*, 1999.
- [6] I. Ivan and P. Pocatilu, *Object oriented Software Testing*, Infocore Publishing, Bucharest, 1999.
- [7] I. Sommerville, *Software Engineering, 6th Edition*, Addison Wesley, 2001.
- [8] B. Ghilic-Micu and M. Stoica, *eActivitățile în societatea informațională*, Editura Economică, București, 2002.
- [9] P. Pocatilu and C. Toma, "Calitatea aplicațiilor mobile," *Proceedings of International Conference "Rolul științei și învățământului economic în realizarea reformelor economice din Republica Moldova"*, Chișinău, 25-26 September 2003, pp. 474-477

Author



Paul POCATILU graduated the Faculty of Cybernetics, Statistics and Economic Informatics in 1998. He achieved the PhD in Economics in 2003 with thesis on Software Testing Cost Assessment Models. He has published as author and co-author over 45 articles in journals and over 40 articles on national and international conferences. He is author and co-author of 10 books, (Software Testing Costs, and Object Oriented Software Testing are two of them). He is associate professor in the Department of Economic

Informatics of the Academy of Economic Studies, Bucharest. He teaches courses, seminars and laboratories on Mobile Devices Programming, Economic Informatics, Computer Programming and Project Management to graduate and postgraduate students. His current research areas are software testing, software quality, project management, and mobile application development.