BENEFITS AND LIMITS OF QUALITY COST CONCEPT APPLIED TO SOFTWARE INDUSTRY

Cătălin Afrăsinei-Zăvoianu
„Babeş-Bolyai" University, Cluj-Napoca

Abstract: *Quality Cost approach is important to be implemented for each product or project of any software company, wherever it's possible, because it provides additional and more accurate information about costs, costs determined by level of product/project quality.*
In order to minimize the costs of the required quality level in software industry is important to find out a balance between prevention costs and failure costs. But, even if the prevention costs are very high, it doesn't assure the elimination of all quality problems or it finally drives the product/project to an unacceptable price from the consumer point of view.

1. What is Quality Cost of a software application?

Quality costs are those costs directly linked to preventing, finding and correcting defective work that generates errors, defects or gaps. Studies have been revealed that these costs are huge running at 20-40% of sales (Juran and Gryna, 1988). We are not sure these values are applicable in software domain also, but surely they have an important proportion, big enough to catch attention and generating focus on optimizing them. We believe many of these costs could be significantly reduced or even completely avoided.

One of the key tasks of the person responsible for the quality inside an organization (regardless he or she is named „quality engineer" or something else) is to decrease the total cost of the quality associated to a product.

Starting from the „classical" definitions of the quality costs, detailed by categories, we are trying to adapt them to software domain in the following:

- *Prevention costs*: are the costs generated by those activities special designed to prevent poor quality. Examples of "poor quality" could be functional misunderstandings, poor requirements description, design errors (of the application or database), coding errors, mistakes in user guides or source code development without a programming discipline, badly documented and hardly maintained. It's important to remember that these prevention costs couldn't be included in a „budget" of testing team or, in other words, these costs cannot be charged only to testing team. Such costs can be associated to departments or teams having programming, design or even sales tasks.

- *Appraisal/evaluation costs*: are the costs of activities designed to find quality problems, such as code inspections and any type of testing (technical,

functional, endurance tests). Design reviews are part prevention and part appraisal. If the purpose is to find possible design errors, then all the activity is included in appraisal category. If for the same design we are looking for ways of increasing efficiency and reliability, then all we are doing is more likely prevention than appraisal.

- *Failure costs*: are the costs generated as a result of poor quality, such as: costs of fixing bugs or cost of dealing with customer complaints. The failure costs are split in two subcategories:
  - *Internal failure costs*: are the failure costs that appear before the company delivers products to its customers. Beside the costs of finding and fixing application bugs, it might be more other costs of errors coming from groups other than developing department. If an error locks someone else's work from the organization, the costs of wasted time or supplementary time necessary to get back to the normal level of work are internal failure costs.
  - *External failure costs*: are those costs that arise and are discovered after the company supplies the product to the customer. It's about customer service costs, the cost of patching a released product and distributing the patch etc.

As a definition, Total Cost of Quality means the sum of quality costs: Prevention, Appraisal, Internal Failure and External Failure.

As an example, noting or sensing some delays from the schedule regarding the implementation of a project that has a software application delivery as a final step, have negative implications on final costs of a project. If the delivery term cannot be kept, it is possible to determine some penalties that are included in failure costs. If it is a must to respect, with any price, the delivery term, then would be possible some project stages to be skipped or artificially reduced just to be in time. It is possible, for example, to reduce the testing time. The negative effects will be visible later on, after the product is delivered to customers, when they are using it. The impact will be smaller or bigger but surely will negatively affect the company's reputation. Both scenarios generate failure costs that belong to external failure costs subcategory.

It is also possible another scenario in the above example: to require supplementary work or allocation of extra human and material resources in order to "retrieve" the lost time or to eliminate the delay. All these actions determine supplementary costs included in internal costs category and fortunately they are not known by the customers and don't alter the partnership with them.

The external failure costs are considerable. It is cheaper to solve the problems before the product is delivered to the customers. A part of these costs has to be very carefully treated. For example, the costs generated by the efforts to decrease the negative effects of errors discovered by the customer should be a distinct chapter of a budget which we could call it "public relations budget". Such a budget cannot be entirely considered as being designated to cover the quality costs but that amount of

money spent to diminish the negative publicity determined by programming errors take part of the failure costs for sure.

In Table 1 we synthesized some examples of quality costs associated to software products, classified in the four categories mentioned above:

Table 1

Quality Costs in Software Industry

Prevention	Appraisal
• Staff training • Requirements analysis • Clear specifications • Complete and accurate internal documentation • Evaluation of the reliability of development tools (before buying them) or of other potential components of the product • Programmers team well dimensioned from number, structure and professional capacity points of view etc.	• Design review • Code inspection • Testing personnel training • "Beta" version testing • Testing automation • Usability testing etc.
Internal failure	External failure
• Bugs fixing • Testing delays • Writing documentation delays • Product promoting or advertising delays • Direct costs because of delivery delays • Opportunity costs determined by delivery delays	• Costs of technical support • Preparation of documents describing the ways of solving problems • Investigation of customers complaints • Incentives given to compensate problems that have appeared • Recoding and retesting of new versions that solve recorded errors • Delivery of new versions • Costs of maintaining different versions of the same product in the market • Lost sales • Lost customers trust • Discounts to resellers to encourage them to keep selling the product • Warranty costs • Product reliability costs • Paid penalties etc.

There is a special category of quality costs, seldom approached in this area of literature. An example could be the costs generated by treating graphic user interface (GUI) with low priority. This is a mistake because the sales and marketing staff need pictures of the user interface of the product much before the application is released.

GUI errors – those that will be fixed later on – might create difficulties when somebody is trying to capture images necessary to be included in the presentations carried on the front of potential customers. All this generate delays, uncompleted (partial) or unattractive presentations, bad product promoting. All these things determine supplementary costs that belong to hidden quality costs, very difficult to be identified and quantified or if the effect is quantifiable there appear difficulties on identifying the causes that generated them.

Considering costs like those of lost opportunities or delays as being numerical estimators of the total quality costs might be controversial. Campanella doesn't include such costs in a detailed list of examples presented in one of his books (Campanella ed., 1990). Gryna recommends against including these cost types because of difficulties that would appear when trying to account the quality cost (Juran and Gryna, 1988). We consider as being useful to include these costs in total quality costs calculation procedure if these can be correctly quantified.

2. Why is important quality costs approach?

Over the long term, a project (or corporate) cost accounting system that tracks poor quality related costs becomes a powerful management instrument. It is what Juran, Feigenbaum and their adepts have promoted and are promoting, describing in an as eloquent as possible manner such systems that follow the total quality management philosophy.

When we talk about software companies, generally low dimension companies, customer oriented, they don't see Total Quality Management as a priority. Their goal is to provide quality products (software applications) by default. It is simply a critical condition such an application to work. In this industry is not acceptable to have „almost working applications". So the quality exists but the question is how much does it costs right now and how it could be optimized. That's the reason why the approach should be a simpler one, more tactical. However, quality costs analysis deployed at project level or software product are important even if a company is not directly involved in Total Quality Management System or other quality management model.

Here is an example: let's suppose that some of the application functionalities were designed in an annoying manner for the customer. The person who raises this idea will present it to the project manager who is possible to reject it as being subjective. The answer will be: "it is not a bug". What to do in such a situation if you don't want to drop this issue? One approach is to talk to other persons, from higher hierarchical level of the company. But without strong arguments the chances to convince somebody are extremely low.

Instead, we propose another approach. Rather than saying that your opinion is that the customers will be unhappy of some functionalities, collect some data that uphold your opinion:

- Questioning those who create documents: is application hardly understandable and its utilization so difficult then causes supplementary efforts to create user documentation? A simpler design would decrease the time allocated for writing and the number of pages of the user manual.
- Questioning those who make staff training: will be needed extra time for training and supplementary papers because of the difficult design?
- Questioning those from technical support and customer service: will raise the costs associated with these activities? Will be necessary more time to train the personnel to offer any kind of support for the application? Will be more complaints from the customers? Have customers asked for refunds in previous versions of the product because of features designed like this one?
- Check for related problems: Is this design having other effects on the reliability of the program? Has it caused other bugs? Has it made more difficult to change the programming code?
- Questioning the sales staff: If you think that these features are very visible, and visibly wrong, ask whether they will interfere with sales demonstrations.
- Check the magazine reviews: Is this problem very possible to be visible enough to be complained about by reviewers? Collect some articles if they already exist.

All these sources and information collected, containing numbers and facts, generate a data package having much more power to convince and justify the idea. It's interesting to notice the differences in this new posture:

- We are no longer presenting our opinion that some features are a problem. We are presenting information collected from several parts of the company and from external sources that demonstrates that these features are a problem.
- This is the way to demonstrate that modifying the features or functionalities is necessary because it is necessary. No one else in the room can posture and say that you're being "idealistic". Even more, statements like this can be done: "This design is going to cost us this amount of money in failure costs. How much will it cost to fix it?"
- The estimates are based on information coming from different sources interacting with the project. If their points of view are correctly presented, we will get support from them and the opinion is not a personal one anymore.

3. Implementation risks

Gryna and Juran have revealed some problems that have generated failures in „quality costs" approaches (Juran and Gryna, 1988; Juran and Gryna, 1980). We mention two of them.

First, it is no very wise to try to get too much and too fast. For example, it's not a good idea to implement the system of quality costs to every project of the company until the system have been proved itself as being a successful one in at least one project. Also, it is better not to try to measure all costs because probably will be very difficult.

Second, it is better not to insist on costs we could call „controversial". Gryna points out several types of costs that other managers might challenge as not being quality-related (Juran and Gryna, 1988). If these costs are included in totals (such as total cost of quality), some others will believe that you are padding these totals, to achieve a more dramatic effect. Gryna's advice is to not include them. It is usually a wise advice, but it can lead you to underestimate your customer's probable dissatisfaction with your product.

4. „Uncovered" side of quality cost analysis

Essentially, quality costs analysis is no more than a specific activity carried out by a group of individuals, having this responsibility, including the use of particular techniques and data analysis, which finally underlines the ways the quality could be achieved with the cheapest possible costs.

Quality cost analysis looks at the company costs, not the customer's costs. The producer and seller are definitely not the only actors that register quality-related costs. Customers are also affected by poor quality. If a producer sells a low quality product, the customer has to face the supplementary expenses in dealing with that bad product.

The most famous example of the quality cost analysis evaluated only at company level without considering the customers costs from their point of view is that one offered by Fiat Punto. The analysis was made starting from costs associated with fuel tank integrity problem. The calculations made by company are synthesized in Table 2 (Keeton, Owen et al, 1989, p 841; Posner, 1982, p 225).

In other words, it looks cheaper to pay an average of 200000$ for each death as compensation than to pay 11$ per car to prevent fuel tank explosions. Ultimately, the lawsuit losses were much higher.

The above example is not a singular one. Another case taken place between General Motors and Johnston in 1992 (Burroughs Corp. v. Hall Affiliates, Southern Reporter, 1982), the last being the victim. It is about a defect part from fuel injection system of one of the pickup models. The truck stalled in an unhappy moment and Johnston's seven-year old grandchild was killed. The Alabama Supreme Court justified an award of $7.5 million in punitive damages against GM by noting that GM

saved approximately $42,000,000 by not having a recall or otherwise notifying its purchasers of the problem related to the truck.

Table 2

Example of a money evaluation of benefits and costs

Benefits and costs associated with fuel leakage discovered at fuel tank
„Benefits"
Statistics — 180 burn deaths, 180 serious burn injuries, 2100 burned vehicles
Unit costs – 200.000$ per death, 67.000$ per injury, 700$ per vehicle
Total „benefit" — 180 x (200.000$) + 180 x (67.000$) + 2100 x (700$) = 49,5 million $
Costs
Sales — 11 million cars, 1,5 million light trucks
Repairing unit cost – 11$ per car, 11$ per truck
Total cost — 11.000.000 x (11$) + 1.500.000 x (11$) = 137 million $.

Of course most software applications don't lead to bad accidents or deaths. Most of the software projects have as target elements like costs, time, efficiency, reliability. Nevertheless, it is better to take into account the fact that in case of failure, this thing might cost the customer much more than the company. In the above table there are presented several external failure costs that are borne by customers:

Seller: external failure costs (costs that are borne by seller, which sales the product with errors)	Client: failure costs (costs borne by client, which buys the product with errors)
• Technical support • Preparation of documents containing solutions of different problems • Investigation of customers complaints • Refunds and recalls • Recoding, testing and releasing new versions • Deliver and install the updated product • Supplementary costs because of maintaining several versions of the application in the market • Lost sales • Lost customer trust and goodwill • Discounts to resellers to encourage them to keep selling the product • Warranty costs • Liability costs	• Wasted time • Lost data • Lost business • Some tensions between client and product seller • Resignation of some employees frustrated by the application • Demos and presentations to potential clients fail because of the software • Costs of replacing the product • Costs of reconfiguring the system • Costs of extra hardware acquisition required just to operate the software at an acceptable level • Supplementary hardware costs • Costs of recovering lost data

The point is to transfer some of the costs borne by a cheated or injured customer back to the maker or seller of the defective product. More and more, there are plenty of cases against computer companies and software companies won by unsatisfied customers in situations like that described above.

Another important aspect, generally neglected (in Romanian software organizations, at least) by those directly involved in implementing some quality principles or even some certified quality systems, is the role of organizational culture and its influence on projecting, planning, implementing, and continuous working of the quality system. Organizational culture has a crucial role on the way the quality principles are interpreted and implemented. It even influences the approaching style applied when quality costs analysis are taking place.

The overall success of a firm and particularly the success of quality approach (including quality cost analysis) depend on the degree of commitment and involvement of both top management and employees (Drăgulănescu, 2007 apud Păunescu, Purcărea et al, 2008, p. 106). As Necşoiu and Mainea stated, the success of a quality system implemented in an organization is directly conditioned by the nature of the organizational culture, by pre-existent cultural values, the flexibility of the organization (Necşoiu, 2006; Mainea 2006).

Of course, the existing organizational culture could not be a proper one and it should require to be adjusted. The very first stage in this case is to realize the necessity of change and then to have the capability of changing the culture in the right way. This topic is not a subject of the present article, and, therefore, it is not detailed at the moment.

5. Conclusions

In order to have accurate information about costs of a product, it is not enough to know the costs of making it because a very important amount is occupied by costs determined by level of product quality. This is why Quality Cost Analysis is so important and important to be implemented for each product or project of the company, wherever it's possible.

The problem of quality cost analysis is that it drives to an underestimation of effects and risks determined by customers' dissatisfaction. We believe that total quality cost estimation of a project or product has to include also external failure costs borne by customers even this estimation remains just fairly close to accuracy because of the difficulty of collecting necessary data.

Significant decreasing of quality costs is an important goal of every company, including software industry. Completely elimination of quality related costs is a great challenge indeed but our opinion is that it is almost impossible to reach this goal, especially in software industry, because the effort to maintain quality at a certain level implies costs of prevention and evaluation at least.

In order to minimize the quality costs in software industry we have to find a balance between prevention costs and failure costs. But, even if the prevention costs are very high this fact does not assure the elimination of all quality problems. That is because of some important features of software application projects:

- High complexity: application coding is even today a creative task, and the quality of the result depends very much on the programmers experience, abilities and talent. In the same time, there are many translations (from functional specification document to analysis model, from analysis model to design model, from design model to source code, from source code to machine code etc) where the human errors can appear and proliferate.
- Intangibility (or invisibility) of the analysis models, source code and documents that stay behind the application interface;
- Difficult to measure and quantify;
- Difficult to test: for big applications it is almost impossible to use formal verification methods to prove the correctness of the code.

In practice, quality control procedures applied to software applications are focused on decreasing defects or errors, much more than guarantee the quality of final product (Norris, Rigby et al, 1993).

References

Barbier, F., Briand, H., Dano, B., Rideau, S. (1998), The executability of object-oriented finite state machines, *Journal of Object-Oriented Programming, SIGS Publications,* 4(11), pp. 16-24, July/August

Burroughs Corp. v. Hall Affiliates, Inc. (1982), *Southern Reporter*, Second Series, vol. 592, pp. 1054-1061

Campanella, J. (ed.) (1990), *Principles of quality costs*, ASQC Quality Press, Appendix B, „Detailed Description of Quality Cost Elements"

Drăgulănescu, N. (2007), *Competitiveness Through Quality – A New Challenge for Romania,*. Proceedings of the seventh international conference of the Central and Eastern European countries, Iaşi, 5th-6th of December 2007

Juran, J. M., Gryna, F. M. (1980), *Quality planning and analysis*, Second Edition, McGraw-Hill, pp. 30-32

Juran J., Gryna, F.M. (1988), *Juran's Quality Control Handbook*, McGraw-Hill, Fourth Edition

Keeton, W.P., Owen, D.G., Montgomery, J.E., & Green, M.D., (1989), *Products liability and safety, Cases and materials*, Foundation Press, Second Edition

Mainea, M. (2006), Cultura calității. Impactul orientărilor culturale asupra abordării calității în firmele româneşti (I), Crearea şi consolidarea culturii calității, *Calitatea – Access la succes*; nnul 7, nr. 11, pp. 15-18

Necşoiu, R. (2006), Maturitatea SMC şi intensitatea orientării către calitate a culturii organizaţionale, *Calitatea – Access la success,* anul 7, nr. 6, pp. 11-14

Norris, M., Rigby, P. and Payne, M. (1993), *The healthy software project: A guide to successful development and management*, John Wiley

Păunescu, C., Purcărea I., Pantea C., Managementul calităţii în organizaţii prin managementul performanţei, *Management & Marketing,* 2008 (1), pp. 105-116

Posner, R.A., (1982), *Tort law: Cases and economic analysis*, Little Brown & Co