Project Management and Quality Assurance in a Distant Software Development Project

Avram Eskenazi(1), Niv Ahituv(2), Nelly Maneva(1), Rumen Radev(1)

(1) Institute of Mathematics and Informatics, Bulgarian Academy of Science, Sofia, Bulgaria **(eskenazi@math.bas.bg)**

(2)The Marko and Lucie Chaoul Chair for Research in Information Evaluation, Faculty of Management, Tel Aviv University, Israel **(ahituv@post.tau.ac.il)**

This work was partly supported by the Joint Research Project "Evaluation of Information Systems and their Supporting Technology" between the Institute of Mathematics and Informatics - Bulgarian Academy of Sciences and the Institute of Business Research, Tel Aviv University.

Abstract: A series of software development projects were initiated in Germany and developed in Bulgaria. The initiator and the developer articulated a method to facilitate project management and quality assurance of distant development. The method covers a number of aspects relating to remote software development: personnel selection, timetable setting, remote and face to face review meetings, communication of change request, message monitoring, and the like. A "locking" mechanism was developed to assure synchronization of change requests and a framework to monitor and control e-mail correspondence was established and maintained. Some conclusions relating to working procedures and cost saving are presented.

I. Introduction

The rapid advancement of telecommunications has significantly facilitated the geographical distribution of software development projects. Today, the so called software development life cycle (SDLC)[1] can be carried out in various locations, that is, the customer initiating the project is located in one place, the system design team in another, the programming is

performed in a country where labor cost is relatively low, while the implementation is undertaken, usually but not necessarily, in proximity to the initiator.

Although all the above is technologically feasible, still remain the questions of project management and quality assurance. How the initiator of a software development project can monitor and assure the quality of the software products from a distant location?

The purpose of this paper is to outline some models and guidelines helping to confront this challenge. The paper presents an approach to Software Quality Assurance (SQA) for a remote development. It is based on normative approach as well as on practical experience of the authors.

In the next section, the background and a brief description of the project and its constraints are presented.

The management of human resources, the SDLC and the solution for the communication problems are discussed in Section 3.

Section 4 summarizes the practical experience gained in a series of projects.

The last section provides some needs for future studies.

II. The Project

A series of consecutive software development projects was proposed to be carried out in Bulgaria by the software development department of a big German corporation. All projects were in the field of the development of courseware and other types of dedicated software. One of the major projects consisted of developing a software tool for the analysis, planning and design of Computer Based Training programs. The following constraints were imposed on the developers in Bulgaria:

- the development cost and the project deadline had to be determined in advance and could not be reconsidered;
- the general characteristics of the hardware platform and the software tools for the project had to be determined by the customer; the definitive selection and purchase had to be made by the developer;
- the developer had to articulate and submit for approval the project management scheme and the quality assurance plan;
- during of the development of the project the customer had to be responsible for the:
  - = requirements definition
  - = external (general) specifications
  - = independent testing
- the developer had to be responsible for the:

  - = detailed specifications
  - = programming (coding)
  - = internal testing
  - = maintenance documentation.

The primary challenge in such projects is to assure quality while reducing the cost of software development. Several general models for quality assurance are known, but it seems that none of them is easily applicable to remote software project. So we try to establish our own quality system and apply it to the specific type of software projects and the specific characteristics of the environment. The main step at the beginning of each project is to develop the quality plan. It comprises the set of procedures determining the sequence in which the basic principles of the quality assurance methodology will be applied, the deliverables that are required and their quality characteristics, the controls that help ensure quality and coordinate changes and the milestones that enable managers to assess progress.

Right at the beginning of the engagement, the development team identified and analyzed the following risk factors:

- the requirements definition was incomplete and fuzzy thus no precise estimate of the size and complexity of the desired software system could be made;
- the distant (remote) development was to some extent new for both sides, hence the lack of experience required finding out appropriate working procedures by examining several possible approaches;
- daily communications had to be established by using e-mail with unpredictable level of reliability and in a language (English) foreign for both sides.

Taking into account those factors the development team elaborated and enhanced the normal approach to SQA, which in practice had to encompass the whole new type of the development process.

III. The approach to the SQA

We considered a set of assumptions and built the whole SQA policy around them.

Assumption 1. The quality of the software product depends on the quality of the people involved in the project and their proper management.

Consequently, the plan for personnel management specified the structure of the teams, the requirements for education, qualification and

experience of each team member and their responsibilities.

Two teams were formed.

The customer team consisted of:

- a project manager, who was responsible for the overall management of the project, as well as for the requirement definition and the external specifications;
- a few testers responsible for performing the independent testing and writing the external specifications

The developer team consisted of:

- a project manager responsible for the management of the developer staff and its duties;
- a quality assurance manager responsible for all quality matters, as well as for the documentation preparation;
- programmers (a chief programmer and three members) responsible for executing the developer's tasks.

The members of both teams had to have good command of English.

The customer's project manager insisted on participating in person in the evaluation of the chief programmer of the developer, to reassure that the candidate meets the requirement for specific knowledge in the subject area. It was decided that such evaluation could only be done through a face to face interview (joint meeting #1 was used for this purpose, among others). The other programmers consisted of highly motivated and educated professionals. They were chosen through a selection procedure, performed by the project and QA managers and the chief programmer of the developer.

Assumption 2. The quality of the software product depends on the quality of the development process. We proposed the following model for gradual development:

Stage	Activities of Team 1 (Customer)	Activities of Team 2 (Developer)	Duration (weeks)
1.	Joint meeting #1: Concluding the contract. Key personnel approval Timetable approval		0.4
2	Creation of the software requirement definition and external specifications	Hardware and software selection and acquisition	7
3	Joint meeting # 2. Face-to-face participatory design of Prototype 1		1

4	Testing of components of Prototype 1. Initial specification for Prototype 2	Development of Prototype 1	12.4
5	Joint meeting # 3. Acceptance test of Prototype 1. Discussion of the initial specification of Prototype 2.		0.6
6	Finalizing specification of Prototype 2 Testing of components of Prototype 2	Development of Prototype 2 Documenting the system	18
7	Joint meeting # 4. Acceptance test for the final system. Evaluation of the whole project.		0.6
		TOTAL	40

The fact that the two groups were at a distance, required a very careful coordination of the above scheme. Due to the relatively small scale of the project, each of the joint meetings increased substantially the overall cost. Their number - four, according to our view, was the absolute minimum, below which the quality of the final software product would have been impaired. A decrease of one meeting could have been possible in case only one prototype had to be developed. But the customer obligatory needed a limited working version of the product at a very early stage for demonstration purposes. The customer during the negotiations unconditionally set this requirement.

Another restriction, again due to cost consideration, was the number of the other country's participants in the joint meetings. This number was established individually for each meeting, according to its goals, but the optimal number happened to be two each time:

Event	Host	Guest Participants
Joint meeting 1	Customer	Project manager; Chief-programmer (candidate for)
Joint meeting 2	Developer	Project manager; Tester
Joint meeting 3	Customer	Chief-programmer ; Programmer
Joint meeting 4	Customer	Chief-programmer; QA manager

The quality control (reviews: walkthroughs, inspections, technical reviews [4], as well as a final evaluation) was accomplished in three different levels:

- internal evaluation by the programmers and the quality assurance

manager;

- external evaluation by the testers in the customer team;
- final evaluation performed at Joint meeting #4 by representatives of both teams, including the QA manager.

The main problem was to establish which forms of review would be performable by only one of the sides. It was decided that the developer could carry out each type of the reviews by using a staff of the QA manager, the chief programmer, and the respective programmer. The customer employed for this purpose the project manager and one or two of the testers. Only the final evaluation had to be performed by both teams.

We were inclined to apply objective quality measurement procedures - either by using a set of metrics [3], or by evaluating through classification methods [2]. The analysis showed that the resources of the two teams would have been by far not sufficient to do that.

Assumption 3. The quality of the software product in a remote development project depends heavily on the quality of the communications system.

It was decided that the main stream of information was to be channeled through the e-mail and only in exceptional cases telephone communications would be used, exclusively for short and organizational matters. The reason was the cost, and certainly, the content, which in many cases could not be imported by phone.

(During the first project of the series we had to use ordinary or express mail, instead of e-mail, which was not available yet, but our experience was rather disappointing, because of the low speed, high cost and, surprisingly, in some cases - the lack of reliability).

We established a set of rules for communication, defining the frequency, way of processing, and saving of all messages, as well as special protocol for assuring the appropriate synchronization.

We decided that only the respective project manager can receive and send messages. The sender had to assign a consecutive number to each message sent. The receiver was responsible for analyzing each message content, assigning an identification number according to a preset content codification system, and dispatch it to the appropriate member of the staff with a routine request for the type and term of reaction. The receiving project manager was also responsible for the saving of all messages received.

A very serious communication problem arose - synchronization. The simplest case was for instance, when a developer's programmer received a

message, worked on it and sent a reply and before receiving this reply, the sender sent another message updating his or her previous request. From that moment and up serious coordination problems used to arise, taking several days to restore the regular communication procedure between these two persons. Obviously, such a situation is not relevant in a non-distant environment, as the second message would have become available to the receiver in real time.

We tried various methods to resolve such problems all of which were in some sense a trade off between speed and reliability. At the end, we voted for reliability. The mechanism established was similar to the "locking" procedures well known from the theory of the distributed processing. Each end receiver (i.e. the person whom the message had been dispatched to by the project manager), immediately after receiving a message, had to "lock" reception of other messages on the same matter, except for additional and clarifying information. The locking had to be obligatory accompanied by a term. Within this term the receiver had to answer the message received, or, as an exception, to determine a new term. So, it was clear that the locking was established at a "person-task" level with the "no-update" condition. A more involved question was how to proceed when a given task was performed by more than one person. In this case, we assigned responsible for this task the "person" having the obligation to lock and communicate, and defined the other(s) task participant(s) reporting to him.

IV. The lessons learned

Our experience from the series of projects can be briefly formulated as follows.

The main reason to organize such type of remote common project was the dramatic difference in labor cost between the customer and developer countries. The decision was to be made between two main alternatives - remote development or transporting the developer team to the location of the customer for almost the whole period of the development. An estimate showed that the first approach would be more cost advantageous, without loss of quality and efficiency. The results confirmed these expectations, except that the duration of the project had to be prolonged by approximately 25% (such a delay could have also happened in the second case). As far as the cost savings are concerned, a very approximate estimate (as some part of the cost information is only available to the customer) shows that the cost of the remote development is about one third of the cost of the second alternative.

Other points to be stressed on are:

- undoubtedly in the frame of this type of software projects the communications through e-mail has by far advantages over communications by telephone, fax, ordinary or express mail in terms of cost, speed, documentation and even reliability;

- a clear, unambiguous and possibly concise communication protocol has to be established in advance and measures have to be permanently taken by the managers so that this protocol is strictly kept;

- as the joint working/organizational meetings become of crucial importance, and as their cost could substantially affect the overall cost of the project, they have to be carefully planned in advance in terms of time, place, agenda and participants; this of course is to be done in the frame of the entire project schedule.

V. The need for further study

Both sides - customer and developer - were satisfied with the results obtained and agreed to continue their cooperation with further software projects by using the remote development approach.

There is one open problem - the maintenance of the software product developed [5]. Due to various factors, the previous projects needed no or very little maintenance. The one described here seems to have good chances to be widely implemented, hence - will need significant maintenance efforts. While theory and practice give good directions on how to organize and carry out maintenance in a traditional environment, we do not know yet how to proceed in a remote development environment. We have just started to elaborate appropriate methods and procedures.

References:

1. Ahituv, N., Neumann, S. and Riley, N.H., Principles of Information Systems for Management, 4th edition, W.C. Brown, Dubuque, Iowa, 1994.

2. Eskenazi, A. Evaluation of Software Quality by Means of Classification Methods. J. of Systems and Software, vol.10, No 3, 1989, p.213-216.

3. Kan, S.H. Metrics and Models in Software Quality Engineering, Addison-Wesley, 1995.

4. Sanders J., E. Curran. Software Quality. ACM Press, Addison-Wesley, 1994.

5. Sommerville, J. Software Engineering, Addison-Wesley, 1992.