# Quality Assurance in Distributed Software Development Collaboration

by

Saeyoon Kim



B.S. Civil and Environmental Engineering
Brigham Young University, 1999

Submitted to the Department of Civil and Environmental Engineering in partial fulfillment of the requirements for the Degree of

MASTER OF ENGINEERING IN
CIVIL AND ENVIRONMENTAL ENGINEERING

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY



MAY 2000
[June 2000]



Author.....
Department of Civil and Environmental Engineering
May, 2000

Certified by.....
Feniosky Peña-Mora
Associate Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by.....
Daniele Veneziano
Chairman, Departmental Committee in Graduate Studies

# Quality Assurance in Distributed Software Development Collaboration

by

Saeyoon Kim

Submitted to the Department of Civil and Environmental Engineering
On May 5, 2000 in Partial Fulfillment of the
Requirements for the Degree of Master of Engineering in
Civil and Environmental Engineering

## ABSTRACT

As the world moves into the age of globalization and many software development projects become larger in their scales and complexity, it has become unavoidable and sometimes expedient to utilize the human resources that are scattered all over the world. As more projects and more people requires collaboration in a geographically distributed environment, it has become extremely important to check and watch for the level of quality in the process of collaboration. Nowadays, it is not enough to have the technical capability to be able to collaborate in a distributed environment. It is also necessary to be able to achieve and maintain the high satisfying level of quality in each and every aspect of the project and the product.

However, trying to achieve and maintain the high satisfying level of quality in any software development project is a challenging task. There are many methods and approaches that can be used to evaluate the project and its processes to check, achieve, and maintain the highest quality possible. Depending on the type and the characteristics of the project, what kind of quality assurance approach and method will be taken should be decided. There are no one right way to successfully check and maintain the quality. Therefore, it is important to select the right approach for the individual project. Especially in distributed software development collaborations, with very few information or knowledge about managing quality in such an environment and project available, it is extremely challenging for the Quality Assurance Managers to try to uphold quality.

Thesis Supervisor: Feniosky Peña-Mora
Title: Associate Professor

# Acknowledgements

First of all, I would like to thank God for giving me the opportunity to experience this, the strength to endure this, and the hope to go on from this.

To my parents, for giving me, even from afar, their love, support, trust, and guidance. I would like to thank them for being the role models and examples for me to follow in my life.

I would like to thank my brothers, Saeheon and Saejoon, for their love and friendship, and for being the best brothers in the world.

I would like to thank my friends, Hyungseok Yoo and Ju Hyun Kim, for their many wise words and friendships in times of my hardships and trial.

To friends I've gotten to know here at MIT:
I would like to thank all the M.Eng IT group people, Ivan Limansky, Erik Abbott, Teresa Liu, Kaissar Gemayel, Alan Ng, Steven Kyauk, Wassim El-Solh, Kenward Ma, Paul Wong, Hermawan Kamili, and Nhi Tan, for their hard work, support, motivation, and friendship. I would also like to thank Keng Yong Chan, Junko Sagara, Kim Luu, Janice Yang, Heidi Li, and Shelly Bramer, who have helped me survive this experience and also made this experience a lot more fun.

I would also like to thank Moon Seo Park, Seong-Cheol Kang, Joonsang Park, Sung-June Kim, and Yun Sung Kim, for their support, for treating me like a family, and for giving me numerous advice and assistance to make this MIT experience a memorable one.

I would also like to thank Professor Feniosky Peña-Mora, for his guidance in writing this thesis, for his encouragement to constantly improve myself, for his time giving advice concerning my future job, and for providing this opportunity to participate in this challenging and worthwhile project.

Lastly, to my mother, who has been and still is the person influencing and shaping my life. I would like to thank her for constantly watching over me from above and motivating me to become a better person everyday.

# Table of Contents

# List of Figures

# List of Tables

# Chapter One

## Introduction

This thesis examines the role of quality assurance in the development of software in a distributed collaborative team project. The thesis is based on the information and knowledge obtained through the research and experiences of managing the quality assurance team of ieCollab, the Intelligent Electronic Collaboration. ieCollab was a software development collaboration project that took place between diverse group of students from the Massachusetts Institute of Technology (MIT) in Cambridge, Massachusetts, USA, the Centro de Investigación Científica y Estudios Superiores de Ensenada (CICESE) in Ensenada, Mexico, and the Pontificia Universidad Catolica de Chile (PUC) in Santiago, Chile, between September 1999 and April 2000.

This thesis will also focus on the problems faced by the Quality Assurance Team in the development cycle of ieCollab Project. It will also focus on the benefits obtained and lessons learned from working as the Quality Assurance Team. It will be shown that in order to maintain the acceptable quality in each step of the software development cycle, the quality assurance team has to take the initiative in every step and not take a reactive role. This study will also show the factors that must be present in a distributed

collaborative software development project in order for the Quality Assurance Managers to effectively check, achieve, and maintain the quality of the project successfully in every phase.

## 1.1 Motivation

As the world moves into the age of globalization and the scale of many projects increases in its size, it has become unavoidable to involve people who are in remote locations to work in the same project. Many projects including software projects had to overcome working in a distributed collaboration to attain a successful outcome. This means that team members of a project frequently have to work with their peers without ever meeting the other person face to face. The technologies have much evolved to provide us with better media of communication that lessens the limitation of this kind of distributed collaboration through teleconferencing, emails, Internet chatting among other technology. However, even this great development of technologies will not solve every limitation and frustration of collaborating in distributed environment.

Given the fact that there will be limitations and frustrations and the fact that there will be a lot of tasks carried out among people who are from different corners of the world, it is extremely important to check for and maintain the highest quality in the process of each and every development stage of projects. The role of quality assurance managers becomes very important in the distributed collaborative environment. The quality assurance managers have to make sure that the quality of the entire project is not

sacrificed due to the disadvantages and difficulties that the distributed teams bring into the project.

## 1.2 Thesis Organization

This thesis is divided into six chapters. The first chapter, which is the current chapter, is the Introduction Chapter. This chapter explains briefly what this thesis is going to be about. It also talks about the motivation behind writing the thesis.

The second chapter will show the overview of quality assurance in a software development project. It will discuss the need of quality, especially in a distributed project, in order for the project to be truly successful. It will show different phases of software development life cycle and the reason why the quality needs to be present and emphasized at each phase. It will also introduce some of the quality factors and criteria and their definitions to which the quality should be measured against. This chapter then will go on to talk about some of the analytical quality evaluations like peer reviews, walkthroughs, audits, and inspections. It will describe planning and preparation that need to be done in order to conduct each of these evaluations. The following section will discuss how the quality assurance team carries out its duties in a distributed collaboration. It will introduce the Distributed Inspection Process Model and describe in detail how an inspection can be executed in a distributed environment. Lastly, this chapter will show the Capability Maturity Model (CMM) and its five distinct levels. It will show how the role of quality assurance comes into play in each level. It will also show that if a project was one of the distributed nature, then the original CMM, which is designed to meet the needs of certain organizations and has a particular order for improvement actions, may be

inappropriate for a distributed collaborative software project and some modifications are necessary.

The third chapter will introduce the Intelligent Electronic Collaboration (ieCollab) Project, which is a distributed collaborative software development project held from September 1999 to May 2000 between students from three universities that are geographically scattered in the western hemisphere. It will discuss the organization and the team. It will discuss how each and every team was formed with at least one of the team members from a different university in order to promote and encourage working in distributed collaboration. The chapter then will focus on the objective of the project, the process model of the project, the challenges that the team faced, and the benefits that the team got over the course of the project. It will discuss how the objective was not just to obtain an end product but to experience and learn from working in a distributed collaboration, to feel and live the difficulties and challenges of having a team member in a remote location in achieving a common task, and to study and research to suggest and recommend ways to improve the collaboration between the distributed teams in a given project.

The fourth chapter will discuss applying quality assurance to ieCollab Project. It will discuss some of the challenges of applying quality assurance to ieCollab since the project was not a collocated software development project but one that was of a distributed collaboration. It will introduce the Quality Assurance Plan (QAP) that the Quality Assurance Managers (QAM) of ieCollab submitted to the team. The QAP became the source of guidelines for the QAM in their effort to check, achieve, enforce,

and maintain the highest quality possible in ieCollab Project. It will show what the role of the QAM was in every phase of the project and what their relationship to the rest of the team was. It will show what kind of activities that the QAM carried out with the guide of the QAP and the examples of analytical quality evaluations that was conducted for ieCollab Project. It will conclude the chapter by discussing the benefit and the lessons learned from working as the QAM in ieCollab Project.

The fifth chapter will outline the recommendations for applying quality assurance in distributed software development collaboration. The recommendations were acquired from the experience of working as the QAM for ieCollab Project and the research that the QAM did in their operation. One of the key recommendations was to keep reminding the entire project members of the importance of maintaining the quality performance and ultimately make them be quality conscious in any type of situations. Because it doesn't matter how effective or efficient the methods of checking and maintaining the quality is, if the project members personally do not care about the quality in their work, the quality will never be achieved.

# Chapter Two

## Software Development and Quality

There are many software projects that use the same tools, techniques, and methodologies and even similar software attributes. However, even with those similarities, there are numerous cases where one succeeds and another fails. Therefore, there has to be other factors that play in the success or failure of a software project. As defined by Evans (1986), the successful development of a software system requires the presence of five factors:

1. A preplanned project environment tailored to the needs, characteristics, and development requirements of the application.

2. A smooth blending of technology, project discipline, and development control based on firm requirements and an integrated set of project plans tailored to the characteristics of the project.

3. Application of tools, techniques, and project methodologies covering each phase of development linked together through project controls and data products. They are tailored to technical characteristics, personnel experience, and technical, administrative, and management constraints of the project.

4. A smooth, controlled transition of data consistent with technical and development standards of the project; a predefined, effective flow of responsibility as the implementation proceeds from phase to phase and as responsibility shifts from organization to organization; and a set of quality gates which monitor and evaluate the quality of the data before the

impacts associated with poor quality affect the quality of the end products of the development.

5. A staff that is technically competent to produce the software, motivated to meet schedule and cost commitments, coordinated to ensure that the activities of one organizational element mesh properly with other segments of the project and that the activities of all segments of the project are focused and directed towards a common set of goals and objectives, and that the staff is committed to produce a quality product consistent with project standards, operational and performance requirements, and the developments of the project.

These factors from Evans (1986) that are listed above do not talk about what kind of tools or techniques are being used but they, if read carefully, all deal with the process of a project. The above factors show how important the process of a project is. Whether a project utilizes high-tech million-dollar tools or highly competent, skilled staffs does not decide the fate of a software project. Rather, the success of a software project is closely related to how the entire project will be processed and how each phase of the project will be carried out. Then how can one make sure that each phase or step of a project is executed in such a manner that it will lead to the success of the project? There have always been problems in three main areas of the development that have become crucial to the success of a project. The three areas as defined by Wallmuler (1994) are as follows:

***Time factors***

- Only 5 percent of all projects are completed on time.
- More than 60 percent of all projects have at least a 20 percent time overrun.
- Many projects are terminated altogether because of delays.

***Cost factors***

- Development cost increases exponentially with the complexity of software.

- The higher degree of integration of modern software systems, complex interfaces between components, and the demand for adequate user friendliness and reliability (particularly in interactive systems) also cause higher development costs.
- In many instances, 60 percent more of the entire software cost of a product is spent on maintenance.
- Delays can reduce market opportunities for a product and render investment unprofitable.

***Product quality factors***

- Errors are often found too late, frequently not until the customer tries to put the system into operation.
- The software product documentation is missing, incomplete or not up to date.
- Because of product faults more than 50 percent of development time and effort is spent on error detection and correction.
- Quality as a development aim can often not be proved because of lack of quality planning.

In every software development project, project managers try their best to avoid these problems or at least try to diminish their impact as much as possible. However, most of the times, it is difficult to avoid all three problems. Since in many projects, bigger pressure lies with meeting the time of the product delivery and the proposed budget, usually the product quality is the one that gets sacrificed or overlooked most of the time.

## 2.1 Need for Quality

Time and money surely are very important factors that might decide how the project would be carried out. However, the quality of the product is an important factor just as the other two. During the development of a project, people might tend to consume themselves heavily on meeting the deadlines or working under a specific budget. But at the end, after they met all the deadlines and worked under all the budgets, if their product has a fatal mistake or quality lacking features, this would not be described as a successful

project. Especially in present days, when the types of software that are being developed and produced take up more responsibility and expectations than before, the quality of those products must be uphold to satisfy the demands.

> As computers pervade more areas of our lives and control more of our environment, the potential effects of unacceptable software quality become increasingly disturbing. In many cases our ability to specify requirements; to design, develop, and test systems; and to maintain the software systems once deployed has been outstripped by the complexity and characteristics of the systems themselves. When assessing the effects of this short fall, consider the potential effects of a banking system of questionable accuracy, of an unreliable software system supporting a manned space mission, or a military system which reports incorrect data or data in the wrong format. The need for absolute, predictable software quality becomes evident [Evans, 1986].

In the discussion of quality, one can not just assume that it only has a relationship to the end product that the project creates. The need for focusing on the quality in each stage of the software development arises because it has a correlation to the other factors, the time and the cost. Having errors and faults not only in the product but also in the stages of the project results in the delay of time and the increase in extra cost. There has to be quality checks in every project stages to eliminate these errors and faults. In this way, the time, the cost, and the quality of the project can be positively affected.

## 2.2 Software Development Life Cycle

The software development cycle is composed of several phases, each of which ends with the occurrence of a milestone event, and each of which produces an identifiable

combination of documents or software items and results in a formal review or set of reviews [Wallmuler, 1994]. In different kinds of projects, each phase might be referred to as different names but in its essence, they all have the same general significance. These general phases are project definition, requirement analysis, design, coding, testing and operation and maintenance.

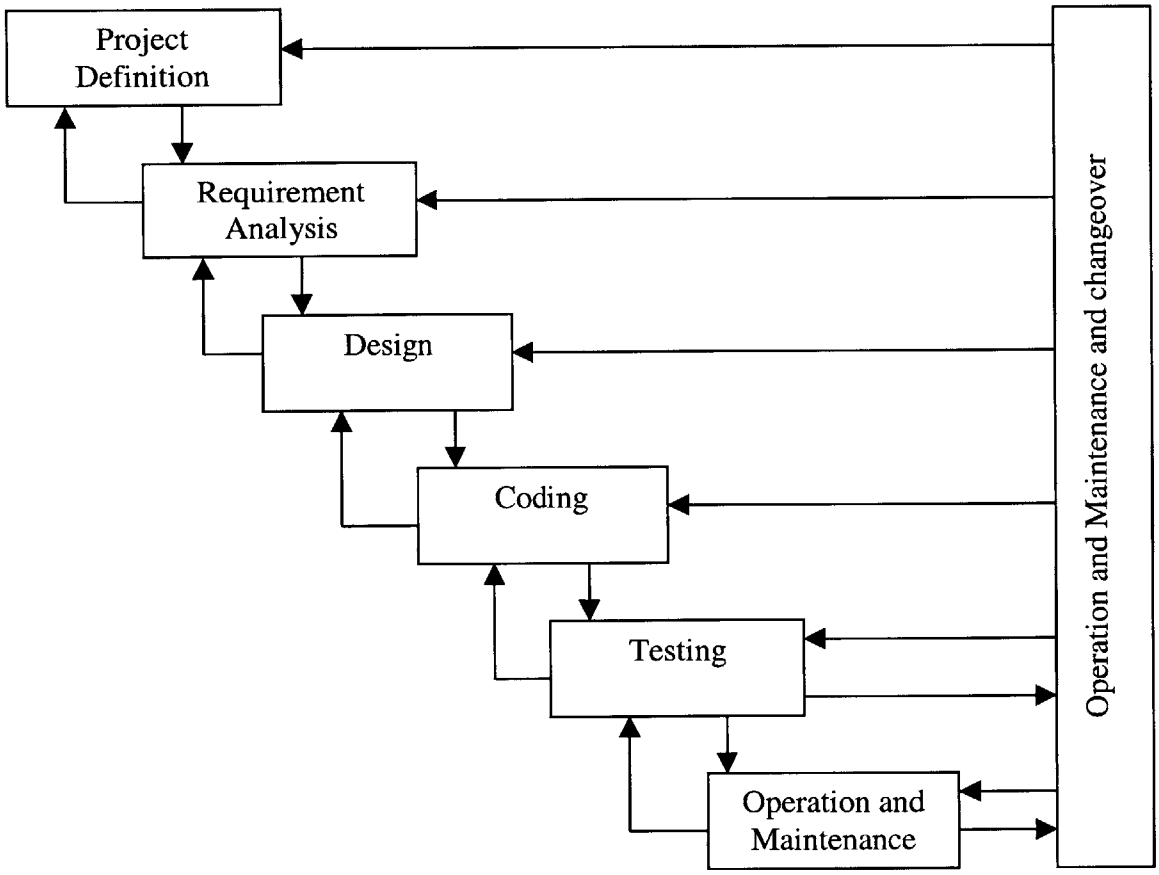


**Figure 2.1** *Software Engineering Development Life Cycle Phases [Wallmuler, 1994]*

***Project Definition***

This phase is the earliest part of the software development cycle in which the basic plan of the development is proposed and set. This phase defines the scope, the purpose, and

the goals of the project: what the project is trying to achieve, what it is going to build, and why. It also outlines the target market, the competitive advantage, and the finance of the project: how and where this product will be utilized, in which area this project has the edge over the competition, and where the project will find funds.

***Requirement Analysis***

The requirement analysis phase is usually the most complex and important phase of software development. It is here that initial requirements are set down defining the system that is to be built. How this phase works out is extremely crucial to the software development. From the project definition declared by the business managers and the marketing managers, requirement analysis team will collect, analyze, and define the requirements of the software system. These requirements will then be reviewed thoroughly to make sure the right system is being proposed and to secure a good-quality product.

***Design***

The design is the bridge between requirements and the programming. From the requirements that are set and defined, the detailed design has to be produced. This design must be complete and accurate according to the proposed requirements in preparation for the implementation. Through creativity and discipline, a good design can be achieved. A good design will make testing of quality and maintenance of software easier.

### *Coding*

Coding phase is when the detailed design that was created from the defined requirements is actually being implemented. People responsible for this phase of the project should be familiar with the project's requirements and the design to be able to write the code that is logical and uniform to the proposed product. Coding can be a very complicated process in which it requires a great amount of collaboration and understanding to achieve the final desired product.

### *Testing*

This is the phase in which the code that was written in the previous phase is being tested to see if it complies with the requirements that were defined earlier. This stage should also check whether the product is reliable enough for installation and user operation.

### *Operation and Maintenance*

This is the final stage of the software development life cycle. One of the earliest activities in this phase is the introductory training of the user. The user's manual would be created for this phase. The maintenance part of this phase should run as long as the product is being utilized.

## 2.3 Software Quality

Quality must be built into the software development cycle from the very beginning. People involved with software development project nowadays must realize how closely related the problems of time, cost, and quality are. Quality assurance has to be playing in each and every phase of the project in order to meet the time and the cost

requirements. It is pretty obvious that a delay in the first phase of the project will affect all the other phases and the project will have a good chance of not meeting the deadline. This is the same case with the errors and faults occurring in the beginning stages of the project. Errors and faults have a tendency to be accumulated and this can be shown in Figure 2.2:

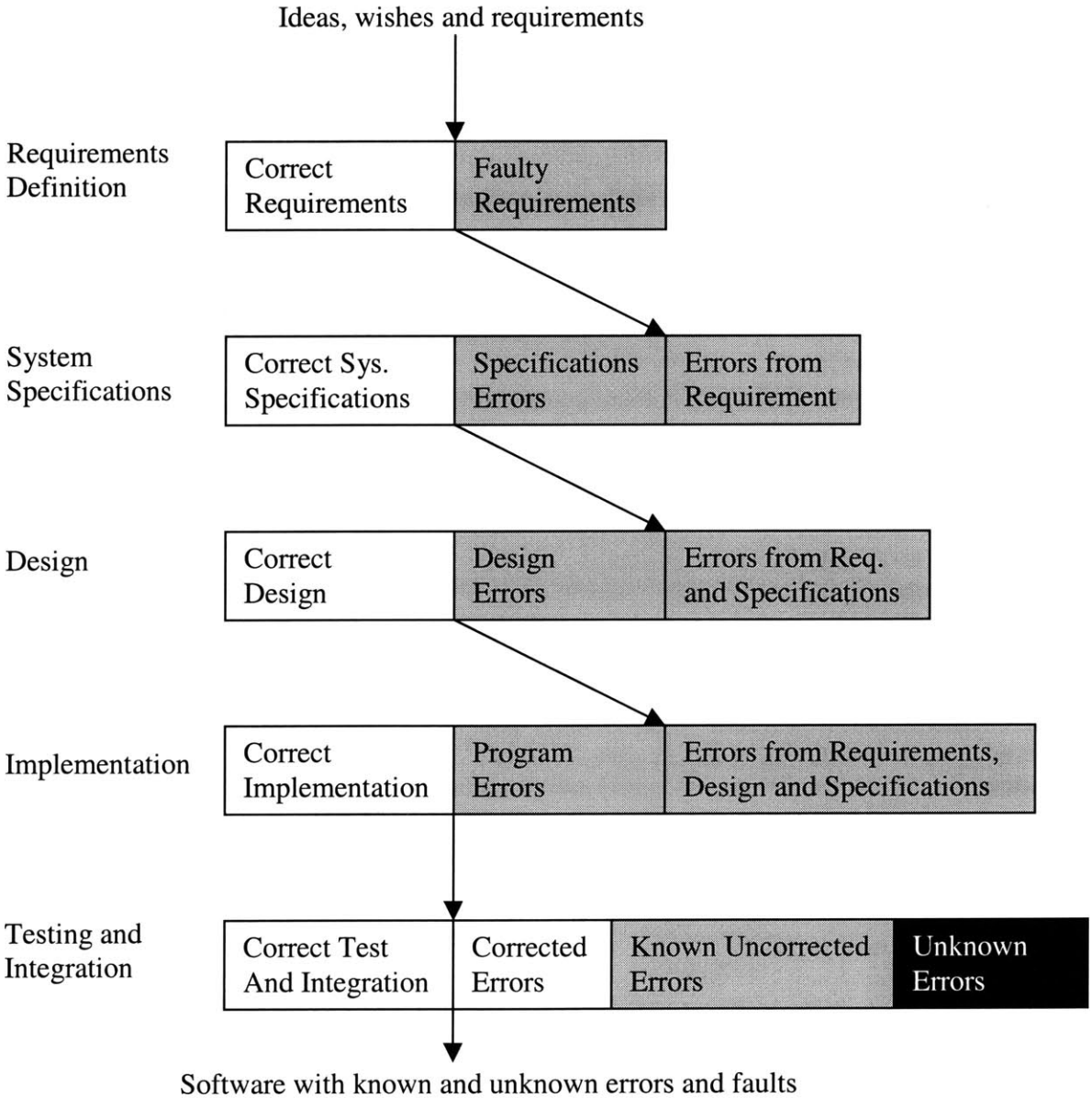


**Figure 2.2** *Accumulative Effect of Errors and Faults [Wallmuler, 1994]*

In Figure 2.2 above, it could be seen that errors that have been generated in the first stage contribute to the next part and so on. The errors that did not get corrected in the first stage results in more errors being created in the latter phases. The error and faults must be found and eliminated in the beginning to reduce the unnecessary amount of time and money wasted in the project. The correction of quality errors and faults can be very expensive, since the cost of correcting the errors increases exponentially with the time that an error remains in the project cycle as shown in Figure 2.3 [Wallmuler, 1994]:

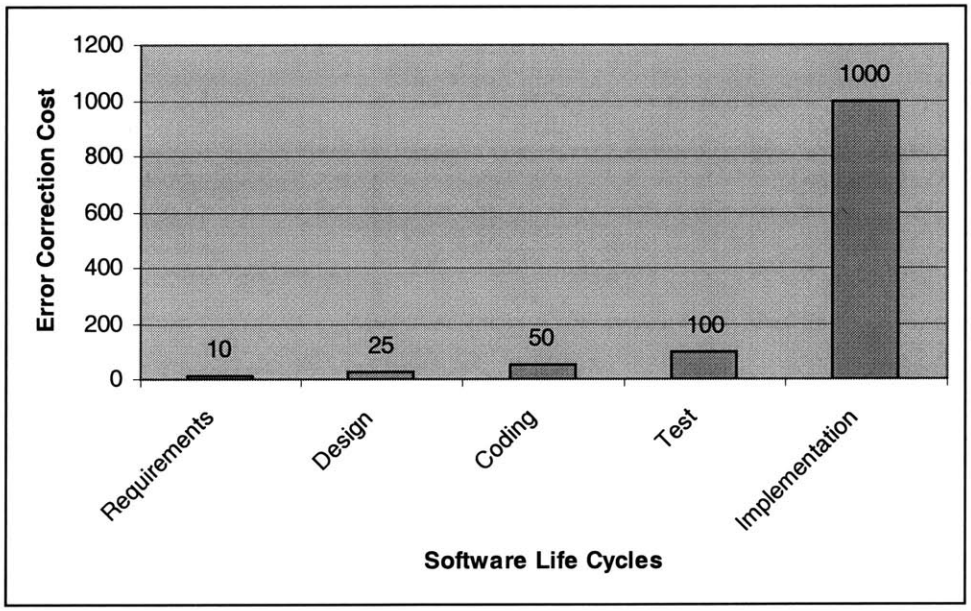


**Figure 2.3** *Error Correction Cost at Different Phases [Cruz, 1999]*

As it can be seen from Figure 2.3, as the development cycle approaches the latter part, the error-correction cost increases tremendously. Errors that are found and corrected in the early stages of the project can save much more money that may unavoidably have been spent later, had those errors been not discovered. It is shown in Figure 2.3 that correcting errors in the later phases can cost 100 times more than at earlier

stages of the development. Finding and correcting the errors in the early stages of the cycle may also save a lot more time for the project. It would be much easier to find and correct errors that occur in a specific stage. Once the error gets passed over to the next phase of the project, it would be difficult and confusing to locate and identify the errors and correct them. This is why it is very important for the quality assurance managers to check and review every phase in the software development life cycle.

## 2.4 Quality Factors and Criteria

The quality of a given project should be measured in various areas and also against various factors to make sure the high standards of quality. Not only the quality during a certain project development phase but also the quality during each and every development phase should be emphasized. Evans (1986) defined in his book that there are three categories relating to specific areas of the software life cycle in which the quality should be concerned:

> ***Performance:*** The requirements that must be stated and agreed to by the user. That stated and agreed requirements will specify how the software will operate.
>
> ***Design:*** The integrity of the design process to implement the software requirements of the user.
>
> ***Adaptation:*** A life cycle concern that affects the user and the maintainer – how easy will the software be to reuse and to evolve to meet new requirements.

Each of the three categories has concerns of its own and quality factors relating to it. The performance category is concerned with such areas as how well the software product functions and how easy it is to use. The design category deals with topics like how easy it is to verify the performance of the software and how easy it is to repair. The adaptation category answers questions like how easy it is to make changes to the software and how easy it is to interface with another application. The following table depicts these concerns and the commensurate quality factors.

**Table 2.1** *Software Quality Factors [Evans, 1986]*

| Acquisition Concern | User Concern | Quality Factor |
|---|---|---|
| Performance<br>How well does it function? | How well does it utilize a resource? | Efficiency |
| | How secure is it? | Integrity |
| | What confidence can be placed in what it does? | Reliability |
| | How easy is it to use? | Usability |
| Design<br>How valid is the design? | How well does it conform to the requirements? | Correctness |
| | How easy is it to repair? | Maintainability |
| | How easy is it to verify its performance? | Verifiability |
| Adaptation<br>How adaptable is it? | How easy is it to expand or upgrade its capability or performance? | Expandability |
| | How easy is it to change? | Flexibility |
| | How easy is it to interface with another system? | Interoperability |
| | How easy is it to transport? | Portability |
| | How easy is it to convert for use in another application? | Reusability |

From Table 2.1, it could be seen that the performance category deals with such quality factors as efficiency, integrity, reliability, and usability which all deal with the

question how well it functions. The design category contains correctness, maintainability, and verifiability, which are about how valid the design is. The adaptation category checks for expandability, flexibility, interoperability, portability, and reusability. Table 2.2 provides brief definitions for these factors. The factors and their definitions will be discussed in extensive detail in Section 2.4.1.

**Table 2.2** *Lifecycle Quality Measures [Evans, 1986]*

| Project/ Program | Quality Factor | Definition |
|---|---|---|
| Performance | Efficiency | Relative extent to which a resource is utilized (i.e., storage space, processing time, communication time). |
| | Integrity | Extent to which the software will perform without failures due to unauthorized access to the code or data within a specified time period. |
| | Reliability | Extent to which the software will perform without any failures within a specified time period. |
| | Usability | Relative effort for training or software operation (e.g., familiarization, input preparation, execution, output interpretation). |
| Design | Correctness | Extent to which the software conforms to its specifications and standards |
| | Maintainability | Ease of effort for locating and fixing a software failure within a specified time period. |
| | Verifiability | Relative effort to verify the specified software operation and performance. |
| Adaptation | Expandability | Relative effort to increase the software capability or performance by enhancing current functions or by adding new functions or data. |
| | Flexibility | Ease of effort for changing the software missions, functions, or data to satisfy other requirements. |
| | Interoperability | Relative effort to couple the software of one system to the software of another system. |
| | Portability | Relative effort to transport the software for use in another environment (hardware configuration and/or software system environment). |
| | Reusability | Relative effort to convert a software component for use in another application |

It should be noted that these factors should be used as a management tool to improve the software development process, specify the quality attributes of a software system by providing an example for specifying reliability, and determine what information managers and software engineers need in order to help them do their jobs better and how to get that information [Kelly, 1993]. It should also be emphasized that there is no universal set of factors that are applicable to all projects and all project environments. Every project must experiment to see what works for them and their environment.

The quality factors and criteria system that is presented above was put together by Evans (1986). The quality assurance team of a given project should know that there are other systems that have been proposed out there. One example of the other systems is the one proposed by Buckley and Poston that consists of twenty criteria. The criteria are economy, integrity, documentation, understandability, flexibility, interoperability, modularity, correctness, reliability, modifiability, validity, generality, testability, reusability, resilience, usability, clarity, maintainability, portability, and efficiency [Vincent, 1988]. However, as it was mentioned above and also stated by Vincent (1988), it is not important which system of quality factors and criteria that the project decides to use. The important thing here is to "ensure that the system is discussed, understood, agreed to, and documented" among every team and people involved in the project, so that they take this system of quality factors and criteria seriously as they use them. It will be discussed in the later chapter (Chapter Four) but it should be noted that ieCollab Project used yet another system of quality factors and criteria from the ones that are mentioned above.

### 2.4.1 Factor Definitions

One of the major obstacles that the quality assurance managers face as they go about making the list of quality factors is not knowing what each quality factor means. Some of the factors seem to be meaning the same thing yet they are listed as different factors. Table 2.2 discussed briefly what the definitions of each factor are. This section will discuss in detail how each quality factor is measured and what they deal with as they are defined by Evans (1986), Schulmeyer (1999) and Sanders (1994).

***Efficiency:*** Efficiency is made up of attributes that have impact on the relationship between the level of performance of the software and the amount of resources used under specific conditions [Sanders, 1994]. The attributes are time behavior and resource behavior. Time behavior measures the response and processing times and on throughput rates in performing its function. Resource behavior measures the amount of resources used and the duration of such use in performing its function.

***Integrity:*** Integrity deals with the security and the auditability of the product. Security is an operationally oriented attribute and auditability is an attribute that helps determine security [Evans, 1986]. It assesses the instrumentation of the product answering question such as how well the software instruments itself, how well execution errors are recognized, and how well special conditions are identified.

***Reliability:*** Reliability is a set of criteria that affect the capability of software to maintain its level of performance under certain conditions for a certain period of time [Sanders,

1994]. There are three methods of checking software reliability. First is on the basis of its failure history. Second is its behavior for a random sample of points taken from its input domain. Third is the number of seeded and actual faults detected by the test team. Seeded faults are those faults that are intentionally put into the program at the beginning of the debugging phase without informing the testing team about it [Schulmeyer, 1999].

***Usability:*** Usability is a set of characteristics that bear on the effort needed for use and on the individual assessment of such use [Sanders, 1994]. It checks for the user-friendliness of the system. It also test to see if the system is extremely difficult to use or requires an excessive amount of continuing education for the skill level of the expected users. It measures understandability, learnability, and Operability. Understandability checks for the user's effort in recognizing the logical concept that is being used in the system and its applicability. Learnability measures the user's effort in learning the system's application. Operability measures the degree of difficulty of operation and operation control.

***Maintainability:*** Maintainability is a set of characteristics that bear on the effort needed to make specified modifications [Sanders, 1994]. A sound development process, resulting in a quality product, is the best insurance for maintainable software. This measures such attributes as analyzability, changeability, stability, and testability. Analyzability is the difficulty associated with diagnosing deficiencies or failures, or identification of parts in need of modification. Changeability measures the degree of need for modification, fault removal, and environmental change. Stability measures the risk of unexpected effect of

modifications and testability measures the degree of difficulty in validating the modified software.

***Verifiability:*** Verifiability deals with the capability to verify that the software design and implementation is in accordance with program specifications [Evans, 1986]. Verifiability is somewhat analogous to the testability of the code. The key criteria are simplicity, modularity, test, document accessibility, and self-descriptiveness. Metrics that are applicable are design structure, complexity, coding simplicity, test checklists, quantity and effectiveness of comments, and descriptiveness of implementation programming language.

***Expandability:*** Expandability deals with the amount of effort involved in increasing the capability of the software [Evans, 1986]. It could take the form of increased or new performance requirements, or enhanced operation to achieve ease of use or efficiency. The attributes of significance are extensibility, generality, modularity, self-descriptiveness, and simplicity.

***Flexibility:*** Flexibility is extremely important to the life of the software product. It measures attributes like modularity, generality, self-descriptiveness, and simplicity. Applicable metrics are interfaces, generality of structure and data, extensibility, quantity and effectiveness of comments, and description of implementation programming language [Evans, 1986].

***Functionality:*** Functionality is a set of attributes that influence the existence of a set of functions and their specified properties [Sanders, 1994]. The applicable attributes include suitability, correctness, and interoperability. Suitability evaluates the presence and appropriateness of a set of functions for specified tasks. Correctness evaluates the provision of right or agreed results or effects. This might include the needed degree of precision of calculated values. Interoperability is the attribute that bears on its ability to interact with specified systems. It is used instead if compatibility in order to avoid ambiguity with replaceability.

***Portability:*** Portability is a set of characteristics that have effect on the ability of software to be transferred from one environment to another [Sanders, 1994]. It touches upon such characteristics as adaptability, installability, conformance, and replaceability. Adaptability bears on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose. Installability measures the degree of difficulty in installing the software in a specified environment. Conformance is the measure of adherence of the software to standards or conventions relating to portability. Replaceability bears on the opportunity and effort of using the software in place of specified other software in the environment of that software.

***Reusability:*** Reusability is similar to portability. The distinction is that portability measures the ability to transfer modules of code from one machine to another while reusability measures the ability to move modules or code to other applications within the

same operating environment [Evans, 1986]. It is, therefore, a lesser requirement than portability. The attributes are application independence, independence, document accessibility, generality, modularity, self-descriptiveness, simplicity, and system clarity. Some of the key metrics are modular interface complexity, quantity and effectiveness of implementation programming language, software system independence, and machine independence.

### 2.4.2 Factor Tradeoffs

After an appropriate set of quality factors are selected and produced for the project, the interrelationships among the selected factors must be considered. It is usually extremely difficult to have high degree of all factors present in the project. If it is unavoidable to have a high degree of some factors, then it might be inevitable to have a low degree of some other factors. Table 2.3 (a) and (b) will show the typical factor tradeoffs. It maybe used as a guide to determine the relationship between the quality factors.

Even if there are tradeoffs, and some factors and criteria might be expected to score low evaluation, the project reviews and audits must still be conducted with all the factors and criteria being considered. However, it should be mentioned in the evaluations and reviews that which quality factors and criteria are in tradeoffs and this will help the users to see which factors and criteria was sacrificed to enhance others in the tradeoffs [Vincent, 1988].

**Table 2.3(a)** *Typical Factor Tradeoffs [Schmulmeyer, 1999]*

| | |
|---|---|
| Integrity vs. Efficiency | The additional code and processing required to control the access of the software or data usually lengthen run time and require additional storage. |
| Usability vs. Efficiency | The additional code and processing required to ease an operator's tasks or provide more usable output usually lengthen run time and require additional storage. |
| Maintainability vs. Efficiency | Optimized code, incorporating intricate coding techniques and direct code, always provides problems to the maintainer. Using modularity, instrumentation, and well-commented high-level code to increase the maintainability of a system usually increases the overhead, resulting in less efficient operation. |
| Testability vs. Efficiency | The above discussion applies to testing. |
| Portability vs. Efficiency | The use of direct code or optimized system software or utilities decreased the portability of the system. |
| Flexibility vs. Efficiency | The generality required for a flexible system increases overhead and decreases the efficiency of the system. |
| Reusability vs. Efficiency | The above discussion applies to reusability. |
| Interoperability vs. Efficiency | Again, the added overhead for conversation from standard data representation and the use of interface routines decreases the operating efficiency of the system. |
| Flexibility vs. Integrity | Flexibility requires very general and flexible data structures. This increases the data security problem. |
| Reusability vs. Integrity | As in the above discussion, the generality required by reusable software provides severe protection problems. |

**Table 2.3(b)** *Typical Factor Tradeoffs [Schmulmeyer, 1999]*

| | |
|---|---|
| Interoperability vs. Integrity | Coupled systems allow for more avenues of access and for different users who can access the system. The potential for accidental access of sensitive data increases as the opportunities for deliberate access increase. Often, coupled systems share data or software, which further compounds the security problems. |
| Reusability vs. Reliability | The generality required by reusable software makes providing error tolerance and accuracy for all cases more difficult. |

## 2.5 Analytical Quality Evaluation

Quality evaluations of a given project can be accomplished through various types of technical reviews. These evaluations can lead to the improvement of the project development process and the product. As stated by Wallmuller (1994), there are several reasons for such evaluations:

- The buyer or contractor wants to know whether a software product meets his or her requirements.
- The project leader is interested in the quality of intermediate and end deliverables of a phase.
- The developer wants to know whether his or her work has reached a high level of quality.

Reviews are also conducted to assess project status, determine software quality, and decide on the acceptability of the product. It is important to use this ongoing and pervasive process to support the quality management program. It is the responsibility of the program and development organizations to provide the necessary management mechanisms to process, and in the final analysis, provide an accurate overall assessment

of the product [Evans, 1986]. In this section, four major technical reviews of analytical quality evaluation will be discussed. They are peer reviews, walkthroughs, audits, and inspections. In addition to these four major technical reviews, verification and validation will be discussed as well.

### 2.5.1 Peer Reviews

A peer review is an informal review that is a structured presentation of technical and process data. It is an evaluation of elements that make up software development to find conflicting areas and recommend improvements [Sanders, 1994]. The review process starts with a definition of the specific goals of the review. The review chairperson and the leader of the group for whom the review is to be conducted decide on the agenda and what they are looking to accomplish from the meeting.

Some of the goals of peer reviews should include a statement of review requirements and needs, a detail plan of how the meeting will be conducted, and an agenda of the meeting. The goals are set to provide some expectations of the meeting, so the people, who will participate, know what they need to do to prepare for the meeting and be ready to discuss the necessary topics. After the goals of the peer review are set, the planning must take place. During the planning, the numbers and types of people who will attend the meeting should be specified and notified. The technical information that is to be presented and the criteria that will be used to evaluate content of the review should be determined. Peer reviews should be conducted following exactly the agenda and the schedule that was planned beforehand. It should also follow the technical requirements that they agreed upon in the planning stage. During the review, the

participants should know what their responsibilities are and how to apply the scoring system to the evaluation. After the review, both the reviewer and reviewee should sum up the data that they obtained from the meeting and write a report. This report should be a resolution of differences that may have been brought up in the meeting and also the list of the action items from the meeting. In case there are issues that are not resolvable then the issue should be included in the report as well [Evans, 1986].

### 2.5.2 Walkthroughs

As outlined by Evans (1986), a walkthrough is an informal review whose primary function is to assess the progress. It is conducted at various points within the development by project proposal. The reviewer in a walkthrough interacts with the review audience rather than conducting a one sided evaluation of a single data item. The objective of walkthroughs is to evaluate application requirements, developing design, code, and levels two to five test data products. Specific walkthrough goals are determined before conducting the walkthrough. In preparation of a walkthrough, the emphasis is put on reviewing the process, and providing constructive criticism. Secondary objectives may include the identification of style errors, improvement in the quality of the material, and the transfer of ideas and understanding between team members. All data products reviewed and approved at walkthroughs are put together by the appropriate development group, entered into the program support library, and released for use in the next phase of development. Some of the planning for walkthroughs are [Evans, 1986]:

1. A walkthrough should be held for any software development material used by more than one organization. The degree of formality is a function of the approval level of data.

2. Walkthroughs shall be scheduled well in advance. The reviewee shall send out a notice, with the review materials attached, to each reviewer.

3. The reviewee should check with all attendees before setting the time and date of the walkthrough to affix a mutually convenient time.

4. The following people shall attend a walkthrough: the reviewee, the appropriate design leader, a peer of the reviewee, and an independent reviewer from outside the development organization (someone from program test, quality assurance, technical support, etc.).

5. Supervisors or managers are invited to walkthroughs when their particular skills or knowledge are required. A supervisor may also attend a walkthrough if interest is expressed, but the supervisor does not attend in a supervisory role, rather as one member of a team reviewing another member's work.

After the planning is done and the date is set, the walkthrough announcement should be distributed to every team involved with the project. Walkthrough should have a moderator to conduct the meeting, a recorder to take notes of the walkthrough, the reviewee, and the representatives from each team to make comments. From the walkthrough, action items that require correction, clarification, or further work will be established. The reviewee will be required to respond to all the action items and make changes after the walkthrough.

During the actual walkthrough, the moderator should be responsible mostly for creating an agenda for the meeting, assigning who will make comments from each team, and monitoring a relevant and useful discussion among other things. The recorder should be taking minutes of the meeting and the action items and issues that come up. The reviewee should present to the entire team the document that is the subject of the

walkthrough. After the presentation is over, the pre-assigned representatives from each team should make comments and raise relevant issues under the direction of the moderator. At the end of the walkthrough, the action items should be handed over to the reviewee to make changes and corrections. If the document is approved at the end of the walkthrough, it will be frozen. The Quality Assurance Managers should follow up with the reviewee to check if the necessary changes and corrections are made and to decide with the Project Managers if another walkthrough should be necessary.

### 2.5.3 Audits

As stated by Summers (1987), "Auditing is a very powerful quality assurance tool, which can be employed at any time during the project." An audit is a formal review that acts as checkpoints, which are in-process evaluations of developing data or process oriented activities. Audits differ from walkthroughs in that they are not participatory in nature. They are more of one-sided evaluations. Throughout the development period regular audits of the development process, management and control procedures being applied to the project, project performance and productivity, and technical compliance of the system to standards should be conducted [Evans, 1986].

In order for audits to be effective, they must follow some guidelines. First of all, the specific goal of the audit must be decided. These goals will differ according to the phase of the project, the experience of the team members, the purpose of the audit, and the use of the audit results. The set goals should be able to define areas like the specific audit assignment, the desired end products, the reason for the audit, and the audit audience.

After the goals have been set, the scope of the audit should be set next. In setting the audit scope, the goals of the audit as well as the schedule and the experience of the audit team members should be considered. Related audit reports or previous working papers should be obtained for references. The scope of audits should be wide as possible. It should not be limited to just being a test of software development activities. The scope should be wide enough so that depending on the outcome of the audit, some events like the rescheduling of tasks or even the removal of code inspections can be decided in order to make sure that the project is on the right track and the productivity is satisfactory [Summers, 1987].

From the goals and the scopes set, the audit team members should develop an audit plan highlighting the audit goals and objectives, the way it will be conducted, the major areas of evaluation, and the expected outputs. When this is all done, then the audit can be initiated. During the initiation, the audit team must prepare a survey guide that will serve as a road map for the auditors to follow [Evans, 1986]. This road map should be based on the preliminary data that was collected. In the actual audit, the audit team must be prepared to carefully look at and evaluate at the current progress of the project with the proposed plan of the project. They must ensure the products are consistent with the project standard and the predefined quality attributes. After an audit is finished, the audit team must analyze the results that they obtained from the audit and report the clearly documented results to the project management.

### 2.5.4 Inspections

The inspections are a formal review that usually take data products or development process characteristics and evaluate them against a predefined set of criteria for compliance. They are much narrower in focus and much more rigorously structured. As stated by Jones (1991), inspections are very different from peer reviews and walkthroughs primarily in its preparation, the roles assigned, and its follow-up. In order to be called an inspection, it must have the following characteristics [Jones, 1991]:

1. The participant must receive training before participating in their first inspection.

2. There needs to be enough preparation time before the actual inspection takes place.

3. There requires attendance at least of:
   - A moderator (to keep the discussion within bounds)
   - A reader (to paraphrase the work being inspected)
   - A recorder (to keep records of all defects)
   - An inspector (to identify any problems)
   - An author (whose work is being inspected)

   However, people can take dual roles if the project is really small.

4. The inspection will follow general standards in terms of timing, recording defects, and polling participants.

5. There should be follow-up meeting to check for actual repairs of the identified defects.

The major purpose of the inspection is to find problems that may cause troubles later and eliminate them in the early stage of the project. Like the audit, the first thing that needs to be done in inspection is the setting of goals. The goals need to define area like the project needs that should be addressed by the inspection. After the goal has been set, the specific standards for the product or the development procedures that are to be

evaluated must be defined. It is recommended that these standards be a relative weight of acceptability than just pass/fail. After these phases are finished, an inspector using the standards defined should conduct the inspection and give evaluation.

Within the inspection meeting, the moderator is the one who is conducting the meeting from the beginning to the end. At the start of the inspection, the moderator should introduce all the participants, "briefly describing their roles, and restating the purpose of the inspection and product." [NASA, 1993] Then, the moderator should introduce the reader who will be presenting the document relating to the product that is the subject of the inspection.

The reader is the one who "provides a logical reading and interpretation of the product." [NASA, 1993] In order to be objective, the reader should be someone who is not deeply related to the product that is being inspected. The inspector is allowed to stop the reader, if allowed by the moderator, while the reader is reading the documents to make comments about unclear matters. The recorder has to be constantly taking down minutes of the meeting and action items that came up. The recorder has to read over the action items and issues that came up during the inspection at the end of the inspection and give a copy of the action items to the reviewee.

A day or two after the inspection, a third-hour can be scheduled to discuss all the open issues that arose during the inspection. This time can be also used if there was not enough time during the original inspection. After the rework has been done by the reviewee to correct all the faults that were found and the issues that were brought up during the inspection, there should be a follow up to make sure the everyone is in

agreement with the changes made and to check to see it any additional reviews are necessary.

### 2.5.5 Verification and Validation

Verification and validation is the systematic process of analyzing, evaluating, and testing system and software documentation and code to ensure the highest possible quality, reliability, and satisfaction of system needs and objectives. They are the process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements [Schulmeyer, 1999].

The difference between verification and validation is simply this:

- Verification is the process of evaluating a system to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.
- Validation is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.

## 2.6 Quality Assurance in Distributed Software Development Collaboration

So far, this chapter gave an overview of the quality assurance in a general software development project setting. However, as it was mentioned in the first chapter, in more complex and larger scale projects, many people collaborate over a period of time

and over different geographical locations to create a product. This section will introduce an example of applying quality assurance in a distributed project. Specifically, it will take a look at one method of analytical quality evaluations in a distributed collaborative software development project: the distributed code inspection.

### 2.6.1 Technology

In order to sustain an effective work of the quality assurance in a distributed collaborative software development project, the project must support the necessary technologies of computer networking and associated hardware, software, services and techniques. The technologies must also be able to understand different ways people work in distributed projects. Here is a list of four different setting that people can work in a distributed collaborative project as defined by Doherty (1997):

- Face to face interaction at the same place and at the same time, e.g., meeting room technology.
- The asynchronous interaction at the same place but at different times, e.g., physical bulletin board.
- Synchronous distributed interaction at different places but at the same time, e.g., real-time document editor.
- Asynchronous distributed interaction at different places and at different times, e.g., electronic mail system.

Without the technologies that can enable successful collaborations in the above settings, it would be extremely difficult to do the job of quality assurance effectively. By having technologies that will enable an effective collaboration in these situations, the decision making process as well as the quality of the result will significantly improve.

### 2.6.2 Distributed Code Inspection Process Model

The major goal of the distributed code inspection is to provide a method that is more flexible than software inspections but more structured than walkthroughs. The model should be flexible enough to allow the code inspection in either the synchronous collaboration or asynchronous collaboration. Figure 2.4 shows the distributed code inspection process model.

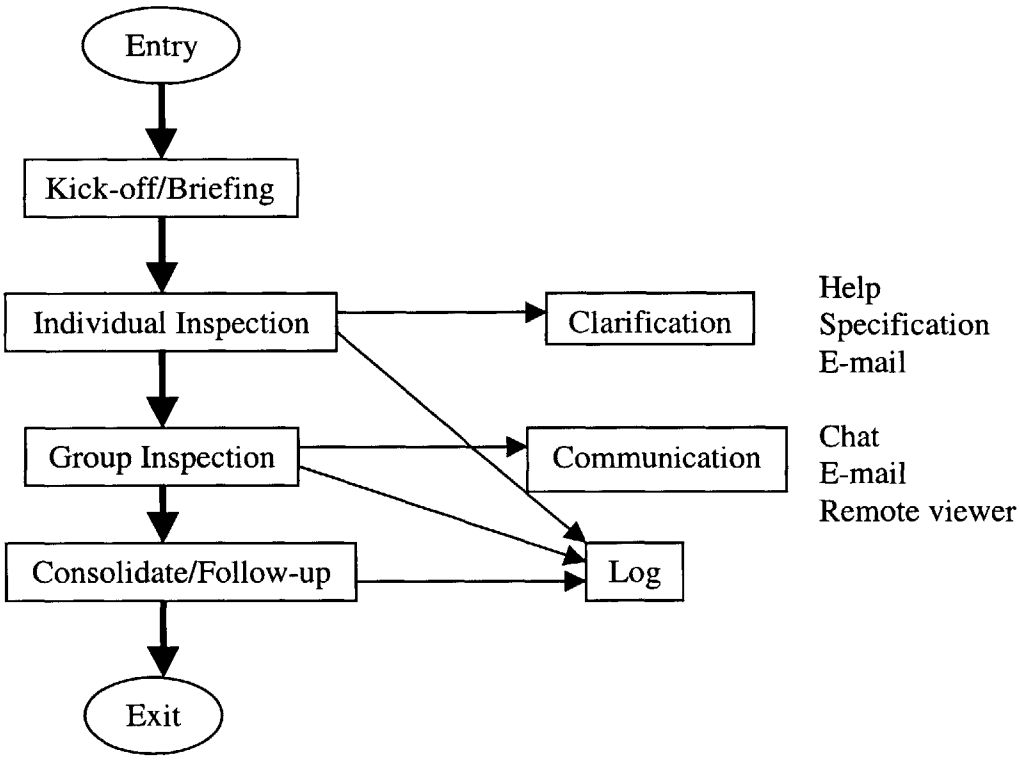


**Figure 2.4** *Distributed Code Inspection Process Model [Doherty, 1997]*

According to Doherty (1997), Figure 2.4 shows how the distributed code inspection model runs for both the asynchronous and the synchronous collaborations.

The only difference between the asynchronous and the synchronous collaborations can be found at the earlier stage of the model.

In the case of asynchronous collaboration, the process will start with the briefing document being distributed via e-mail. This document will brief the inspectors about the source code and also the length of the process. After familiarizing themselves with the materials, the inspectors can proceed with individual inspections. However, in the case of synchronous collaboration, the process will start with a kick-off meeting, which will brief the inspector about the source code and other objectives of the process. Then, the inspector can proceed with individual inspections [Doherty, 1997].

After this stage, the inspector can log any recommendations or error findings into the log book. At this point, any clarification can be sought from the moderator or the author using the facilities provided. The process then continues with group inspection where every member of the team will login at the same time and discuss the issues using the facilities provided. Further recommendations and error findings can be logged at this stage. During the last stage, the moderator will consolidate the log book and make sure that the objective set in the plan is achieved. The recommendation is forwarded to the author or the editor and the process exits [Doherty, 1997].

### 2.6.3 Summary of Distributed Code Inspection Model

This model of distributed code inspection provides an alternative method to the code inspection process in a distributed software development project. This model compensates for the fact that team members of the project are in geographically

distributed locations and might not be on-line at the same time. This model allows for the continued maintenance of quality in distributed software development collaboration. This kind of model can be used not only for the inspection method, but also the other quality measures as well to check, achieve and maintain quality in their software development project.

## 2.7 Capability Maturity Model

The Software Engineering Institute of the Carnegie-Mellon University (SEI) developed the Capability Maturity Model (CMM) based on the work on software process improvement by Watts Humphrey. It was developed in order for assessing the capability of possible future contractors for the US Department of Defense [Sanz, 1994]. Organization can use the CMM to evaluate the maturity of their current process. They could also use the model to set a goal for their development process improvement. The CMM divides organization software process maturity into five levels. Greater the maturity level, the project will have higher productivity and quality, and have lower development risk. The levels are defined by Sanders (1994) as follows:

- ***Level 1 – Initial:*** The software process is characterized as ad hoc, which means the changes are occurring as they go along. Few processes are defined and success depends on individual effort.

- ***Level 2 – Repeatable:*** Basic project management processes are established to track cost, schedule and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

- ***Level 3 – Defined:*** The software process for both management and engineering activities is documented, standardized and integrated into an organization-wide software process. All projects use a documented and

approved version of the organization's process for developing and maintaining software.

- ***Level 4 – Managed:*** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using these measures.

- ***Level 5 – Optimizing:*** Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies.

The role of quality assurance comes into play in pretty much every level. In level one, which is the initial stage, there are constant crises and things are changing as they go along. The role of quality assurance managers (QAM) in this level would be to be very flexible in checking and maintaining quality assurance. Instead of trying to plan out things in concrete ways, the QAM should try to meet and match whatever situations or events that occur as they go along. In level two, which is the repeatable stage, the QAM should try to set a few standards and rules of software quality assurance. They should try to explain the conformity with standards, procedures, and requirements. In level three, which is the defined stage, there are definite forms of the management of process improvement. The process no longer depends on the individual since they are clearly specified. The project can be carried out in the planned phases. The QAM should also define the necessary elements of quality assurance like the analytical quality evaluations such as review processes.

In level four, which is the managed stage, there are quantitative basis for planning and decision making. The QAM needs to develop the software quality management, which is an analysis of the managerial structure that influences and controls the quality of the software in a software quality assurance activity. They should have the definite

structure in which clearly defined tasks and responsibilities can be found. They should also develop quantitative process management by incorporating quality measuring process like inspections. In level five, which is the optimizing stage, there are not only management of process, but also optimization of the process itself. The QAM should develop ways to check and maintain the quality of the project through preventing defects. They should find methods and strategies to predict possible faults that might occur and resolve them before they really occur.

The CMM reflects the business needs of large organizations working in the US defense software sector and it specifically designed within itself a particular order for improvement actions. The CMM model may not reflect the business needs of a distributed collaborative software development project and the model's order for improvement actions might not be appropriate for the distributed project. According to the CMM, a process cannot be implemented before the preceding process has been implemented successfully. There are also many elements within the process that needs to happen before it can move unto the next process. However, in a distributed collaborative project it is common for individual elements in a process to be delayed significantly due to difficulties arising in distributed collaboration. It will be inefficient to wait for every element in one process to be completed before moving to the next process. It would be more productive and efficient for the project to just define some of the key elements in the process and as long as those are completed, even if other minor elements are not, then the project should be able to move into the next process.

## 2.8 Conclusion

This chapter gave a general overview of quality assurance in the software development project. It discussed why it is so important to check for and maintain quality from the very beginning of the software development project and continue to do so through out the process. It also touched upon the software development cycle, the role and the description of each phase, and their integration.

Software quality section discussed how costly and fatal it would be not to find and fix errors and faults in the project in the early phases of the project compared the latter phase. It simply stated that the cost of correcting the errors increases exponentially with the time that an error remains in the project cycle. It is extremely important, therefore, to find and fix the errors and faults when they come up to ensure the best result for the product and the project.

It then went on to discuss the quality factors and criteria which one should establish to measure quality in a given project. With quality factors and criteria, a number of technical reviews should take place to carry out analytical quality evaluation of various points in the project. Verification and validation should also be done constantly to make sure the maintenance of the quality in the project is done.

Quality Assurance in Distributed Software Development Collaboration section talked about applying quality assurance to distributed software development collaboration. It gave an example of an analytical quality evaluation that can be applied to a distributed collaborative environment, particularly the inspection. It stressed the fact

that it would be extremely difficult to carry out the task of the quality assurance managers of a given distributed collaborative software development project if right technologies are not available. It also introduced the distributed code inspection process model, which provides flexibility and extendibility to enhance the software quality in distributed software development collaboration.

The last section (2.7) discussed the Capability Maturity Model (CMM). It discussed what the model represents and the different levels that the model consists of. It also discussed how the role of quality assurance comes into play in each level of the model. It also argued how the distributed collaborative software development project will affect the CMM. It stated that the CMM is designed for a specific type of organizations and that it may not be suitable to use it in the case of a distributed project and some modifications should be made in order to implement it.

# Chapter Three

## Intelligent Electronic Collaboration Project

Intelligent Electronic Collaboration (ieCollab) consisted of 34 graduate students from the Massachusetts Institute of Technology (MIT), the Centro de Investigacion Científica y Estudios Superiores de Ensenada (CICESE) , and the Pontificia Universidad Catolica de Chile (PUC). Diverse group of students from these three universities came together in a single collaboration to participate in the distributed software development project.

This project, ieCollab, was to bulid an internet-based collaborative application service provider for communicating information and sharing software application in a protocol-rich Java meeting environment. By creating and managing virtual team, ieCollab's collaborative solution will offer organizations new ways to leverage off-site expertise, improve communications, and reduce project costs and duration. Some of the users of iecollab may be virtual teams in automotive, aerospace, construction, defense, and software industries. ieCollab's internet based meeting tools solve the problem of personnel relocations with reliable forums for communication among distributed teams.

## 3.1 The ieCollab Team

The team, ieCollab, consisted of 23 students from MIT, six students from CICESE, and five students from PUC. Each of these students has pursued degrees in information technology and computer science with undergraduate and professional backgrounds ranging from engineering to business. Each student also brought with him or her diverse ethnic backgrounds into the project. In the team there were several represented nationalities ranging from India, China, Hong Kong, Indonesia, Korea, Mexico, Chile, to USA. Students were allowed to choose their roles among the following ten roles:

- Business Mangers
- Marketing Managers
- Project Managers
- Requirement Analysts
- Designers
- Programmers
- Testers
- Configuration Managers
- Quality assurance Managers
- Knowledge Management Managers

Each team tried to consist of at least one student from each university to encourage working with remote partners in a distributed environment and acquire experience from it. Every student was assigned two roles in the project. Usually, the second role was a programmer. Every student worked mainly as his or her primary roles required. However, during the programming phase of the project, everyone whose second role was a programmer worked exclusively in programming.

Since majority of students had no prior experience working in their assigned roles or in the software development project, it was very important for them to acquire knowledge about what their tasks and responsibilities are in the role. The definitions of each of the roles are as follows:

***Business Managers and Marketing Managers*** – The major duty of business managers and marketing managers was to define the scope of the project, the buyers and users, and ultimately the purpose of the project. They had to define such things as the market/customers, business concept, competitive advantage, and finances of the project. They were responsible for developing the business plan for the project.

***Project Managers*** – The major responsibilities of project managers were to plan, monitor, and control the project, to coordinate all activities, and to sustain communication and provide support. They were responsible for scheduling meetings, coming up with agendas for the meetings, setting up due dates for deliverables, and keeping the team members aware of where they are supposed to stand in the project. They were also responsible to develop the project manager plan and to allocate resources. The major role of project managers was to be "Responsible for providing overall leadership to the team," [Yang, 1998] and to create project updates through the project cycles.

***Requirement Analysts*** – The major responsibilities of requirement analysts were to translate the business and marketing managers' desires into specifications, to represent

software behavior, to define the software's main functions, and to create requirement analyst plan.

***Designers*** – The major responsibilities of designers were to communicate the requirements of the software, as defined by the analyst team, to the programming team, and to create a plan containing all software information that was pertinent to the programmers and testers.

***Programmers*** – The major responsibilities of programming leaders were to define the two programming groups in the project, to create a high-quality, well-documented, concise software code that adheres to the design plan, to assign individual and group task relevant to the programming, and to provide necessary knowledge and information.

***Testers*** – The major responsibilities of testers were to check correctness and consistency between object-oriented analysis and object-oriented design, to develop test cases, to perform unit test (functions of each class), to perform integration test, and to perform validation test by verifying the output of the system.

***Configuration Management Managers*** – The major responsibilities of configuration managers were to help programmers to organize, control, and keep track of their work by coordinating software generated from different programming teams, promptly informing software status, and being the safety net. They were also responsible for preventing simultaneous updates.

***Quality Assurance Managers*** – The responsibilities of quality assurance managers were to coordinate audits and walkthroughs, to participate in reviews and inspections, and to review all plans submitted by each team. They were responsible for monitoring software quality by alerting project management team of inadequacies and making sure the teams follow their responsibilities. They also monitored the development process by checking if each activity was completed adequately before moving to the next activity.

***Knowledge Management Managers*** – The major responsibilities of knowledge management managers were to maintain the project web-site and check it weekly, to compile all project documents, to make documents from previous projects available to all, and to create user's guide and technical guide.

The major difference between this year's team and the last year's team was the addition of new Business/Marketing managers. By adding Business/Marketing managers to the team, the project was able to incorporate a sense of real business operation. This addition also introduced the concept of entrepreneurship, that software needs buyers and users and that ignoring these consumers in the software development process in to ignore the major purpose of the endeavor. ieCollab students were able to learn to determine an appropriate market for their product and present their business idea to venture capitalists in order to learn to design, develop, and sell in nine months time. Figure 3.1 shows the project organization of the ieCollab.

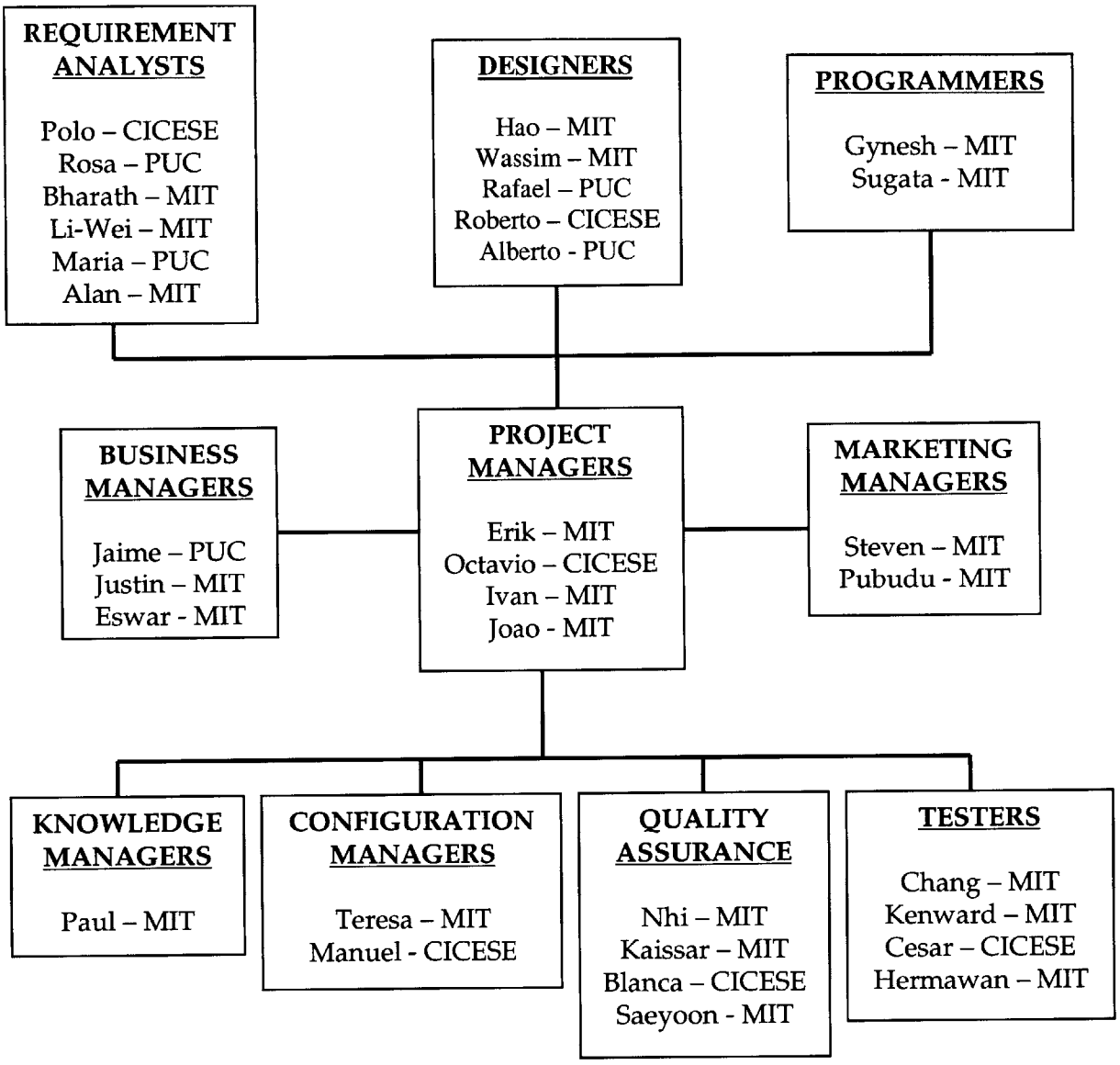


**Figure 3.1** *ieCollab Team*

## 3.2 The ieCollab Project Objective

The main aim of the ieCollab Project was to provide the environment in which students in geographically distributed teams can work together to achieve a common goal and try to learn every aspect and process of realizing a single project. The teams of students were given the responsibility of improving distributed collaborative software

tools that allow synchronous and asynchronous collaboration. In order to accomplish this goal, every student had to learn about topics ranging from software development, collaboration and organizational strategies, entrepreneurship, collective memory, and technology. These topics were mainly discussed and defined in class by the Professors (Peña-Mora, Favela, and Fuller).

In the software development part, the students learned about both the design and implementation of team-based large-scale software development project. The students worked on a software development plan, which covered project management, requirement analysis, system architecture and design, quality control, programming, configuration management, and testing. The students also learned about the object-oriented methodology and object-oriented programming that was used during the implementation phase [1.120, 1999].

In the collaboration and organizational strategies, the students obtained insights into virtual corporations, distributed operations, organizational culture, and communication. The focal point of the collaboration and organizational strategies was to provide such an environment so that it would be unavoidable for students to experience distributed collaboration and learn about the effective ways of communicating with your team members in remote locations in order to accomplish the team's goals and tasks [1.120, 1999].

In the entrepreneurship aspect, the students had to learn about how the software needs buyers and users, and ignoring these consumers in the software development process is to ignore the major purpose of the endeavor. The students also had to learn to research, analyze, and determine an appropriate market for the product and present their ideas according to what they found [1.120, 1999].

With the collective memory part, the students were able to realize that when producing software, it is important to store all the information regarding the software development process. The reason for this is that other people who might not have any relations to the project may look at the experience and have the background on which to build upon it if they have the desire to in the future. They can learn from the past experience and try to be more efficient in their projects [1.120, 1999].

On the subject of technology, the students learned that the technology is only one part of the entire software development process. It does not matter how much technology you have or how advanced your technology is. Without other factors that constitute the project such as collaboration and design, the project will not succeed. However, it is also very important to keep up with the evolving technology and think about what new opportunities that the new technology can bring that could not have been possible before [1.120, 1999].

This was the main objective of the project. Of course, one of the main goals was to successfully finish the product. However, the goal that had even greater significance and relevance to this project was to learn about all the factors and components that make

up the process of the software development. Learning about how things are done, carried out, and executed was a greatest objective.

## 3.3 The Process Model

The process model used in ieCollab started out as a waterfall model, which is a model in which a project progresses through an orderly sequence of steps from the initial software concept through system testing. The project in a waterfall model holds a review at the end of each phase to determine whether it is ready to advance to the next phase [McConnell, 1996].

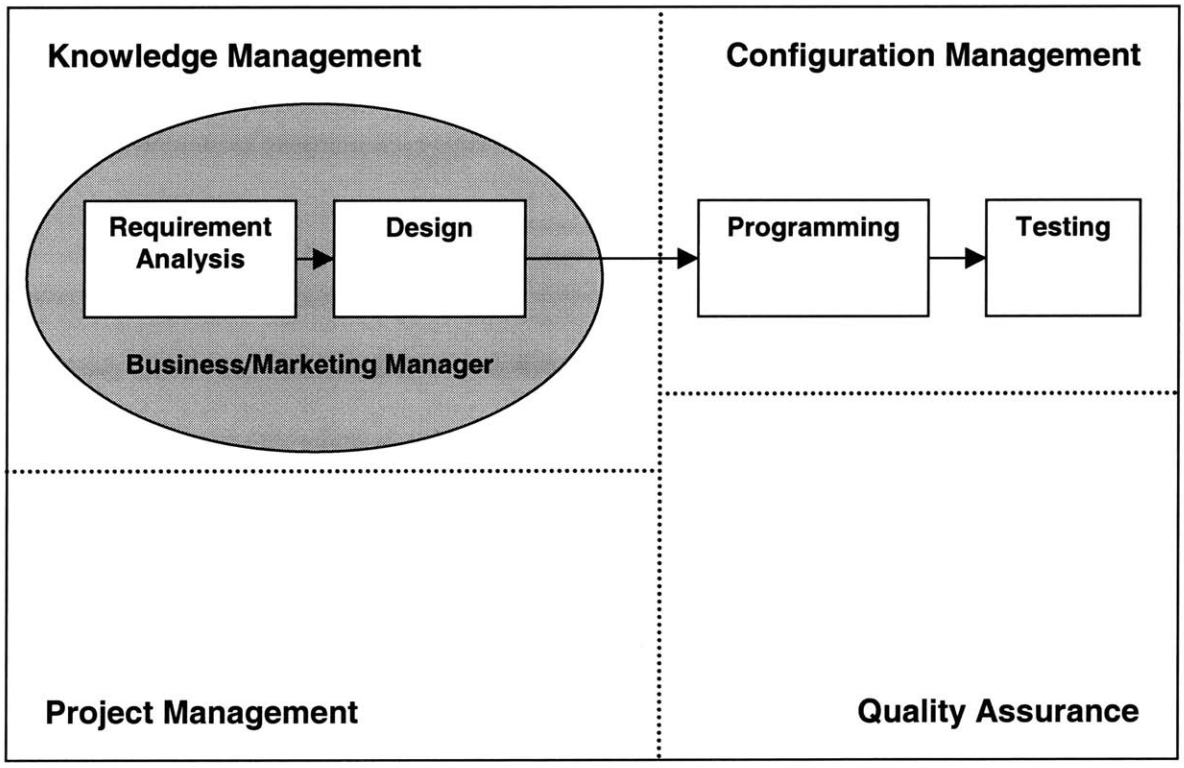


**Figure 3.2** *Waterfall Model for ieCollab*

The original intent of the ieCollab was to follow the waterfall model: when the requirement analysis phase is completely done then it would move on to the next phase of

design and so on. Additionally, as it can be seen in the figure 3.2, the business and marketing management team would have guided requirement analysis and design phases to ensure that everything was going smoothly and the teams met the set definition and market demand.

The major advantage of using the waterfall model was the ability to check and make sure that one phase is done before moving to the next phase. It is an orderly, step by step process. For a project like ieCollab that contains many students who have no experience in software development before, it seemed like a good idea to have review of at the end of every phase. Also the fact that ieCollab was a distributed project was a good reason to follow the waterfall model. Since it would be difficult to efficiently keep track of the tasks that need to be done over the geographically distributed environment, it would be beneficial to have a check point at the end of each phase to make sure that the project is moving along the right direction. It also would have been easier for the Quality Assurance Team to keep track of the quality of the work that was being done by each team in each phase. By waiting for one phase to be completely finished before moving into the next one, the Quality Assurance Managers (QAM) can concentrate on checking, achieving, enforcing, and maintaining the quality in that specific phase.

However, as the project progressed and problems started to arise, there had to be a change in the process model that the project followed. This was also the drawback of using the waterfall model. Since, in the waterfall model, the succeeding phase has to wait till the current phase is completely finished; sometimes the succeeding phase has to wait a really long time if the current phase falls behind the schedule. As problems with delay

and revisions continued, it was inevitable to move on to the next phase of the project even if the current phase was not completely finished. Therefore, the design team began its operation as requirement analysis team went back to make more changes to their documents. At this point, the project was following the incremental model, which is a model in which each linear sequence produces a deliverable increment of software [Pressman, 1997].

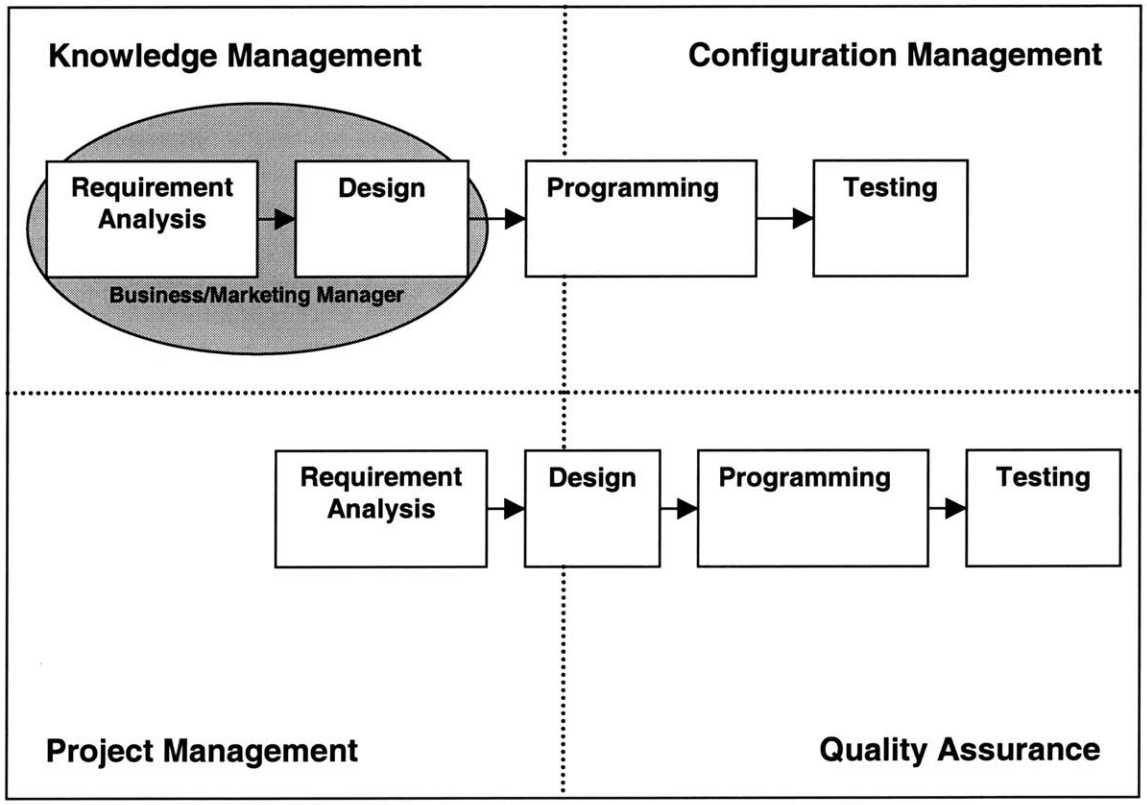


**Figure 3.3** *Incremental Model for ieCollab*

As it can be seen in the figure 3.3, the project now was proposing to operate in several different increments. Even though the entire requirements were not completed, for some parts of it that were finished, the design team took them and went ahead and started to develop models. For the parts that were not finished, the requirement analysis

team would finish and give them to design team to develop as the second increment. This change in process model was needed since the project was taking far too much time in the beginning phase of the development and falling behind in the schedule.

The advantage of using the incremental model was the ability to be more efficient with the schedule and be less risky by breaking the project into smaller sub-projects. It was a good option for ieCollab to employ since the project was divided into smaller parts like meeting management and transaction management, and the project falling behind the proposed schedule. The team members learned that the incremental model could be a good model for a distributed software development project since it is extremely hard to coordinate things in a distributed collaboration and have every task of a certain stage be finished by certain time. Even in an undistributed project, meeting the schedule is the most difficult aspect. If the teams in the distributed project are evenly divided across the geography, then using the incremental model, the project may be divided into sub-projects and have each region responsible for one sub-project to increase productivity and efficiency.

Following the incremental model laid a bit of challenges on the QAM's part. In the waterfall model, the QAM was able to focus and give priority to the current activities that were going on in the project. However, with this incremental model, the QAM had to be concentrating on many different activities at once, checking qualities in the requirement analysis phase at the same time it was enforcing qualities in the design phase. This was the main drawback of the incremental model. Not only for the QAM but for the other teams as well, having many different activities going around at the same

time was a complicated process that required a great deal attention and well coordinated management in the activities and operations.

## 3.4 The Challenges and the Benefits

One of the major challenges that ieCollab team faced in its operations was the inexperience of the students. It was mentioned before that most of students were graduate students majoring in information technology. However, the backgrounds of great number of students were in civil and environmental engineering. They came into the project with very little knowledge and experience in software engineering. Most of the students were not familiar with the software development process, and their programming skills were inadequate. This project required the software development in JAVA programming language, but most students had no experience with JAVA. In order to overcome this obstacle, lectures were given every week on Tuesdays and Thursdays by professors from three universities to train and familiarize students with the concept and process of software development. The inexperience in JAVA programming required more of personal effort. The students had to train themselves by taking the class, 'Foundation of Software Engineering', in the fall semester.

Another major challenge was the distributed collaboration. It was mentioned before that every team consisted of students from at least two different universities to encourage working in distributed environment. This was one of the main objectives of the project, yet at the same time one of the greatest challenges of the project. When there were lectures, videos and audio were used to enhance a better learning environment. However, when the time came for the students to collaborate about their own team tasks,

there was a heavier use of emails and chat tool. There soon existed frustration, impatience, and hopelessness. Everyone realized how hard, difficult, and cumbersome it was to work with partners they can only see in writings and voices. Many students assumed that the reason for the frustration and ineffectiveness was due to the being too far away from the partners in Mexico and Chile. However, a strange phenomenon was that there seemed to exist almost the same amount of frustration and difficulties collaborating with students at a same location. So, the challenge was not with remote partners but with collaboration itself. It is expedient for any type of projects to succeed, team members must learn to work together, be patient, and be cooperative.

The challenges, that ieCollab Project faced, which mentioned above were also the benefits that came out of it. One of those benefits obtained was learning how to work in teams of people who have different backgrounds and different experiences. As it was stated before, many students were coming from a diverse academic and ethnic background with varied length of work experiences. The students were able to learn the amount of effort and work it takes to make a team out of those people and collaborate to achieve success. Especially in this day and age of working world, where everything has been globalizing, it is unthinkable that people will only work with a very small pool of people with just the similar backgrounds. The working world requires people to work in any kind of environment with any type of people and expect success from them. ieCollab Project has provided the students with the chance to experience the real working world before the students actually entered it.

Another benefit that came out of this project was the first hand experiences of how the technology has developed and is capable of doing. The students in the project have heard about the technology that enables the distributed collaboration but have never really taken part in experiencing it or applying it to a given task. Through this project, the students were able to learn how to apply the developing technologies to enable a challenging task, in this case, the distributed collaboration.

## 3.5 Conclusion

This chapter introduced the ieCollab Project. It gave an overview of the project and discussed what the project was basically about and what the project was proposed to do. It also talked about the project participants and how they are divided into different teams. The Team Section also discussed what the role and responsibilities of different teams were.

The Project Objective Section discussed that the main objective of the project was designed more to learn about the other factors involved in software development process such as collaboration and organizational strategies, and entrepreneurship, than the actual software development itself. The most important thing that was stressed throughout the project was to learn and experience the process of the distributed software development collaboration.

The Process Model Section described two types of process model that was used in the project. It was interesting to note that depending on the circumstance that the project

was in and the problems it was facing, sometimes it was effective and efficient to change the process model of the project to achieve success.

Finally, the last section discussed the challenges faced by the project and the benefits gotten from the project. Mostly, the problems focused on the lack of experience and knowledge on the students and also the problems rising due to working in an unfamiliar environment such as a distributed collaboration environment. However, when the challenges are great, so are the lessons learned from them. From working in ieCollab Project, the students were able to learn to work together with very diverse group of people in a very different environment than usual. The experience that they have obtained from the project in distributed collaboration and applying technologies to make the collaboration more effective will be beneficial as they go about working in the real working world.

# Chapter Four

## Role of Quality Assurance in ieCollab

Applying quality assurance to Intelligent Electronic Collaboration (ieCollab) Project was very different and unique from other projects. First of all, ieCollab Project was a distributed software development collaboration project. There have been many cases of applying quality assurance to software development projects but very few cases of applying quality assurance to software development projects in a distributed collaboration. The second thing was that the Quality Assurance Managers (QAM) of ieCollab had no prior experience of working as a quality assurance manager. The knowledge acquired was from the lecture about quality assurance given by a professor during class. The lecture outlined some of the responsibilities of the QAM of ieCollab Project as follows [1.120, 1999]:

- Monitor both product process and compliance to good practice
- Highlight problems in the early phases
- Produce statistical results of problems and provide guidance for solutions
- Develop a good plan for resolving problems, identifying by when and whom a problem should be resolved
- Assure that the software is compliant to user requirements.

Besides the knowledge acquired from the lecture, the QAM also found, gathered, and distributed all the IEEE Software Engineering Standards for each team in the project that each team should comply with. However, during the research, it was difficult to find other materials that showed directly and clearly how to manage quality assurance in a distributed environment.

## 4.1 The Challenges

Applying Quality Assurance to ieCollab Project presented a big challenge to the Quality Assurance Mangers (QAM). As mentioned above, no one on the Quality Assurance Team had any experience ever working as a Quality Assurance Manager. Not only that but also no one has ever seen how the quality assurance work in a software development project. There were many books and references about quality assurance in software development projects. However, there was hardly any information on managing quality assurance in distributed software development projects.

Another challenge on top of the inexperience was the working in a distributed environment. The Quality Assurance Team was made up of three students from Massachusetts Institute of Technology (MIT) and one student from Centro de Investigacion Científica y Estudios Superiores de Ensenada (CICESE). The Quality Assurance Managers were not sure not only how to manage quality in a distributed software development collaboration project, but also how to collaborate amongst themselves in the distributed quality assurance team collaboration.

With the presence of these challenges, the QAM decided to concentrate on the development of the Quality Assurance Plan (QAP), which would become the source of guide for Quality Assurance team to plan, prepare, and execute the necessary steps to achieve quality in ieCollab Project. The QAM also decided to monitor each and every activity during every phase of the project by having Analytical Quality Evaluations, such as Walkthroughs, in order to maintain quality throughout the project.

## 4.2 Quality Assurance Plan for ieCollab

The quality assurance plan is the central aid for planning and checking Quality Assurance of a given project. It contains all deliberately chosen quality assurance measures for a software project, and consequently it is the written proof of quality control. [Wallmuler, 1994] The plan will serve as the specific guidelines for all the line operations, as well as providing a base line against which performance can be measured. Therefore, the quality assurance plan must cover the total life of the product or service for its conception until the end of its life [Mills, 1981].

### 4.2.1 The Purpose

The purpose of the Quality Assurance Plan is to describe a methodology for quality assurance and quality control during the software development cycle of ieCollab. A clearly defined procedure ensures pre-defined quality standards that are met in the final product. In addition, constant central checking during development ensures a smoother

transition from one design phase of product version to another. Finally, this document will serve as a foundation for next year's quality assurance team in developing methodology for next year's product.

This Quality Assurance Plan (QAP) presents the quality assurance methods used in the development of ieCollab during the 1999-2000 product development cycle. Specifically, quality control is applied to Transaction Management (Version 1) and Meeting Management (Version 2) of ieCollab. Collaboration Server (Version 3) and Application Server (Version 4) will be developed in the next product cycle.

The software development cycle is composed of five phases. Each phase is inspected by Quality Assurance Managers (QAM) for deliberately chosen quality measures. These measures are found in the QAP. The five phases and QAM's involvement in each cycle are described in Table 4.1 (a) and (b) below.

**Table 4.1(a)** *Tasks of the QAM during the Software Development Process [Cruz, 1999]*

| PHASE | FUNCTIONS |
|---|---|
| **Concept Exploration** | • Define the QAM's responsibilities and the QAP.<br>• Review the software development plan and audit procedures done by the Project Manager.<br>• Participate in the meeting with the client to obtain the software requirements.<br>• Implement the QAP |
| **Requirement Analysis** | • Review the Project Manager's plan and develop audit procedures.<br>• Participate in the requirement specification review.<br>• Review tools, techniques and methodologies used in the software development process. |

**Table 4.1(b)** *Tasks of the QAM during the Software Development Process [Cruz, 1999]*

| PHASE | FUNCTIONS |
|---|---|
| **Design** | • Participate in all the design reviews.<br>• Monitor the procedure for correcting design errors.<br>• Review the final design document.<br>• Review the final operator manual.<br>• Review the use of methodologies in design.<br>• Review preliminary test plans. |
| **Programming** | • Participate in some code inspections.<br>• Monitor use of defined tools, technologies and methodologies.<br>• Review final test plans.<br>• Witness of all the tests done.<br>• Audit code errors for correction.<br>• Audit change request record tracked by the Configuration Manager. |
| **Operation / Maintenance** | • Review the changes proposed by the user/customer.<br>• Audit the change record.<br>• Audit for correction of customer problems.<br>• Inspect updated documents: Requirements Specification, Preliminary and Detailed Design, Operator Manual, Error and Changes record |

### 4.2.2 The Management

This section presents the organization structure that influences and controls the quality of the software. The QAM work directly with other quality monitoring teams, including Configuration Management, Project Management, Testing, and Knowledge Management (Figure 4.1). The QAM collaborate with team members in Requirements Analysis, Design, and Programming in supporting and enforcing quality standards (Figure 4.2).

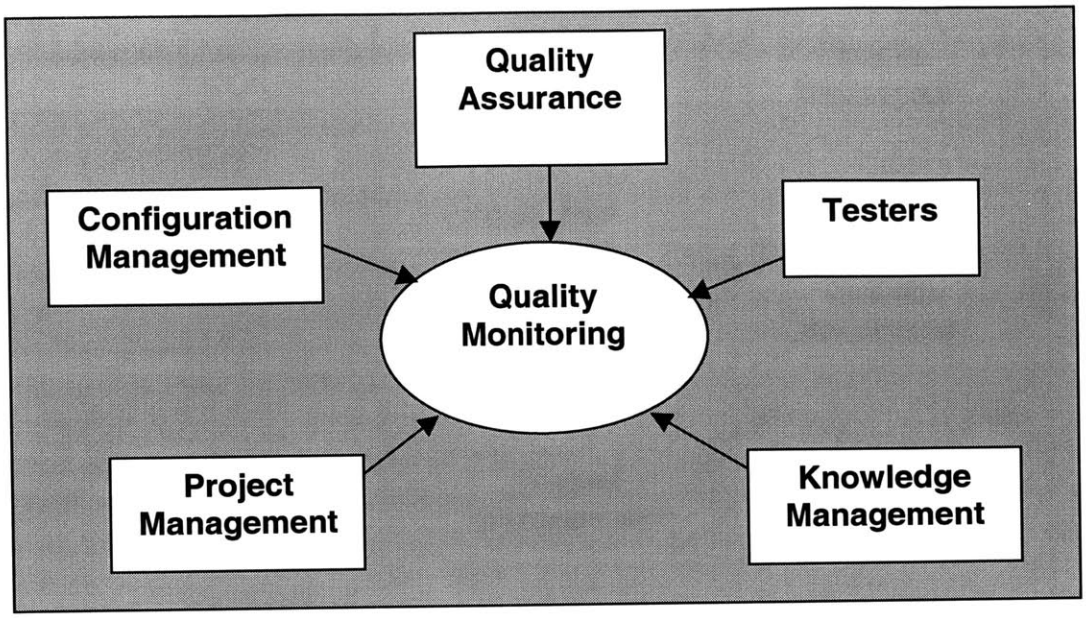


**Figure 4.1** *Quality Monitoring Organizational Structure*

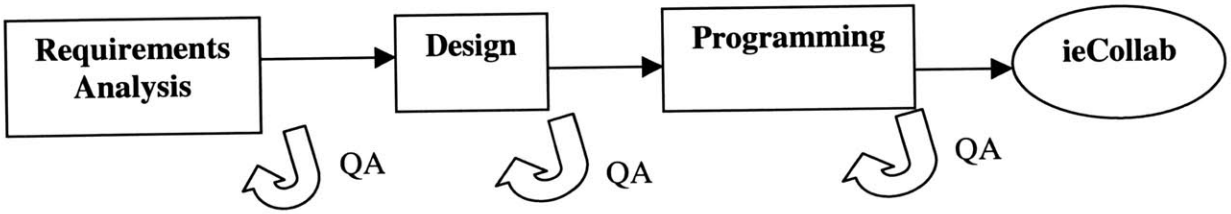


**Figure 4.2** *Quality Support and Enforcement Organizational Structure*

#### 4.2.2.1 Tasks

Throughout each task, QAM assumes a lead role in ensuring that the task is complete and enforces the resolution or effect of each task. Table 4.2 (a) and (b) describe the collaboration of QAM with the teams involved in software development. In each interaction, QAM assumes an active role in facilitation interaction.

**Table 4.2(a)** *Interaction by Quality Assurance Managers and Primary Roles [Cruz, 1999]*

| ROLE | INTERACTION |
|---|---|
| **Knowledge Management** | • The QAM will work closely with the Knowledge Management Managers to make sure that the documentation created during the project meets the software standards for documentation.<br>• The QAM will verify that the Knowledge Management Managers have created a repository for all the documents produced throughout the project for easy access. |
| **Testing** | • The QAM will make sure that the Testers' Plan is adequate to the project and is being performed through all the development phases.<br>• The QAM will work closely with the Testers to make sure that the software product is free of errors.<br>• The Testers will test the code developed by the programmers and will trace back to the requirement documents to make sure that the deliverables accord with the client's requirements. |
| **Configuration Management** | • The QAM will make sure that the Configuration Management Manager's Plan is adequate to the project and is being performed through all the development phases.<br>• The QAM will review the changes, errors and configuration record, to ensure that an appropriate error log is kept through every phase of the development process, that the changes are properly implemented, and that the baselines are saved and products are not lost. |
| **Project Management** | • The QAM reviews the development plan to ensure that it is created and followed. |
| **Requirements Analysis** | • The QAM reviews the requirements to ensure that accurately and completely represents the expectations of the customer. The requirements must also be clear enough to everybody in the developer's group, especially to the designer. |
| **Design** | • The QAM reviews the design document to ensure that the designer has selected the appropriate methodology and that the final product of design document meets the performance, design and verification requirements. |

**Table 4.2(b)** *Interaction by Quality Assurance Managers and Primary Roles [Cruz, 1999]*

| ROLE | INTERACTION |
|---|---|
| **Programming** | • The QAM reviews the programming and implementation of the system to ensure that the code produced meets the stated requirements specification, and is reliable, efficient, easy to understand by human users, easy to be verified by execution, and easy to be read and modified by a software mainatainer. |

### 4.2.3 Software Documentation

A major portion of quality assurance is the revision of all the documents that are being produced in order to make sure that they are clear, and understandable, that they follow the standards, and are inline with what the customer asked to be developed. The QAM have conducted analytical quality evaluations to the documents that are stated below:

***Requirement Specifications***

A document containing a collection of requirements/specifications gathered from interviewing the client. These requirements must be defined clearly so that the people involved in the remaining stages of the development software life cycle can implement. A walkthrough will be conducted to verify the correctness of the Software Requirements Specifications.

***Design Specifications***

This document serves as the bridge between requirements and the actual implementation by the programmers. The designers must make a good design of data structures, the

architecture of the software to be built and interface modules. This document is very important because the better the design documents is made, the easier it will be for the Test Engineers to come up with test cases. Audits throughout the development of the Design/Software Specifications will be conducted by the QAM. At the final stage of the Design Document, a walkthrough of inspection will also be conducted by the QAM for verification purposes.

***Project Manager Plan***

The plan must have a vision and mission of the project. It must also have a work plan that contains a schedule of events to accomplish both of them. A walkthrough will be conducted by the QAM to check compliance with the standards in writing the Project Manager's plan.

***Configuration Management Plan***

The plan must state how the Configuration Management Manager will account for version control, and change control of documents and code produced by the team members. A walkthrough will be conducted by the QAM to check compliance with the standard in writing the Configuration Management Plan.

***Test Plan***

This document contains the methods and means by which it is proven that the design conforms to the requirements and the source code conforms to the design and the

requirements. A walkthrough will be conducted by the QAM to verify compliance with the standards in writing the Test Plan.

***Testing Report***

The testers will create reports with the result from executing the test plan (revising documents and code). Audits will be conducted by the QAM after every revision to verify the process the Testers are implementing while revising the code.

***Knowledge Management Plan***

The plan must state the standards that the team must follow to create HTML, Word and PowerPoint documents. The Knowledge Management Managers must also state how they will be managing ieCollab's web page. A walkthrough will be conducted by the QAM to verify the standards.

### 4.2.4 Software Engineering Standards

The standards that were used in the project documentation are the IEEE Standards for Software Engineering. These standards are stated below.

| | |
|---|---|
| 730-1998 | IEEE Standard for Software Quality Assurance Plans |
| 828-1998 | IEEE Standard for Software Configuration Management Plans |
| 829-1983(R1991) | IEEE Standard for Software Test Documentation (ANSI) |
| 830-1998 | IEEE Recommended Practice for Software Requirements Specifications |
| 1008-1987(R1993) | IEEE Standard for Software Unit Testing (ANSI) |
| 1016.1-1993 | IEEE Guide to Software Design Descriptions (ANSI) |
| 1042-1987(R1993) | IEEE Guide to Software Configuration Management (ANSI) |

| | |
|---|---|
| 1058.1-1987(R1993) | IEEE Standard for Software Project Management Plans (ANSI) |
| 1063-1987(R1993) | IEEE Standard for Software User Documentation (ANSI) |

### 4.2.5 Analytical Quality Evaluations

There are many ways to perform Analytical Quality Evaluations in a given project. Most of these approaches involve a group meeting to evaluate a product. However, some reviews do not require a review meeting by the group. ieCollab Project will incorporate the following four types of evaluations:

- Peer Reviews
- Walkthroughs
- Audits
- Inspections

**Peer Reviews**

Peer reviews are the most informal of all technical reviews. Peer reviews are mainly used for checking the source code to detect errors before execution or compilation. Issues that should be revised while doing peer reviews are correctness, misuse of variables, omitted functions, poor programming practices and redundancy.

**Walkthroughs**

A walkthrough is an informal review that evaluates the Requirement Analysis, Design, Code, Testing and Integration of a software project. The goals of this informal review should be determined prior to conducting the walkthrough and are identified in the notice announcement of a walkthrough. The main emphasis of the walkthrough is to review the

process of the different phases of the software life cycle. In a walkthrough, the primary participants are the moderator, recorder, reviewee, and two to three reviewers.

### Audits

Audits serve to insure that the software is properly validated and that the process is producing its intended results. In an audit, the review leader is responsible for validating changes in the report. The Quality Assurance Managers will perform audits and they will be responsible for sending an Audit Notification to all the participants in the audit.

### Inspections

Inspections should be presented as a more formal approach that can be viewed more as work product reviews. Inspections require a high degree of preparation of the review participants, but the benefits include a more systematic review of the software and more controlled and less stressed meeting. Software formal inspections are in-process technical reviews of a product of the software life cycle conducted for the purpose of finding and eliminating defects. The major difference between walkthroughs and inspections is that an inspection process involves the collection of data that can be used to feedback on the quality of the development and review process.

## 4.2.6 Quality Factors and Criteria for ieCollab Project

As it was briefly mentioned in Chapter two, ieCollab did not use the quality factors and criteria system that was suggested by either Evans or Bucley and Poston. Instead, it used the system based on the study of McCall, Richards, and Walters. The

reason why the QAM of ieCollab decided to use this system instead of the other two was that this system had the reputation of being the best documented system, "both in terms of the definitions, trade-offs, and enhancement between Factors and Criteria and in suggesting the impacts on life-cycle development." [Vincent, 1988] The fact that the last year's project team also used a quality factor and criteria system based on this one was an added incentive. The table of explanation of the trade-offs between software quality factors was already introduced in Chapter in Table 2.3. The table that lists the definitions of the quality factors from the system proposed by McCall, Richards, and Walters that were used in ieCollab can be found in the appendix (Appendix 8).

## 4.3 Quality Assurance Actions on ieCollab Project

With limited knowledge, experience, and numerous challenges, the Quality Assurance Managers of ieCollab had a difficult time organizing and carrying out their responsibilities. With the frequent advice from the professor, the QAM started to check and maintain the quality of the ieCollab Project from the early stage of the project. The QAM first created a template of how each and every document, which is going to be posted or submitted, should look like based on IEEE standards and asked every team to follow that template.

The QAM asked every team except for the team that has submitted a document to read and go over the new document, and make comments about the document. The QAM also asked for the team that has submitted a document to send out a message to every team notifying that a new document has been posted. This was done not only to ensure that any document that gets posted go unread or unnoticed by the teams, but also

to check for the quality of document by every member of the project. If something that is inconsistent or incorrect is found then it can be noted back to the team that has submitted that document to go back, check, and fix or give reasons for not fixing. While the QAM was asking for the rest of the team to make comments on the submitted documents, the QAM also was checking the documents to see if they comply with the IEEE standards and give notification if they didn't comply. The Comments were made through emails or posting them on the ieCollab Project Web Repository in the beginning. In the latter stage, the comments were made and submitted using the Comment Report form. (See Appendix 1)

The QAM conducted a number of walkthroughs over the course of the project. Walkthroughs were usually conducted for a specific team during a specific phase for a specific document. The Project Managers set and announced the date of walkthroughs. When the Project Managers notified the teams of the walkthrough of a specific document by a specific team, the QAM started making agendas for the meeting. The QAM started assigning one person from each team to make comments at the walkthrough. The QAM also appointed who would be the moderator and the recorder. The form for the walkthrough action item list can be found in the appendix.

The QAM also conducted audits to validate documents and any changes that were made to the documents. The QAM performed audits in the project to check that the work being done by each team was of the high quality. The QAM conducted the audit of the two requirement specification documents and the two design documents for meeting management and transaction management. The audit was held to check for

inconsistencies between the document, make them consistent, and validate them. The form for the audit checklist can be found in the appendix.

The QAM also came up with Quality Control in Software Development: Programming Standards (See Appendix 2), for inspection purposes, to check and maintain quality in programming that was done in the project. This was basically the methods used to inspect the codes that was written during the programming stage and check if they maintain high quality complying to the standard, the requirements, and the design of the project. The form for the general inspection checklist can be found in the appendix.

The QAM also submitted the validation plan for the client interface and requirements for ieCollab. (See Appendix 3) This validation plan was submitted to validate and verify the requirements of ieCollab and check if they meet the Requirement Analysis Specification that was submitted at the earlier stage of the project. This document tested the Client Interface for requirements outlined in Requirement Specification for ieCollab V2, Meeting Management. Subsequent validation documents of Transaction Management and Meeting Management for the Requirements Specification were decided to be created as those layers are completed. In addition, validation of Transaction Management, Meeting Management, and Client Interface for Design Specifications were set to be presented in separate documents.

## 4.4 An Example of Analytical Quality Evaluations in ieCollab

This section will show an example of an audit that was performed by the QAM in the ieCollab Project. The audit was performed on March 10, 2000 with the help of the Project Management team to validate two requirement specification documents and two design specifications documents. The audit found faults and inconsistencies in the Requirement Specification Document for Meeting Management, the Design Specification for Client Interface, the Design Specification for Meeting Management, and the Design Specification for Transaction Management.

***Requirement Specification Document for Meeting Management:***

Page 5, second bullet under the roles of normal workgroup member:

- It should be 'list members of workgroup and/or users' to be consistent with the sequence diagram shown on Page 8.

Page 8:

- Add another label in the sequence diagram for Relinquishing work group membership to be consistent with the roles specified earlier.

Page 9:

- The roles of 'set meeting agenda, set meeting roles and select a meeting template' should be clearly stated as roles when the normal user becomes a meeting leader. Stating it clearly will remove ambiguity.

Page 13, section 3.1.1.4, last line:

- The resources that need to be cleaned maybe should probably stated explicitly.

***Design Specification for Client Interface:***

Page 4, line 3:

- The word 'variable' should be replaced by 'function'.
- Under the column 'requirement met', the section number in the parenthesis should be checked. They do not correspond correctly to the document they are supposed to point at.

Page 9:

- There is no line arrow connecting the registration window and the main window after the UID is returned from the server.

No sequence diagrams for logging out, performing search, create W.G., Schedule Meeting and Meeting Logs, from the GUI.

***Design Specification for Meeting Management:***

Page 2, section 1.2, line 2:

- There is no description of each diagram in the document. So, either the description has to be written or this sentence should be rephrased.

Page 7, Page 13:

- The function 'update meeting' should cater for 'meeting roles, change agenda, participants, cancel meeting etc'. In the sequence diagram on Page 13, only a few of the activities such as change leadership, meeting times etc are addressed and shown.

Page 3 – Page 9:

- In the column where a particular requirement met is shown, the section number in the parenthesis should be rechecked to make sure it is actually pointing to the right part of the requirement specification. Some of them seem to be misplaced.

Page 13 – Page 14:

- The diagram is inconsistent as far as displaying the error message is concerned. Where the return value of –1 is there, a label showing what it signifies should be inserted in Page 13 as is done in Page 14.

Page 14:

- Check for access time is shown in the sequence diagram here although, such constraints are not laid out in the requirement specification document.

Page 15:

- No provision to say what happens to the Meeting leader if he leaves the meeting before it completes. Does someone else obtain that role? If a chairman is controlling the meeting, then does the normal user ask for permission to leave? This detail should be highlighted in the sequence diagram. The way it shows now is as if the user leaves whenever he can without any involvement of the chairman.

Page 20:

- When a meeting is removed, how do you notify users that the meeting has been cancelled? There is mention in the requirement analysis that the icon color changes when the status of the meeting is null or void.

Page 21:

- There is no label in the diagram showing what the return value -1 means. This is just a consistency problem, which has to be maintained between the diagrams.

Page 22:

- In the sequence diagram there is no mention of how to add a new WG, to list or to remove old-meeting logs, which have been mentioned as possible functionality while updating a workgroup.

There is no sequence diagram showing how to activate or start a meeting once a user has joined. There is also no mention of the minimum of 10 minutes before which any user cannot start the meeting and also the constraint that the chairman should be present while starting the meeting. Also there is no mention of how to save meeting logs by the chairman.

***Design Specification for Transaction Management:***

There is no sequence diagram for registering a new broker on our ieCollab server. For a user to use the services of ieCollab through an A.S.P., the ASP should first be registered with ieCollab.

Page 20:

- There is no sequence diagram for updating the profile although a slight description is made.

Page 21:

- There is no ASP server shown in the diagram for user registration. The user will trigger the ASP server to send a service request to ieCollab server. This is not highlighted in the sequence diagram.

Page 22:

- Even the logging in sequence diagram does not show the broker.

First of all the sequence diagrams are not consistent with those present in the design specification for meeting management.

There are no sufficient sequence diagrams to explicitly show the programmers all the functionalities of the transaction. Only four sequence diagrams are shown out of which one is on logging in, logging out, registration and account administration. None of them are complete as highlighted above.

The account administration diagram does not show the details of billing information and how to pay through a credit card or some other way as specified in the requirement specification.

None of the diagrams show the functions that need to be made and the arguments they are passing.

More importantly, one does not get the feeling that they are supposed to code an ASP model after reading this document. All the sequence diagrams show the direct use of the ieCollab server through the ieCollab interface rather than through a third party ASP provider.

Through this audit, the QAM was able to uncover some of the basic inconsistencies that were present in the requirement specification documents and the design specification documents. The QAM tried to focus mainly on the technical inconsistencies and faults found in the documents. The QAM did not spend a lot of time looking over the grammatical errors. However, the QAM did note that there were numerous grammatical and spelling mistakes found and requested that both teams responsible for the documents go over them and make corrections. Since the programming phase was well into its third week, the whole motivation for bringing out this report so late in the project was to highlight the mistakes and learn lessons from it.

## 4.5 The Benefits from working as the QAM

One of the benefits obtained from working as the Quality Assurance Managers (QAM) of the ieCollab was learning about the importance of quality in software development projects. The QAM learned that without quality, it would be meaningless to talk about the success of a given project. However, the QAM also learned how difficult, challenging, and time consuming task it was trying to check, enforce, and maintain quality in the project.

Another benefit was the experience of working with a remote partner to achieve quality in the project. As it was mentioned before, the Quality Assurance Team was made up of three students from MIT and one student from CICESE. The QAM learned to be patient, and to be independent and be prepared to handle whatever situations that could result in the distributed collaboration setting. The QAM learned from experience that it was important to work with the remote partner as much as possible in order to utilize the partner's talent. However, when the situation did not permit the collaboration with the remote partner, the rest of the QAM had to be prepared to carry out the entire activities without her.

## 4.6 Conclusion

This chapter discussed the quality assurance aspect of the ieCollab Project. It showed how applying quality assurance to ieCollab was very different from applying quality assurance to other software development projects. It also talked about what the responsibility of the Quality Assurance Managers should be, then, in such an environment. Some of the challenges that the Quality Assurance Managers faced and had to overcome, such as the lack of knowledge and information about performing quality assurance properly and the collaboration in distributed environment for software project development, were listed and discussed.

The Quality Assurance Plan Section showed the plan that was submitted by the Quality Assurance Managers to plan and check quality assurance of ieCollab Project. The plan showed the purpose of the plan, the management of quality assurance, the

relationship and interaction with other teams in the project, the tasks in each and every phase of the project, and the planning and conducting of four analytical quality evaluations (peer reviews, walkthroughs, audits, and inspections) in great details. The section tried to show the significance of the Quality Assurance Plan for this plan was the specific guideline to achieve and maintain highest quality possible in the project.

The following section described some of the actions that the Quality Assurance Managers of ieCollab performed over the course of the project. It discussed the creation of templates for documents, enforcement of making comments to the submitted documents, conducting of walkthroughs, audits, inspection and validation of codes and product. This section tried to show that the in order to achieve high quality in a given project, the quality assurance team has to be involved and working throughout the life of the project in both larger and smaller areas.

The following section of this chapter showed an example of Analytical Quality Assurance Evaluations that was performed in the ieCollab Project. Particularly the audit that was performed on the requirement specifications and the design specifications was described.

The last section of this chapter discussed some of the benefits and lesson that the Quality Assurance Managers learned from the ieCollab Project.

# Chapter Five

## Recommendations for Quality Assurance in Distributed Software Development Collaboration

As it was discussed in Chapter Four, the Quality Assurance Managers (QAM) of Intelligent Electronic Collaboration (ieCollab), faced many challenges and problems as they worked to achieve and maintain the high quality in the project. As the QAM tried to overcome those obstacles, they accomplish quite a bit of the responsibilities given to them, but they also made mistakes and could not be effective in some areas. This chapter will discuss the suggestions and the recommendations, which the QAM came up with from dealing with the challenges and problems, to have more effective quality assurance role in a distributed software development collaboration.

## 5.1 Software Development Knowledge and Training

One of the major challenges that not only affected the QAM but the entire project was having insufficient amount of knowledge and information about the software development process. As it was mentioned before, many students came into the project without any prior experience or knowledge in software development. The only learning

and training that the students received concerning the software development and their roles were the lectures given at the beginning of the project before the start of the actual development phase, usually just one lecture per topic.

This was a very inadequate training for the QAM since most of the students on the quality assurance team had little idea what their role was as they entered into the development phase of the project. This resulted in ineffectiveness, confusion, and frustration on the part of the QAM because they had to carry out responsibilities while they tried to learn about what they are supposed to do.

The recommendation that the QAM makes on this matter would be to have ongoing lectures and training session, even a brief one, over the course of the project to increase and enhance the knowledge required to effectively function as the Quality Assurance Managers. This would be especially more effective if greater part of the training can be given after each team has been formed and everyone knows what they are going to be doing in the project. The students will pay greater attention and have more interests in the lecture and grasp more from it if they feel that it is going to help them do their job more effectively.

Another possible way to accomplish this recommendation would be to have a special training session during the Independent Activities Period (IAP) to fill up the lack of the knowledge and information. This would be especially beneficial for the Master of Engineering students who must carry a lot of class loads during the regular semester. Since it would be hard for them to find the time during the regular semester to go through

an extra training, having it during the IAP would give them more time to get into the training and obtain greater output from it.

## 5.2 Quality Assurance in a Distributed Collaboration

The Quality Assurance Team was made up of three students from MIT and one student from Chile. However, it was extremely difficult for all four students to work together. This was mainly due to one of the team members located in Chile. It was frustrating and patience-testing experience trying to communicate with the team member in a remote location. At the beginning, everyone was excited to work with someone who lived on the other end of the world. But as the project progressed and the team was asked to do real work and produce real documents, the problems started to arise.

Many meetings were held without the participation of the member in Mexico when the chatting system disconnected or it took forever for the email reply to come. The notifications of walkthroughs were sent to the member in Mexico but it was difficult to actually have her contribute to any walkthroughs. After having meetings for a while without the member in Mexico participating, it became natural to meet just amongst MIT students. Everyone on the team realized how difficult it was to collaborate with someone who you can not see when there existed time constraints and due dates. Everyone also realized how easy it was to exclude that member in the remote location and just work amongst us.

The recommendation that the QAM makes on this subject of distributed environment would be to encourage having more frequent contact and interaction

especially with the team members in the remote location to establish a relationship where people depend on their team members who are in a far away location. Whether through email or chatting there needs to be more communication with the team members in geographically dispersed locations. By having more communications, the team members can build the relationship of trust and find out more about the other members' skills and talents that they might be able to use later on in the project. So, even if the chatting service or the video/audio conferencing does not work, the team members would want to continually work with that remote team member for his/her talent and ability.

## 5.3 The Quality Assurance Enforcement Power

One other challenge that the QAM had to face was the dependence on the Project Management Team to enforce many quality assurance issues in the project. It was not clear from the beginning what kind of role the Quality Assurance Team was going to play and how much enforcement power the Quality Assurance Team has. The QAM had to rely on the Project Managers to enforce some of the things that the QAM wanted the entire team to do or follow on.

The QAM tried very hard asking people in the project to make comments to any newly submitted documents. However, the strict enforcement of submitting comments to the new documents was enforced only when the Project Managers (PM) took the situation into their hands and enforced with the PM power. In some instances, the Project Managers did even the announcement of walkthroughs.

The recommendation the QAM gives would be to grant some enforcement power to the hands of the QAM in order for the QAM to work more effectively in the project. The Quality Assurance Team should have an enforcement power in the project independent of the Project Management Team. It would be inefficient, ineffective, and time-consuming if the QAM had to go and borrow power of the Project Managers every time to try to enforce something for the project. In order for the project to be successful and have satisfying level of quality in it, the QAM should be able to check, achieve, maintain, and enforce quality at an even higher level than the Project Managers do.

## 5.4 The Quality Motivation

One of the most important things that a project needs in order to be successful and have high level of quality would be to have a constant level of motivation driving the people to work hard and achieve the best quality they could in the work that they are doing. However, as is the case with many projects, it is hard, if not impossible, to maintain that level of motivation through out the life of the project. It was also evident in ieCollab Project that the people started to lose motivation to work harder and maintain the level of desired quality in their work as the project went on. Dr. Paul Peach discussed psychological aspect of this "Quality Motivation":

> How, then, can we motivate human beings to do what we want them to do, and in particular to work to quality standards? I think we must start by admitting that monetary rewards in themselves will not do the job; we must somehow appeal to the workman's desire for self-approval. In some cultures—the German and Scandinavian, for example—quality workmanship is traditional; it takes little or no inducing to get workman to maintain quality standards, because for any workman to get a reputation

> for careless or inferior workmanship would quickly earn him the disesteem of the other members of his culture. When there is no tradition of quality workmanship, management had a problem. How to induce a workman who, to be blunt, does not care a tuppence whether what he produces is good or bad—how to induce him to produce the kind of quality you want? [Peach, 1975]

What Dr. Peach stated above was a similar situation with ieCollab Project. As people began to face more problems and challenges in ieCollab, their frustration and impatience grew. It was hard to motivate people to continue to work hard; moreover produce a high quality work. At first some of the motivation came from just working in this distributed collaboration. People felt challenged yet motivated to work with their counterparts in Chile and Mexico to achieve success in this project. That was one of their driving motivations. However, as the distributed collaboration did not seem to work out well and people found themselves working more locally than in distributed environment, that motivation died out. The people needed some other motivations to keep working hard but they didn't know where to find that motivations. ieCollab Project needed ways to keep the people motivated throughout the project. Dr. Peach discussed some suggestions that might keep motivating people to maintain quality in their work:

> The essential point is, I think, that we must recognize that if a job bores the worker, he will take no pride in it. Somehow, we must manage to keep his interest alive. .... If the worker can see his contribution to the final product as something important and essential, we can perhaps appeal to his self-respect. We cannot do this by verbal persuasion alone; aside from the message content, any message from "the bosses" to the workers is automatically suspect. It may be as well to recognize the fact that if few workers have quality as a goal, few "bosses" care much about the welfare or happiness of their workers and the workers know it [Peach, 1975].

The recommendation that the QAM would like to make about maintaining the motivation level high in order to maintain a high quality in the project would be to discuss with the professors and come up with ways to keep the motivation level of the team members up throughout the project. Whether by having more lectures and training session to make people feel like they are constantly learning or by other forms of competitive reward system that might get one team to work harder than other teams, it would be extremely important to find some ways to sustain the motivation level that the people have at the beginning stage of the project till the end.

## 5.5 The Future of Quality Assurance

The last recommendation that the Quality Assurance Managers (QAM) would like to make would be concerning the future of the quality assurance. It has become an inescapable fact that the distributed collaboration will be applied more in the future as the developing software product becomes more sophisticated and larger in scale. The way for the QAM to keep maintaining the desirable level of quality in their projects would be to constantly put in their effort in the development and research work to enhance quality assurance measures in various areas. However, most of all, it will be imperative to keep reminding the entire project members the importance of maintaining quality performance and ultimately make them be quality conscious in any type of situations.

The technologies have made the distributed collaboration possible and they will also make better quality checking devices possible. It is no news that many methods of computer assisted quality control have been developed and utilized to check and maintain

quality in a project since long ago. Despite all the progress that has been made, the software development process will always be difficult to plan and to create in the future. This, of course, influences quality assurance. When the software development process can not be planned out clearly and perfectly, people will try to optimize the situation by focusing on the time and the cost factors of the project overlooking and sometimes ignoring or sacrificing the quality factors of the project [Wallmuller, 1994].

It is definitely very important trying to come up with new ways or develop new technologies that might enable better quality checking. However, the QAM must remember that in the future, as today, the abilities of qualified software developers and their quality consciousness will be the central element in successful quality assurance. It would not matter how highly technical ways of checking and maintaining quality are in a project. If the people who are involved in the project do not care about quality, then it would be extremely difficult to achieve quality in that project and its product.

It is recommended that more research work should be done to develop quality assurance tools that can be used in distributed collaborative settings like the distributed inspection process model that was introduced in Chapter Two. The availability of information on performing quality assurance tasks in a distributed collaboration is very scarce. There needs to be more defined and better outlined methodologies describing the effective ways of carrying out the duties of the quality assurance team in a distributed collaboration.

It is also recommended that there should be more research done in the area of ways to train the project team members' minds to be quality-conscious. Better quality

will result if every person involved in the project is constantly thinking, worrying, and looking for ways to improve quality in their own work as well as in the project as a whole. One possible suggestion to train people's minds to be quality-conscious would be to have a weekly seminar on various topics related to quality.

## 5.6 Conclusion

This chapter presented recommendations for quality assurance in distributed software development collaboration. It discussed some recommendation that could be taken into account to effectively check, maintain, and achieve quality in a distributed collaborative project. It listed several areas where the QAM of the ieCollab faced problems and challenges in and ways to overcome them to manage a successful quality assurance.

The first area that it touched upon was the improvement of the knowledge and information level of the students coming into the project with limited experience on distributed collaborative software development. It recommended an ongoing training and teaching to assist the students to better obtain the necessary ability and be more effective in carrying out their responsibilities in their designated assignment.

The second area that it discussed was the challenge to achieve a better success in working in a distributed environment. It suggested that in order for people to work together effectively in a distributed collaborative project, they must have more outside communications through whatever means possible and build the relationship. As a team, they must be able to contribute their part to the team and the team members should expect

that from each other and depend on each other for. By doing so, the people would want to work more with the other person whether he/she is in a local area or a remote area.

The third area that it talked about was recommending to give more independence and enforcement power to the Quality Assurance Managers (QAM) to effectively carry out their duties. The section discussed how the QAM had to depend on the Project Managers most of the time to enforce the quality assurance issues in the ieCollab. The QAM must have enforcement power, in some cases independent even of the Project Managers, to check, achieve, and maintain quality in a project.

The fourth area of the recommendations was in trying to keep the motivation level of the people in the project high enough to sustain the quality level in their work and performance through out the life of the project. It discussed some of the psychological factors that affect the people's motivation level as the project goes on. It also discussed the importance of implementing specific methods, such as special training and competitive rewards, to drive motivational level up during the time of frustration and difficulties. There has to be motivations in order for a high quality to be present.

The last area of recommendations was to constantly trying to improve the ways to check, achieve, and maintain quality in the project through development and research work in the area of quality assurance. But the thing that would be even more important than to improve the technologies of quality checking was to establish a general sense of quality consciousness in every team members' minds.

# Chapter Six

## Conclusion

Assuring the quality of a software development project has become a great responsibility as software development projects have evolved in their sizes and complexity. The quality assurance managers' role to check, achieve, enforce, and maintain quality in a given project has become difficult as more projects found themselves collaborating in a distributed environment. This thesis examined the role of quality assurance in the distributed software development collaboration project called Intelligent Electronic Collaboration (ieCollab). ieCollab Project engaged a total number of 34 students from MIT, CICESE and PUC to experience an actual distributed collaboration to develop a software system that would enable a better collaboration between geographically distributed teams between September 1999 and April 2000.

This thesis tried to show the importance of achieving and maintaining quality in any given software projects. It discussed such topics as the need for quality, the quality assurance in software engineering, the quality factors and criteria, the factor definitions and tradeoffs, analytical quality evaluations such as peer reviews, walkthroughs, audits,

and inspections. It also introduced the Distributed Inspection Process Model to show an example of analytical quality evaluations that can take place in a distributed setting.

It also discussed the ieCollab Project in detail describing the teams, the objective of the project, the process models that the project used to implement, and the challenges that it faced as a whole. The benefits and the lessons learned from the project were also discussed. Everyone in the project had to work hard to overcome those challenges that were described in the thesis, and the learning and the experience from working hard to overcome the difficulties were just as great as the challenges.

The thesis then talked about the role of the Quality Assurance in ieCollab. It discussed how applying quality assurance was different from applying quality assurance to other software projects. It also discussed how that difference brought about problems and challenges on the Quality Assurance Managers' part in trying to check, achieve, and maintain quality in ieCollab. It also explained about the Quality Assurance Plan that was submitted by the Quality Assurance Managers (QAM) and how it became the source of guidelines for the QAM to follow to build, check, evaluate, and enforce different quality measures in ieCollab.

This thesis finally discussed some of the recommendations that the QAM found from their works and experiences in ieCollab. One of the recommendations was to encourage providing a continued learning and training environment to prepare and teach the students to fulfill their roles more effectively and completely. It also made recommendations in areas such as quality assurance in a distributed collaboration, the quality assurance enforcement power, the quality motivation, and in the future of quality

assurance. Especially in the future of quality assurance section, it discussed that even if the project has a well-prepared classroom with all the enabling technologies that would better facilitate distributed collaboration, it still may not increase the success and quality of the distributed collaborative project. The QAM emphasized the importance in bringing up the issue of quality from the beginning of the project till the end, constantly reminding the entire members of the significance of quality, and ultimately establishing the sense of quality consciousness in the minds of the people.

In ending, as it was mentioned in the Software Development and Quality Chapter (Chapter Two), the software industry has been one of the growing markets in the economy. Through globalization of markets, it has become unavoidable for people in remote locations, in a geographically distributed environment to work together in collaboration to strive for a common goal, and achieve and accomplish success. As the scale of software projects and the number of people working without face to face contact increases, the quality assurance has become a crucial part of the software development projects. In this age of increasing expectations, the quality assurance must be prepared to handle situations and questions that might not seem possible at the moment. For without quality, it will be hard to define what a successful software development is.

# References


[1] Cruz, Gregorio, Quality Assurance in Geographically Distributed Software Development, M.Eng Thesis, Civil and Environmental Engineering, MIT, May 1999.

[2] Doherty, B.S., "Software quality through distributed code inspection", Department of Computer Science and Applied Mathematics, Aston University, United Kingdom, 1997.

[3] Evans, M. W., Marciniak, John, Software Quality Assurance and Management, New York: Wiley & Sons, 1986.

[4] IEEE, Software Engineering Standards, IEEE Press, 1997

[5] Jones, Capers, Applied Software Measurement – Assuring Productivity and Quality, New York: McGraw-Hill, Inc., 1991.

[6] Kelly, Mike, Management and Measurement of Software Quality, Vermont: Ashgate Publishing Company, 1993.

[7] McConnell, Steve, Rapid Development, Washington: Microsoft Press, 1996.

[8] NASA, Software Formal Inspections Guidebook, Washington D.C.: Office of Safety and Mission Assurance at NASA, August 1993.

[9] Nesi, Paolo, Objective Software Quality, Italy, Department of Systems and Informatics, University of Florence, 1995.

[10] 1.120 Information Technology M.Eng. Project Syllubus, Civil and Environmental Engineering, MIT, 1999.

[11] Peach, Paul, Bajaria, H. J., Quality Assurance – Methods Management and Motivation, Michigan: Society of Manufacturing Engineers, 1981.

[12] Pressman, R. S., Software Engineering: A Practitioner's Approach. New York: McGraw-Hill, 1997.

[13] Ross, M., Building Quality Into Software, South Hampton: Computational Mechanics Publications, 1994.

[14] Sanders, Joc, Software Quality – A Framework for Success in Software Development and Support, Wokingham: Addison-Wesley Publishing Company, 1994.

[15] Schulmeyer, G. Gordon, McManus, J. I, Hand Book of Software Quality Assurance, New Jersey: Prentice Hall, 1999.

[16] Summers, Chris, Software Quality Assurance, Reliability and Testing, Vermont: Gower Publishing Company, 1987.

[17] Tasso, C., Adey, R.A., Pighin, M., Software Quality Engineering, Massachusetts: Computational Mechanics Inc., 1997.

[18] Vincent, J., Waters, A., Sinclair, J., Software Quality Assurance – Practice and Implementation, New York: Prentice Hall, 1988.

[19] Wallmuller, Ernest, Software Quality Assurance: A practical approach, New Jersey: Prentice Hall, 1994.

[20] Yang, Bob, Managing a Distributed Software Engineering Team, M.Eng Thesis, Electrical Engineering and Computer Science, MIT, May 1998.

# Appendices

# Appendix 1

## COMMENT REPORT

| Date: | | IeCOLLAB | CR Ref: | |
|---|---|---|---|---|
| Subject: | | | SDate: | |

| # | Type | Comments | Action Needed From |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

| WBM | Written Before Meeting | PPI | Previously Postponed Issue | PTN | Postponed Till Next |
|---|---|---|---|---|---|
| URG | Urgent Matter | RMD | Reminder of Issue | SEE | See Attachment |

| Attachment Type: | | | Diskettes | | Prepared By: | |
|---|---|---|---|---|---|---|

# Appendix 2

## QUALITY ASSURANCE WALKTHROUGH ACTION ITEM LIST

Subject of Review: _____
Material prepared by: _____ Date of Review: _____
Moderator: _____ Recorder: _____ Design Leader: _____
Reviewers _____
Current Status _____ New Status _____

| NUMBER | DESCRIPTION OF ACTION ITEM | RESOLUTION |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# Appendix 3

## QUALITY ASSURANCE AUDIT CHECKLIST

Subject of Review: _____
Material prepared by: _____ Date of Review: _____
Moderator: _____ Recorder: _____
Reviewers _____
Current Status _____ New Status _____

| DOCUMENTS | COMMENTS | ACTION |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# Appendix 4

## QUALITY ASSURANCE INSPECTION CHECKLIST

Subject of Review: _____
Material prepared by: _____ Date of Review: _____
Moderator: _____ Recorder: _____
Reviewers _____
Current Status _____ New Status _____

| ITEM | COMMENTS | STATUS |
|---|---|---|
| PLANNING | | |
| OVERVIEW | | |
| PREPARATION | | |
| INSPECTION MEETING | | |
| THIRD HOUR | | |
| REWORK | | |
| FOLLOW-UP | | |

# Appendix 5

## QUALITY ASSURANCE INSPECTION ACTION ITEM LIST

**Subject of Review:** _____

**Date of Review:** _____

**Moderator:** _____

**Recorder:** _____

**Reviewers:** _____

| NUMBER | LOCATION | BRIEF DESCRIPTION | INSPECTOR |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Appendix 6

## QUALITY FACTORS DEFINITIONS in ieCollab

| QUALITY FACTOR | DEFINITION |
|---|---|
| Accuracy | • Extent to which a program satisfies its specifications and fulfills the user's mission objectives.<br>• Does the product do what I expected to do? |
| Reliability | • Extent to which a program can be expected to perform its intended function with required precision.<br>• How does it satisfy the requirements over a period of time? |
| Efficiency | • The amount of computing resources and code requirement by a program to perform a function.<br>• Does the product utilize hardware resources well? |
| Integrity | • Extent to which access to software or data by unauthorized persons can controlled.<br>• Is it safe? |
| Usability | • Effort required learning, operating, preparing input, and interpreting output of a program.<br>• Can I easily learn to handle it? |
| Maintainability | • Effort required locating and fixing an error in an operational program.<br>• Can I correct a fault easily? |
| Testability | • Efforts required testing a program to ensure it perform its intended function?<br>• Can I test the product without additional cost after making changes? |
| Flexibility | • Effort required modifying an operational program.<br>• Can I execute a change easily? |
| Portability | • Effort required transferring a program from a hardware configuration and/or environment to another.<br>• Can I use the product on different hardware? |
| Reusability | • Extend to which a program can be used in other applications related to the packaging and scope of the functions that the program performs.<br>• Can I use parts of the product for other applications? |
| Interoperability | • Effort required to couple one system with another.<br>• Can I create interface to other systems? |

# Appendix 7

## Quality Control in Software Development:

## Programming Standards

**Please return to QA Team Leader Nhi Tan**

Name of Inspector: _____

Date: _____

ieCollab Version: _____

File Name/Identifier: _____

Function/Class Name: _____

Software Version: _____

Creator: _____

Time Spent: _____

| Standard | Applicable | Present | Offending Line |
|---|---|---|---|
| ***FILE HEADER*** | | | |
| File name | | | |
| Description of code | | | |
| Creation date | | | |
| Original author | | | |
| Modification date | | | |
| Modifier | | | |
| Description of modification | | | |
| Hardware requirements necessary for proper execution | | | |
| Software requirements necessary for proper execution | | | |
| Design document reference | | | |

| Standard | Applicable | Present | Offending Line |
|---|---|---|---|
| ***FUNCTION HEADER*** | | | |
| Description of action performed by function | | | |
| Description of parameters passed into function as arguments | | | |
| Description of return value | | | |
| Creation date | | | |
| Original author | | | |
| List of source aiding in design of function (URL, book, person) | | | |
| Modification date | | | |
| Modifier | | | |
| Description of modification | | | |
| Hardware requirements necessary for proper execution | | | |
| Software requirements necessary for proper execution | | | |
| Design document reference | | | |
| ***FILE NAME*** | | | |
| File name is the same as the Class name? | | | |
| ***CLASS LAYOUT*** | | | |
| Class name is the same as in the design document? | | | |
| Public, Protected, Private variables defined first in this order? | | | |
| Class broken into the following order:<br>• Constructors<br>• Operators<br>• Methods to access object variables<br>• Methods to change object variables<br>• Parent class method overrides<br>• Other functions | | | |

| Standard | Applicable | Present | Offending Line |
|---|---|---|---|
| Subsections separated by a blank line? | | | |
| Subsections separated by a single line comment? | | | |
| ***VARIABLE DECLARATION COMMENTS*** | | | |
| Description of variable role if not apparent from name? | | | |
| USE OF I, J, K | | | |
| i, j, k used only in Do While & For loops? | | | |
| ***STATEMENT GUIDELINES*** | | | |
| Function arguments separated by space? | | | |
| Infinite loops coded using an empty for statement (for(;;)) ? | | | |
| One variable declaration per line? | | | |
| Additional declarations of same type one per line and indented below the first declaration? | | | |
| "do" statement in a do-while loop on a separate line? | | | |
| "while" statement on the same line as the closing brace? | | | |
| "do" portion of function statement on a separate line as that of the opening brace? | | | |
| Opening and closing braces in the same column? | | | |
| ***NAMING CONVENTIONS*** | | | |
| Modifiers for variables follow guidelines given in the table below? | | | |

# Appendix 8

## COMMENT REPORT

| Date: | 4/1/2000 | IeCOLLAB | CR Ref: | |
|---|---|---|---|---|
| Subject: | Quality Control: Programming Standards | | SDate: | |

| # | Type | Comments | Action Needed From |
|---|---|---|---|
| 1 | | Please add creater/modifier name to file. | Engine.java, ProfileWindow.java |
| 2 | | Please add creation date to file. | DBTool.java |
| 3 | | Please add comments describing input arguments, return values and function purpose. | Engine.java, ProfileWindow.java, CollabUser.java |
| 4 | | Please add comments describing class purpose at the top of the file. | CollabUser.java, ProfileWindow.java, Engine.java |
| 5 | | Please line open and close parenthesis by column. | CollabUser.java, DBTool.java |
| 6 | | Please add hardware, software requirements necessary for proper execution. | CollabUser.java, ProfileWindow.java, Engine.java, DBTool.java |
| 7 | | Please add design document reference. | CollabUser.java, ProfileWindow.java, Engine.java, DBTool.java |
| | | | |
| | | | |

| WBM | Written Before Meeting | PPI | Previously Postponed Issue | PTN | Postponed Till Next |
|---|---|---|---|---|---|
| URG | Urgent Matter | RMD | Reminder of Issue | SEE | See Attachment |

| Attachment Type: | Documents | | Diskettes | | Prepared By: | Nhi Tan |
|---|---|---|---|---|---|---|

# QUALITY ASSURANCE WALKTHROUGH ACTION ITEM LIST

Subject of Review: Requirement Specification Version 1.4
Material prepared by: Requirement Analysts Date of Review: 1/18/00
Moderator: Nhi Recorder: Saeyoon Design Leader: Polo
Reviewers Representatives from each team
Current Status _____ New Status _____

| NUMBER | DESCRIPTION OF ACTION ITEM | RESOLUTION |
|---|---|---|
| 1 | Put the dates in the References and Links part in the first page. | Corrected |
| 2 | Include the page number in the Outline | Corrected |
| 3 | Fix the bi-directional. | Corrected |
| 4 | Each figure should have a caption. | Corrected |
| 5 | Clarify the definition of user. | Corrected |
| 6 | Proper referencing. | Corrected |
| 7 | Reference to Documents. | Corrected |
| 8 | Anyday.com - bibliography including day and year. | Corrected |
| 9 | Use unique identifier - 'user & broker' or 'user' only. | Corrected |
| 10 | Work on version 1. | Corrected |
| 11 | Include use-case scenarios of how the different types of transactions are occuring. | Corrected |
| 12 | Number the paragraphs. | Corrected |
| | | |
| Open Issue | Whether or not to allow the Yahoo.com user to login to Anyday.com with the same username and password. | |

# QUALITY ASSURANCE AUDIT CHECKLIST

Subject of Review: Requirement Specification and Design Specification
Material prepared by: R.A. and Designers Date of Review: 3/10/00
Moderator: Kaissar Recorder: Eswar
Reviewers Quality Assurance Team and Project Managers
Current Status _____ New Status _____

| DOCUMENTS | COMMENTS | ACTION |
|---|---|---|
| Requirement Specification Meeting Management | Grammatical Errors.<br>Inconsistency in roles specified in the sequence diagram. | Correct errors and clarify ambiguity. |
| Requirement Specification Transaction Management | | |
| Design Specification for Meeting Management | Add description of each diagram.<br>Recheck the section number for clarity. | More description and clarity in the document. |
| Design Specification for Transaction Management | Add sequence diagram.<br>No ASP server.<br>Inconsistency in diagrams. | Work on diagrams and inconsistency. |
| Design Specification for ieCollab Client Interface | Misuse of terms.<br>Ambiguity in diagrams. | Check the documents and eliminate ambiguity. |
| | | |
| | | |

# Validation of Client Interface
# For ieCollab Version 2
# Client Interface

## Quality Assurance Team

---

**Created by: Nhi Tan**
**Date: February 15, 2000**
**Participants on Modification**

- offline Session: Nhi Tan

---

- **References and Links**
  Requirement Specifications for ieCollab V2, Meeting Management Spec. 1.4 February 17, 2000
- Requirement Specifications for ieCollab V1, Transaction Management Spec. 1.3 February 17, 2000
- Requirement Specifications for ieCollab V1, Meeting Management Spec. 1.3 February 17, 2000

---

## Outline



---

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to validate and verify the requirements of ieCollab as outlined in the Requirement Analysis documents are met in the product.

### 1.2 Scope

This document presents a test of ieCollab with respect to the original requirements. This document tests the Client Interface for requirements outlined in Requirement

Specification for ieCollab V2, Meeting Management. As the product moves closer towards completion, subsequent validation documents of Transaction Management and Meeting Management for the Requirements Specification will be created as those layers are completed. In addition, validation of Transaction Management, Meeting Management, and Client Interface for Design Specifications will be presented in separate documents.

## 2. Validation

The following is a validation of the GUI with regard to the Requirements Specification. The logical format follows the flow of the document in Requirements Specification.

### 2.1 GUI Description

The client interface is somewhat different from the requirements in the Requirement Analysis document (Section 3 and Appendix A). However, the spirit of the requirements, such as hidden options behind buttons and mouse clicks, is kept.

### 2.2 Buttons

#### 2.2.1 Edit user profile button

Follows requirements.

#### 2.2.2 Schedule Meeting button

- A lot of the requirements are under the "edit Meeting" button. Most requirements are met either under this button or under edit Meeting.
- No seach tool.
- In edit Meeting, the lists are invited members. There is no list of members who have accepted the invitation.
- No date and time field.

#### 2.2.3 Create Workgroup button

- A lot of the requirements are under the "edit Workgroup" button. Most requirements are met either under this button or under edit Workgroup.
- No seach tool.
- In edit Workgroup, the lists are invited members. There is no list of members who have accepted the invitation.
- Unable to set Workgroup policy functionality.

#### 2.2.4 Logoff button

Unable to test if logoff cleans resources (for both force and accidental logoff).

#### 2.2.5 Search button

Does not function

### 2.3 Lists

#### 2.3.2 The Workgroup List
- Workgroups are not differentiated by appearance (ie, color, type format)
- Not dynamic (no pop-up information window when a Workgroup is selected)

#### 2.3.3 The Meeting List
- Meetings are not differentiated by appearance (ie, color, type format)
- Not dynamic (no pop-up information window when a Meeting is selected)

#### 2.3.4 The Old Meeting Log List
- Logs are not differentiated by appearance (ie, color, type format)
- Not dynamic (no pop-up information window when a Log is selected)

#### 2.3.5 The Invitations List
- Invitations are not differentiated by appearance (ie, color, type format)
- Not dynamic (no pop-up information window when an Invitation is selected)
- No Meeting cancellation message.

## 3. Conclusion
The product is a work in progress and this document only serves as a gentle reminder to follows the specifications when finishing the product.

---

# Validation
# Of
# Requirement Analysis & Design Documents

## Version 1.0

---

**Version 1.0 by Eswar Vemulapalli**

Date: March, 2000
Participants on Modification:

- offline Sessions: Eswar Vemulapalli

---

**References and Links**
(All references are stored at http://collaborate.mit.edu/1.120.html)

- Design Specification for ieCollab Client Interface February 22, 2000
- Requirement Specification Meeting Management (version.1.4) February 17, 2000
- Requirement Specification Transaction Management (version 1.6) February 17, 2000
- Design Specification for Meeting Management (version 0.2) February 22, 2000
- Design Specification for Transaction Management (version 1.0) March 1, 2000

---

## Outline



---

## 1. Introduction

This document presents the inconsistencies present between the requirement specification documents and the design documents. The documents reviewed are the two requirement specification reports and the two design specification reports for meeting management and transaction management respectively.

The purpose of this document is to present in a formal layout the differences between the requirement specifications and the design specifications. It also highlights the lack of detail in some aspects of the design specifications.

## 2. Requirement Specification Document for Meeting Management

- On P/5, second bullet under the roles of normal workgroup member.
  - It should be 'list members of workgroup and/or users' to be consistent with the sequence diagram shown on p/8.

- On p/8,
  - Add another label in the sequence diagram for Relinquishing work group membership to be consistent with the roles specified earlier.

- On p/9
  - The roles of 'set meeting agenda, set meeting roles and select a meeting template' should be clearly stated as roles when the normal user becomes a meeting leader. Stating it clearly will remove ambiguity.

- On p/13, section 3.1.1.4, last line.
  - The resources that need to be cleaned maybe should probably stated explicitly.

## 3. Design Specification for Client Interface

- On p/4, line 3
  - The word 'variable' should be replaced by 'function'.
  - Under the column 'requirement met', the section number in the parenthesis should be checked. They do not correspond correctly to the document they are supposed to point at.

- On p/9
  - There is no line arrow connecting the registration window and the main window after the UID is returned from the server

  - No sequence diagrams for logging out, performing search, create W.G., Schedule Meeting and Meeting Logs, from the GUI.

## 4. Design Specification for Meeting Management

- P/2, section 1.2, line 2
  - There is no description of each diagram in the document. So either the description has to be written or this sentence should be rephrased.

- P/7, p/13
  - The function 'update meeting' should cater for 'meeting roles, change agenda, participants, cancel meeting etc'. In the sequence diagram on p/13, only a few of the activities such as change leadership, meeting times etc are addressed and shown.

- P/3 – p9
  - In the column where a particular requirement met is shown, the section number in the parenthesis should be rechecked to make sure it is actually pointing to the right part of the requirement specification. Some of them seem to be misplaced.

- P/13 – p/14
  - The diagram is inconsistent as far as displaying the error message is concerned. Where the return value of –1 is there, a label showing what it signifies should be inserted in p/13 as is done in p/14.

- P/14
  - Check for access time is shown in the sequence diagram here although, such constraints are not laid out in the requirement specification document.

- P/15
  - No provision to say what happens to the Meeting leader if he leaves the meeting before it completes. Does someone else obtain that role? If a chairman is controlling the meeting, then does the normal user ask for permission to leave. This detail should be highlighted in the sequence diagram. The way it shows now is as if, the user leaves whenever he can without any involvement of the chairman etc.

- P/20
  - When a meeting is removed, how do you notify users that the meeting has been cancelled? There is mention in the requirement analysis that the icon color changes when the status of the meeting is null or void.

- P/21
  - There is no label in the diagram showing what the return value –1 means. This is just a consistency problem, which has to be maintained between the diagrams.

- P/22
  - In the sequence diagram there is no mention of how to add a new WG, list or remove old meeting logs which have been mentioned as possible functionality while updating a workgroup.

  - There is not sequence diagram showing how to activate or start a meeting once a user has joined. There is also no mention of the minimum of 10 minutes before which any user cannot start the meeting and also the constraint that the chairman should be present while starting the meeting.

  - Also there is no mention of how to save meeting logs by the chairman.

## 5. Design Specification for Transaction Management

- There is no sequence diagram for registering a new broker on our ieCollab server. For a user to use the services of ieCollab through an A.S.P., the ASP should first be registered with ieCollab.

- P/20
  - There is no sequence diagram for updating the profile although a slight description is made.

- P/21
  - There is no ASP server shown in the diagram for user registration. The user will trigger the ASP server to send a service request to ieCollab server. This is not highlighted in the sequence diagram.

- P/22
  - Even the logging in sequence diagram does not show the broker.

- First of all the sequence diagrams are not consistent with those present in the design specification for meeting management.

- There are no sufficient sequence diagrams to explicitly show the programmers all the functionalities of the transaction. Only four sequence diagrams are shown out of which one is on logging in , logging out, registration and account administration. None of them are complete as highlighted above.

- The account administration diagram does not show the details of billing information and how to pay through a credit card or some other way as specified in the requirement specification.

- None of the diagrams show the functions that need to be made, the arguments they are passing etc.

- More importantly, one does not get the feeling that they are supposed to code an ASP model after reading this document. All the sequence diagrams show direct use of the ieCollab server through the ieCollab interface rather than through a third party ASP provider.

## 6. Conclusion

This report covers some of the basic inconsistencies present in the documents. It should be noted that only technical shortcomings were addressed. It was seen too trivial to address grammar mistakes although there are aplenty. Since the programming phase is well into it's third week, the whole motivation for bringing out this report so late in the project is to highlight the mistakes and learn lessons from it. Also it was kept in mind that continued development of the product in the future will take place and this report will hopefully go a long way in helping rectify the mistakes and inconsistencies put forward.