

Quality Assurance in Geographically Distributed Software Development

by

Gregorio Cruz

B.S., Civil and Environmental Engineering (1998)

Massachusetts Institute of Technology

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 1999

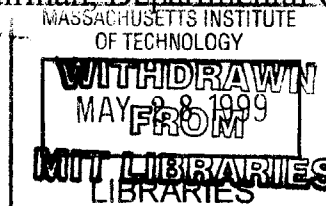
June, 1999

© 1999 Massachusetts Institute of Technology
All rights reserved

Author.....
Department of Civil and Environmental Engineering
May 13, 1999

Certified by.....
Fenioksy Peña-Mora
Assistant Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by.....
Andrew J. Whittle
Chairman, Departmental Committee in Graduate Studies



ENC

Quality Assurance in Geographically Distributed Software Development

by

Gregorio Cruz

Submitted to the Department of Civil and Environmental Engineering
on May 13, 1999,
in partial fulfillment of the requirements for the degree of
Master of Engineering in Civil and Environmental Engineering

ABSTRACT

This thesis examines quality assurance standards and practices for the development of software systems in a geographically distributed environment. The knowledge acquired came from managing the quality process of DiSEL (Distributed Software Engineering Laboratory), a distributed software project conducted between the Massachusetts Institute of Technology (MIT) and Centro de Investigacion Cientifica y de Educacion Superior de Ensenada (CICESE), a university of higher learning in Mexico.

This thesis will also identify the problems that distributed software engineering teams face when collaborating on a project. It will be shown that as a software project becomes distributed, the need to verify the quality of the software process increases. A special focus will also be given to the problems that affected the performance of the Quality Assurance Engineer (QAE) in such environment. This study found that the team must be kept informed of all the events surrounding quality assurance and one way to do this is by creating a repository, such as a web site, to store all quality assurance work. Thus, every member will have access to the QAE's work throughout the software development process, potentially increasing the performance of the whole team.

Thesis Supervisor: Feniosky Pena-Mora
Title: Assistant Professor

Acknowledgements

First of all, I would like to thank God. Lord, without you, nothing is possible. Thank you for the strength that you have given me to finish my school career here at MIT. To my parents, I would never be able to repay you for all the love, support, and guidance that you have giving me. Dad, you are the best and I thank you for giving me the opportunity to study in the United States. Madre, tu eres la fuerza que alimenta mi alma y la razon por la que he bucado superarme para poder darles una vida mejor, te quiero mucho mama. To my brother Edgar whom I would never finish thanking for taking care or mom and dad. A mis hermanas, lupita y toña, las quiero mucho. To my girlfriend, I love you.

To my advisor, Fenioksy Peña-Mora, thank you for all the revisions made to this thesis and specially for pushing me to stay in school and get my master degree. Thank you to the CDI team from whom I learned a lot about software engineering and also how to sell and idea.

Thank you to Carlos Labrada, who became more than a friend. To Ramon Rodriguez, Ivan Gonzale-Gallo, Antonio (Chamo) Fuentes, Juan Carlos Deniz, Emmanuela Binello, Barabara Jimenez. Thank you all for you friendship, I would never forget you.

Table of Contents

TABLE OF CONTENTS	4
LIST OF FIGURES.....	7
LIST OF TABLES	7
INTRODUCTION.....	9
OVERVIEW	10
SOFTWARE ENGINEERING.....	11
<i>Life Cycle Phases</i>	12
MANAGEMENT	15
FACTORS AND CRITERIA	19
TECHNICAL REVIEWS.....	22
CONCLUSION.....	25
THE DIESEL PROJECT	27
THE TEAM.....	28
OBJECTIVE	32
ENVIRONMENT	33
<i>Communications</i>	35
PROCESS MODEL	36
<i>Application</i>	37
CONCLUSION.....	37
APPLYING QUALITY ASSURANCE TO DIESEL	38
ROLE INTEGRATION	39

QUALITY ASSURANCE PLAN.....	40
<i>Management</i>	43
<i>Schedule</i>	46
<i>Software Documentation</i>	47
<i>Technical Reviews</i>	50
Peer Reviews.....	50
Audits.....	51
Walkthroughs.....	51
Inspections.....	56
<i>Example of Technical Reviews Performed in DiSEL</i>	63
Audit of Design Document v2.0.....	64
Walkthrough on Project Manager’s Plan.....	65
CONCLUSION.....	67
RECOMMENDATIONS	69
CLASS STRUCTURE FOR DISTRIBUTED SOFTWARE PROJECTS.....	69
APPLICATION OF QUALITY ASSURANCE TO DISTRIBUTED PROJECTS.....	72
CONCLUSION.....	81
CONCLUSION.....	82
BIBLIOGRAPHY	85
APPENDIX 1: AUDIT NOTIFICATION	87
APPENDIX 2: PREPARATION LOG	88
APPENDIX 3: WALKTHROUGH ANNOUNCEMENT.....	89
APPENDIX 4: ACTION ITEM LIST	90
APPENDIX 5: INSPECTION NOTIFICATION SHEET	91

APPENDIX 6: INSPECTION CHECKLIST92

APPENDIX 7: AUDIT NOTIFICATION TO REVISE DESIGN DOC. V2.0.....93

APPENDIX 8: PREPARATION LOG TO REVISE DESIGN DOC. V2.094

APPENDIX 9: WALKTHROUGH ANNOUNCEMENT OF PM PLAN95

List of Figures

FIGURE 1. SOFTWARE ENGINEERING DEVELOPMENT LIFE CYCLE PHASES.....	13
FIGURE 2. QUALITY ASSURANCE ORGANIZATIONAL DIAGRAM	16
FIGURE 3. RELATIVE COST OF ERROR CORRECTION [WALLMULLER, 1994].....	17
FIGURE 4. SOFTWARE WITH KNOWN AND UNKNOWN ERRORS AND FAULTS	18
FIGURE 5. PROJECT ORGANIZATION.....	32
FIGURE 6. CLASSROOM ENVIRONMENT	35
FIGURE 7. WATERFALL MODEL	36
FIGURE 8. SPIRAL MODEL	36
FIGURE 9. QUALITY ASSURANCE ORGANIZATIONAL DIAGRAM	44
FIGURE 10. AUDIT NOTIFICATION SHEET	75
FIGURE 11. WALKTHROUGH ANNOUNCEMENT SHEET.....	76
FIGURE 12. INSPECTION NOTIFICATION SHEET.....	77
FIGURE 13. REVIEWER'S PREPARATION LOG	78
FIGURE 14. DOCUMENTATION SPECIALIST ACTION ITEM SHEET	79
FIGURE 15. MODERATOR'S CHECKLIST FOR INSPECTIONS	80

List of Tables

TABLE 1. FACTORS OF SOFTWARE QUALITY.....	20
TABLE 2. THE IMPACT OF NOT MEASURING OR NOT SPECIFYING SOFTWARE QUALITY FACTORS.....	20
TABLE 3. CRITERIA OF SOFTWARE QUALITY FACTORS	22
TABLE 4. TASKS OF THE QAE DURING THE SOFTWARE DEVELOPMENT PROCESS	42
TABLE 5. INTERACTION BY QAE AND PRIMARY ROLES	44
TABLE 6. INTERACTION BY QAE AND SECONDARY ROLES	46
TABLE 7. QUALITY ASSURANCE SCHEDULE	47
TABLE 8. OUTLINE FOR A NEW DISTRIBUTED COURSE.....	71

Chapter One

1 Introduction

This thesis examines quality assurance standards and practices for the development of software systems in a geographically distributed environment. The knowledge acquired came from managing the quality processes of the Distributed Software Engineering Laboratory (DiSEL), a distributed software project conducted between the Massachusetts Institute of Technology (MIT) and Centro de Investigacion Cientifica y de Educacion Superior de Ensenada (CICESE), a university of higher learning in Mexico.

This thesis will also identify the problems that distributed software engineering teams face when collaborating on a project. It will be shown that as a software project becomes distributed, the need to verify the quality of the software process increases. A special focus will be given to the problems that affected the performance of the Quality Assurance Engineer (QAE) in such environment. This study found that the team must be kept informed of all the events surrounding quality assurance and one way to do this is by creating a repository, such as a web site, to store all quality assurance work. Thus, every member will have access to the QAE's work throughout the software development process, potentially increasing the performance of the whole team.

1.1 Overview

“We live today in a society in which the computer plays an ever increasing role [and] one of the major elements contributing to the success of the computer is software” [Wallmuller, 1994]. With the explosion of the Internet, software projects that used to be local have become distributed. Having distributed teams working over the Internet has allowed the life cycle for the development of “systems and software to drop, [making] the evaluation of quality difficult” [Evans, 1986]. Quality is a factor that determines a software project’s success so the implications are significant.

In addition to quality, there are two other factors that contribute to the success of a project. These two factors are time and cost. Wallmuller, in his book, states findings by Nenz on the problems regarding the three factors mentioned above [Nenz, 1983 and Wallmuller, 1994]. The problems are as follow:

Time factors

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

Cost factors

- Development cost increases exponentially with the complexity of software. The high 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 per cent 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 per cent 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.

Out of the three factors described above, product quality seems to be sacrificed more often by project managers because of their pressure to finish a project on time and within budget. Little attention is given to software or system quality, even though it has been proven that quality is essential for the advancement in the development of software products, or for the integration of many systems.

In order to produce quality work within any distributed Software Engineering project, quality assurance management must be throughout the development of the software product. A Quality Assurance Engineer (QAE) supervises the project and makes sure that good quality work throughout. From the Systems Engineering Phase to the Operation and Maintenance Phase, the QAE is always making sure that the team is following the standards of software engineering and that they are producing a software product as stated in the software requirements document.

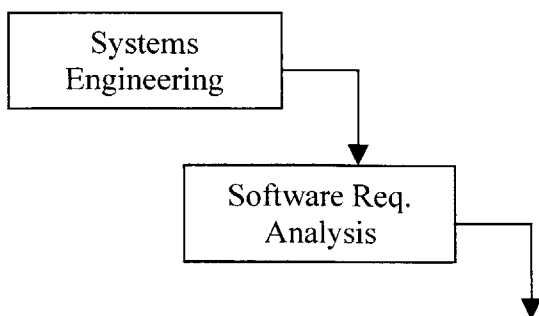
1.2 Software Engineering

Software engineering has enabled software systems to be developed more easily by breaking the software development process into stages. At every stage of the software

development life cycle, the QAE should be able to apply standards of software engineering. Because of the high complexity and low quality management on software engineering projects, software systems were considered hard to make. However, thanks to the software systems that have motivated software engineers, forced an articulation of project goals and engineering's responsibilities, as well as helped to smooth the flow of work, and data, software engineer have been able to integrate large software systems and make higher quality products [Evans, 1986].

1.2.1 Life Cycle Phases

The software development life cycle is composed of several phases (Systems Engineering, Software Requirement Analysis, Design, Implementation, Testing, and Operation and Maintenance), each of which ends with a milestone event and each of which produces an identifiable combination of documents or software items. These cycles are the foundation for good software engineering projects. Below, Figure 1 shows the most commonly used cycles executed in a waterfall model, which is the most common software process used in software engineering.



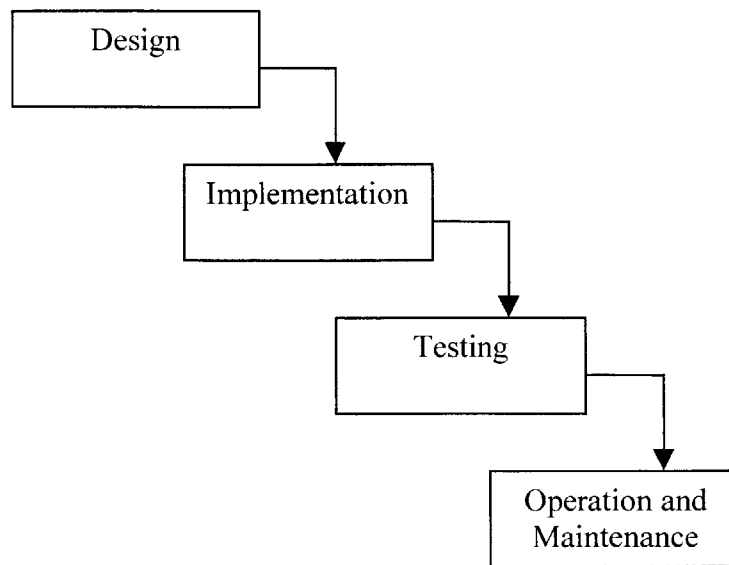


Figure 1. SOFTWARE ENGINEERING DEVELOPMENT LIFE CYCLE PHASES

Systems Engineering - During this phase, the systems on which the actual software project is going to be developed are tested to see that it complies with the standards that are going to be used in the development of the new system. The most important part of this phase is to check how well the hardware will be working with the software system already in place. Not to mention how well the new system will be integrated with the old system.

Software Requirement Analysis - At this stage of the software life cycle, the Requirements Analysts interview the client to gather the actual requirements that will serve as the foundation for the project. All of the requirements taken from the client are put into the software requirements document. At the end of this stage, the software requirements document is reviewed by the QAE. If the document meets the agreed upon

software standards, is written clearly and in an understandable way, then the document is frozen and used as the basis for the design document.

Software Design – During this cycle, the Designer takes the software requirements document that was frozen on the previous cycle and starts designing the architecture of the system that is going to be developed. The Designer serves as the bridge between the Requirement Analyst and the Programmers. So it is very important that the Designer creates a good design document that can easily be implemented by the Programmers. The Designer must also make sure that they are designing what has already been stated in the requirements document. A good design makes testing of quality and maintenance of software easier [Wallmuller, 1994]. Thus, it is essential that the QAE verify every phase during the design cycle through technical reviews such as Audits (See Section 1.5). At the end of the design cycle, the QAE will conduct a more formal technical review such as a Walkthrough or Inspection (see Section 1.5). After the document has been review, it is frozen and used as a baseline for the following cycle.

Implementation – During this cycle, the Programmers transform the already frozen design document into code and the actual software systems. Based on the quality of the design document, the programmers will have an easy or difficult implementation. To verify that good programming is being done, peer reviews (see Section 1.5.1) are conducted to assure the quality of the code.

Testing - This is the stage where the Test Engineers check the system to see that it complies with what was stated in the requirements phase. Based on the quality of the design document, the Test Engineers should be able to formulate test cases.

Operation and Maintenance - This is the final stage of the software engineering development life cycle. At this stage a manual is developed. The manual will serve as a user's guide for the system interface. The maintenance phase serves as the "introductory training of the user" [Wallmuller, 1994].

1.3 Management

To manage the quality of Distributed Software Engineering Projects, the Quality Assurance Engineer (QAE) must develop a Quality Assurance Plan (QAP). The QAP will serve as the central aid for planning and checking quality assurance. This plan will contain of the quality measures for a software project, and consequently is the ultimate quality control reference document of the project.

To aid with the management of quality in distributed Software engineering projects, the QAE needs to work closely with all the members of the team. In particular, the QAE must work closely with four roles from the software engineering development group. These roles can be seen below on Figure 2.



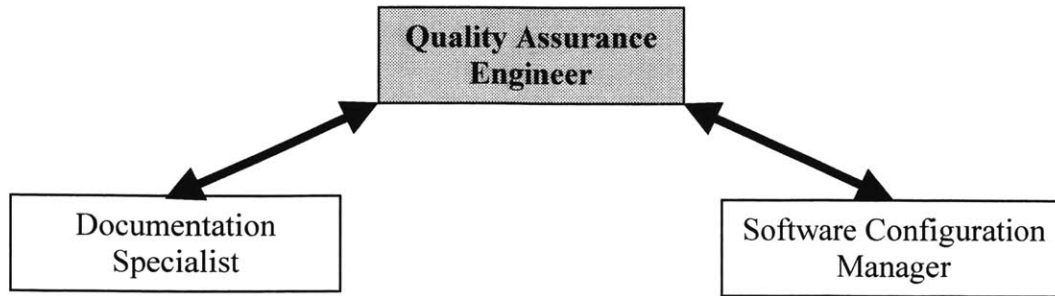


Figure 2. *QUALITY ASSURANCE ORGANIZATIONAL DIAGRAM*

The Validation & Verification Engineer, the Documentation Specialist, the Test Engineer, and the Software Configuration Manager more roles primarily responsible for assuring that the software development cycles are completed. These four roles are the personnel who also aid the QAE in making sure that the final product is of the highest quality. The Validation & Verification Engineer makes sure that the client’s requirements are stated in the requirements document and are followed throughout the development of the system. The Documentation specialist keeps tracks of all the data that is produced and he/she makes sure that everything is put into a repository. The Test Engineers test the code from the programming to the user interface stages. The Software Configuration Manager keeps track of all the versions of documents and code. It is essential that the Configuration Manager makes a plan to show how the code will be locked and released. This will make sure that the programmers are working on the latest version of the code.

The remaining engineering involved in the project also interact with the QAE but not in the same manner as the primary roles. The QAE, with the help of the primary roles, enforce the project’s process. To this end, the QAE schedules several technical reviews

to verify the work that is produced by the secondary roles. This enables the QAE to track quality assurance through the project.

As the QAE starts managing the project, he/she must pay close attention to the early phases of the life cycle. The QAE should try to correct all of the errors found at the beginning of the project, as they are the expensive as errors found at later stages. Correcting errors in the later phases can cost 100 times more than at earlier stages of the development. Figure 3 below shows the cost of fixing errors at the different cycles in a project [Wallmuller, 1994].

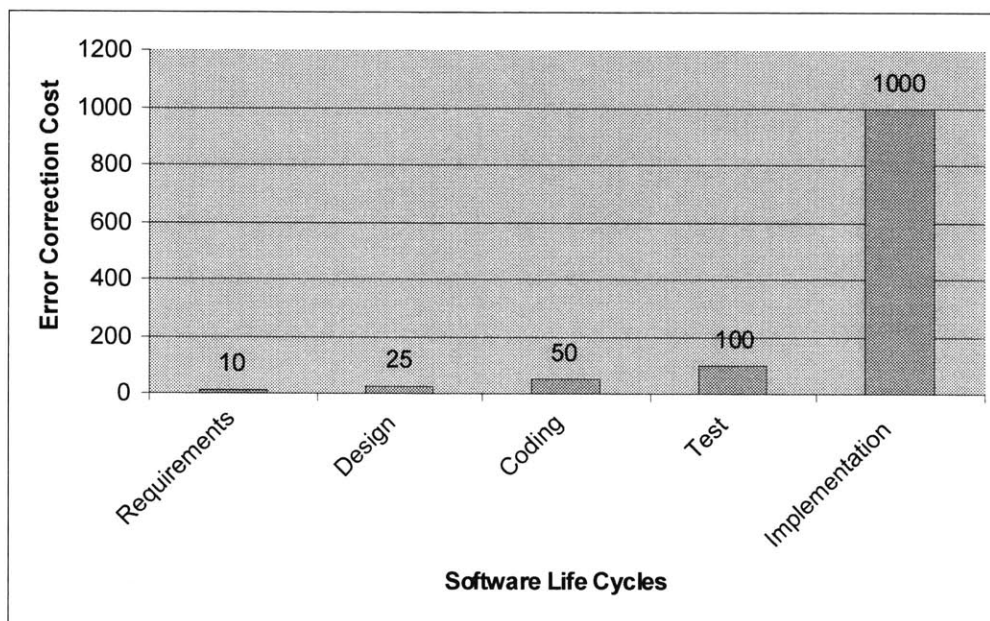


Figure 3. RELATIVE COST OF ERROR CORRECTION [WALLMULLER, 1994]

For distributed teams, the time spent on the early phases of the project is important and all the errors found should be fixed immediately. It will be expensive in time and money for the team if they start addressing errors in the subsequent cycles. Errors that do not get fix at the beginning will multiply and will be added to the errors

that are found in the later stages. As stated in Figure 4, the estimated cost to fix an error in the requirements cycle is relative small but if they are not addressed and fixed, they have the potential to become more problematic bugs in the system.

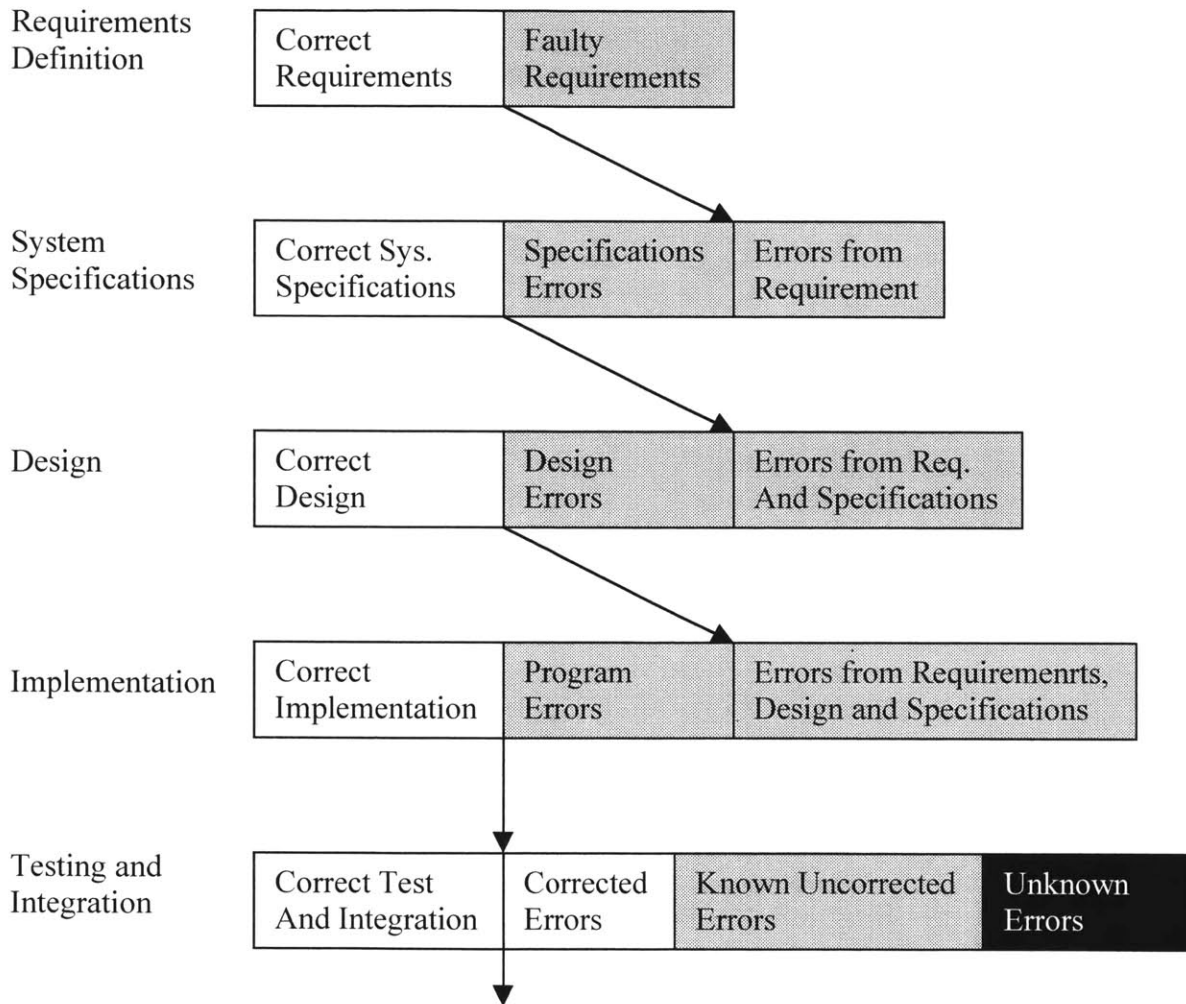


Figure 4. SOFTWARE WITH KNOWN AND UNKNOWN ERRORS AND FAULTS

According to Mizumo [Mizumo, 1983] and Wallmuller the creation of errors and faults is due to an accumulation effect [Wallmuller, 1994]. Errors that are created in the early stages are not corrected and consequently, they are passed on to the next cycles. This is another reason why quality management needs to review every cycle in the

software development life cycle. Figure 4 is taken from Wallmuller's book and it shows how errors will be carry to the next stages and will become very costly and not fixable [Wallmuller, 1994].

1.4 Factors and Criteria

Factors are a way to evaluate the quality of a software product. Some of the questions ask to determine factors are; Can the product be reproduced? Is the product well documented? Can the product be used for other applications? Did the product satisfy the requirements stated at the beginning of the project? Can other systems be built on top of the system that is being created? These questions can help discern whether a software product is of good quality. These factors are used as measurements to aid the reviewers during technical reviews.

The most defined factors for stating the quality of a software projects are from a study conducted by J. A. McCall, P.K. Richards, and G.F. Walters for the Rome Air Development Center (RADC) [McCall, 1977]. Table 1 below shows the most commonly know factors and a brief definition as stated by Vincent (It should be noted that Vincent's work was based on McCalls, Richards, and Walter's work) [Vincent, 1988].

Furthermore, Table 2 below shows how the different factors are implemented into the different software engineering life cycles and the impact they have if they are not measured correctly. Table 2 is also drawn from Vincent book (page 24-25) where he makes some modifications to McCall, Walters, and Richard's original study.



Table 1. FACTORS OF SOFTWARE QUALITY

QUALITY FACTOR	DEFINITION
Correctness	Extend to which a program satisfies its specifications and full fills the user's mission objectives.
Reliability	Extend to which a program can be expected to perform its intended function with required precision.
Efficiency	The amount of computing resources and code requirements by a program to perform a function.
Integrity	Extend to which access to software or data by unauthorized persons can be controlled.
Usability	Effort required to learn, operate, prepare input, and interpret output of a program.
Maintainability	Effort required to locate and fix an error in an operational program.
Testability	Effort required to test a program to ensure it performs its intended function.
Flexibility	Effort required to modify an operational program.
Portability	Effort required to transfer a program from one hardware configuration and/or environment to another.
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.
Interoperability	Effort required to couple the system with another.

Instead of having three areas of life cycle phases, Development, Evaluation, and Operation, Vincent recast these phases with the most commonly known software cycle phases, which are Study or Analysis, Design, Development, and Operation. This is very applicable to the system that was developed in the Distributed Software Engineering Laboratory (DiSEL) project, which will be described in more detail on Chapter 2. DiSEL starts with the Requirements Analysis, and then it goes through the design followed by the implementation and testing phase. This takes care of the Study, Design and Development cycles of the Software Engineering life cycles. The last cycle, Operation, was not implemented because it was beyond the scope of the class.

Table 2. THE IMPACT OF NOT MEASURING OR NOT SPECIFYING SOFTWARE QUALITY FACTORS

SOFTWARE ENG LIFE CYCLE PHASES	STUDY	DESIGN	DEVELOPMENT		OPERATION		
Phases	Requirements Analysis	Design	Code and Debug	System Testing	Transition	Operation	Maintenance
Correctness	△	△	△	□		□	□
Reliability	△	△	△	□		□	□
Efficiency	△	△	△			□	
Integrity	△	△	△			□	
Usability	△	△		□			□
Maintainability		△	△		□		□
Testability		△	△	□	□		□
Flexibility		△	△		□		□
Portability		△	△		□		
Reusability		△	△		□		
Interoperability		△	△	□	□		

Legend:  = where quality Factors should be measured  = where impact of poor quality will be realized

Vincent’s second alteration is made to the placement of the “Transition stage of the Operation phase-suggesting that the impacts listed would affect the user in making a transition to a new software system” [Vincent, 1988]. Vincent talks about the impacts felt when making a transition from an “old software system to the product being developed, [thus moving] the transition phase to the beginning of the operation phase” [Vincent, 1988].

Along with the factor mentioned above, Criteria were created (see Table 3) which are those characteristics that define the quality Factors [Vincent, 1988]. Criteria aids the SQA in determining whether the Factor being revised is present” [Vincent, 1988]. Vincent states four reasons, gather from McCall’s work, for creating Criteria for each Factor.

1. Criteria offer a more complete, concrete definition of Factors.

2. Criteria common among Factors help to illustrate the interrelation between Factors.
3. Criteria allow audit and review metrics to be developed with great ease.
4. Criteria allow us to pinpoint area of quality Factor that may not be up to a predefined acceptable standard.

Table 3. CRITERIA OF SOFTWARE QUALITY FACTORS

Criterion	Definitions
Traceability	Those attributes of the software that provide a thread from the requirements to the implementation with respect to the specific development and operational environment.
Completeness	Those attributes of the software that provide full implementation of the functions required.
Consistency	Those attributes of the software that provide uniform design and implementation techniques and notation.
Accuracy	Those attributes of the software that provide the required precision in calculations and outputs.
Error Tolerance	Those attributes of the software that provide continuity of operation under nonnominal conditions.
Simplicity	Those attributes of the software that provide implementation of functions in the most understandable manner.
Modularity	those attributes of the software that provide a structure of highly independent modules.
Generality	Those attributes of the software that provide breadth to the functions performed.
Expandability	Those attributes of the software that provide for expansion of data storage requirements or computational functions.
Instrumentation	Those attributes of the software that provide for the measurement of usage or identification of errors.
Self-Descriptiveness	Those attributes of the software that provide explanation of the implementation of a function.
Executive Efficiency	Those attributes of the software that provide for minimum processing time.
Storage Efficiency	Those attributes of the software that provide for minimum storage requirements during operation.
Access Control	Those attributes of the software that provide for control of the access of software and data.
Access Audit	Those attributes of the software that provide for an audit of the access of software and data.
Operability	Those attributes of the software that determine operation and procedures concerned with the operation of the software.
Training	Those attributes of the software that provide transition from current operation or initial familiarization.
Communicativeness	Those attributes of the software that provide useful inputs and outputs which can be assimilated.
Software Systems Independence	Those attributes of the software that determine its dependency on the software environment (operating systems, utilities, input/output, and routines.)
Machine Independence	Those attributes of the software that determine its dependency on the hardware system.
Communications Commonality	Those attributes of the software that provide the use of standard protocols and interface routines.
Data Commonality	Those attributes of the software that provide the use of standard data representations.
Conciseness	Those attributes of the software that provide for implementation of a function with a minimum amount of code.

1.5 Technical Reviews

Technical reviews are used in software engineering to verify the software process of the project. The four major technical reviews most commonly used are Peer Review, Audits, Walkthroughs, and 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. Nonetheless, peer reviews can also be used to verify documentation that is being produced such as the Requirements Analysis document.

Audit - 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, Quality Assurance Engineer (QAE), is responsible for validating changes in the report. The QAE makes sure that a notification is sent to all the participants of the Audit.

Walkthroughs - A walkthrough is an informal review that is used to evaluate most of the stages (Requirement Analysis, Design, Testing, and Implementation) in the Software Development Life Cycle. 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/author, and two to three reviewers. During the review process, the reviewers check the documents to make sure that the document is clear and understandable and that it is up to standards. Other types of revisions that a reviewer can make are grammatical errors.

The walkthrough is basically composed of three steps that, if followed, that help assure a successful walkthrough in which more errors are found and corrected before moving into the next cycle. The three steps for conducting walkthroughs are planning, conduction and follow-up.

In the planning stage, the goals are determined and everyone participating in the review receives a notice from the moderator. This notification has all the information necessary to prepare for the walkthrough. The moderator then determines if everyone is prepared for the walkthrough. If they are, the moderator will allow the walkthrough to take place. The stage when everyone meets to have the walkthrough is called the Conduction Stage. Finally, after the walkthrough, the moderator does a follow up on the author of the document to make sure that all of the points raised have been clarified and corrected.

Inspections - This technical review should is as a more formal review from which more feedback can be obtained than from Walkthroughs, Audits, or Peer Reviews. Inspections require a higher degree of preparation of the review participants, but the benefits include a more systematic review of the software and more controlled and less stressful meeting. Formal software inspections are in-process technical reviews of a product conducted to find and eliminate defects. The major difference between Walkthroughs and Inspections is that an Inspection process involves the collection of data that can be used to give feedback on the quality of the development and review process.

The participants in an Inspection are similar to the Walkthrough but on this technical review, all of the participants go through more steps to complete the review. These steps are those suggested by NASA's Office of Safety and Mission Assurance in their article in the Software Formal Inspection Guidebook [NASA, 1993].

- Planning
- Overview
- Preparation
- Inspection Meeting
- Third Hour
- Rework
- Follow Up

At the Planning stage, the moderator checks the document to be revised and decides whether it is ready to be revised. The moderator then decides whether everyone participating in the Inspection knows the product that is being revised. If not, the moderator will ask the author to give an Overview of his/her document. Once everyone has been introduced to the product submitted for Inspection, the moderator prepares an Inspection notification sheet and distributes it to the participants. This sheet contains all the information about the Inspection. If the time allocated for the Inspection is not enough, a third hour is called to finish the Inspection. Once the Inspection and the third hour (if needed) are complete, the author has a specified amount of time to correct his/her document. The moderator then does a follow up to make sure that the author has corrected the product that was submitted for review.

1.6 Conclusion

This chapter gave an overview of the quality methods necessary to produce systems that are error free. It also covered software engineering and how it has allowed software systems to be created more systematically, faster and easier. A description of the roles played during software engineering was also introduced to show how they add value to producing software engineering systems.

Moreover, Section 1.3 showed the importance of managing the quality of software systems. It introduced the roles within software engineering that help the Quality Assurance Engineer (QAE) verify the quality of a project. Section 1.3 also introduces how costly errors are if they are not corrected at the early stages. The longer you wait to fix an error, the more costly it will be.

Finally, an introduction is given of how quality in software projects is checked by performing technical reviews throughout the development process. Factors and Criteria, which are parameters that help to evaluate how good a software product is, are used while conducting technical reviews.

Chapter Two

2 The DiSEL Project

The Distributed Software Engineering Laboratory (DiSEL) allowed the Quality Assurance Engineer (QAE) to manage the quality of a distributed Software Project. DiSEL consisted of nine students from the Massachusetts Institute of Technology (MIT) and four students from Centro de Investigacion Cientifica y de Educacion Superior de Ensenada (CICESE), an Institute of higher education in Ensenada, Mexico. The project was setup to be operated like a small company. Every member participated in every cycle of the project as each was assigned two roles, a primary role and a secondary role (The roles played by the members of DiSEL were similar to the ones played in professional Software Engineering projects). Even though the members of the team were playing their first role most of the time, they still needed to know the secondary role in case they were asked to perform it. Every member also had a counterpart from the other university. The time frame for DiSEL was two semesters (nine months).

In distributed projects such as DiSEL, there are many challenges that arise. Besides software engineering problems, DiSEL participants experienced communication problems because the project was being conducted between two different languages

(Spanish and English) and many cultures (Mexican, American, Indian, Turkish, Korean, Uruguayan, and French).

Although there are many areas of research, this thesis focuses on the issues related to Quality Assurance. Other aspects of software engineering practices such as requirement analysis, design, implementation or communications between different languages and cultures are beyond the scope of this research.

2.1 The Team

As mentioned above, the team consisted of nine students from MIT and four students from CICESE. The students were assigned roles as follows. On the first of day of classes, the students were given a test in order to show their knowledge of software engineering. After they took the exam, the students were asked to choose two roles and rank them in order of preference. Students were able to choose from the following roles.

- Project Manager
- Quality Assurance Engineer
- Software Configuration Manager
- Documentation Specialist
- Requirement Analyst
- Designer
- Programmer
- Test Engineer

At the end of the day, the professors took the exams and the different choices from the students. A week later, the professors announced the roles that the students would be playing. The decision was made based on the two choices that the students had plus how they performed on the exam. Every member of team was assigned two roles, a primary

and a secondary role. They also had a member from the other university within their primary role. This would make sure that collaboration between the two universities took place.

Since the students did not know what the jobs were for the different software engineering roles, the professors provided definitions of all the roles in software engineering. The definitions of the software roles are as follow.

Project Manager – “Responsible for providing overall leadership to the team,” [Yang, 1998] and creating project updates through the software development cycles. Responsible for creating the project schedule and making sure that every member of the project is aware of the deliverables for every phase of the software life cycle.

Quality Assurance Engineer – Responsible for making sure that every member of the project is following the standards of software engineering. Responsible for developing the Quality Assurance Plan that will contain the schedule of all the technical reviews. The QAE will conduct the review and will be the person who will approve the final deliverables of every phase.

Software Configuration Manager – Responsible for identifying and defining items in the system, controlling the change of these items throughout their life cycle, and verifying the complete and correctness of these items [IEEE, 1997].

Documentation Specialist – Responsible for maintaining a repository where all the documents and code will be stored for future use. He/she will also be the person who will take minutes of all the projects meeting.

Requirement Analyst – This is the person responsible for interviewing the client and gathering requirements for the project. Responsible for writing the software requirement documents where all the specifications are clearly stated. The software requirements documents should be detailed enough so that the test engineers can start creating the user test cases.

Designer – Responsible for turning the requirements specified by the requirement analyst into the design architecture. The design document should be detailed enough so that the programmer will not have trouble in implementing the design.

Programmer – Responsible for turning the design document into actual code. Should be able to know how to work on a code that is being used by different people since the project will be distributed.

Test Engineer – Responsible for testing the implementation of all the code by the different programmers.

As already mentioned above, the team members of DiSEL course were assigned more than one role. The minor role of each member required him/her to be on call

whenever the other members of his/her secondary role needed extra assistance. The project manager made sure that everyone played their own roles and if a role needed extra assistance, the secondary roles were brought in to aid the role that was overwhelmed with work.

Once the project got to the design phase, the responsibilities assigned to each role changed. The designer and programmer roles were merged and two design/programmer groups were formed. One of these groups represented the 3D Interface group and the other group was the User Interaction Awareness group. The reason for this change was to have more collaboration between the CICESE and MIT students. For each of the two groups that were formed, there was at least one member from each university. This made the communications between the two schools a requirement.

Figure 5 shows the project organization of DiSEL with the two new groups containing the designers and programmers. To understand how primary and secondary roles appear on Figure 5, the following method was used. Primary roles appear at the top while secondary roles appear at the bottom.

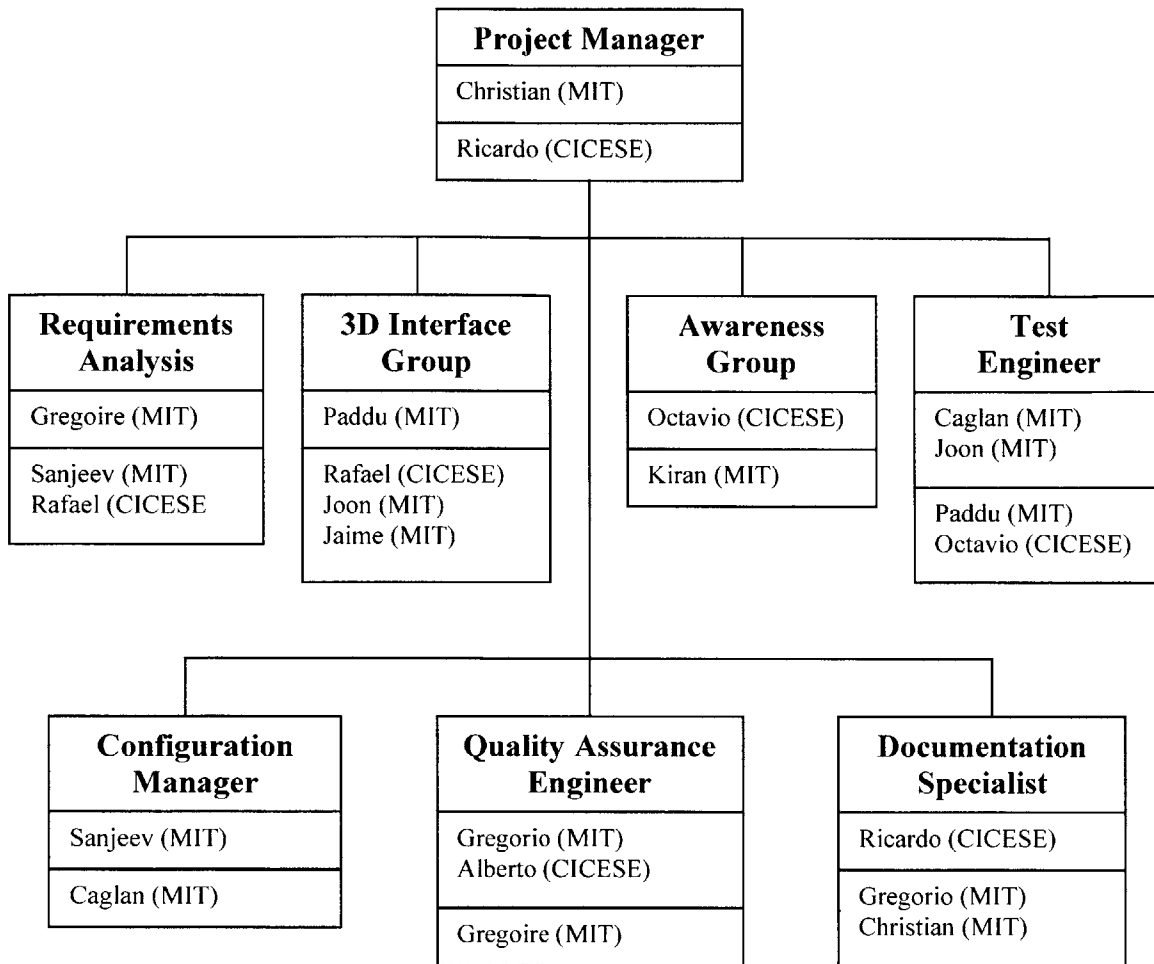


Figure 5. PROJECT ORGANIZATION

2.2 Objective

The objective for the DiSEL project was to provide experience in software engineering practices and process in a physically distributed atmosphere to students by making them collaborate with students from other nationalities (Mexico). Students were expected to assume a primary and secondary role for the duration of the project. This allowed every team member to learn more than one role. Furthermore, the leadership of

the roles was divided between the two participating universities (CICESE and MIT) in order to give equal power of decision.

Throughout the project (one academic year), students were educated on the issues and complexities involved in medium-scale software development production taking place in a distributed environment [Yang, 1998]. At the end of the project, the group was expected to produce a system that would facilitate the interaction between distributed teams. There was only one restriction for the project. The DiSEL team would have to build the new system on top of CAIRO, a system that had been created by a former MIT PhD Student, Karim Hussein [Hussein, 1998].

In addition to learning software engineering practices, the DiSEL team was also asked to participate in the 50K-Entrepreneur competition at MIT (MIT 50K, 1998). This allowed the members of DiSEL to learn how to develop and sell an idea in nine months. Thus, the students were asked to present an idea and develop a business plan to be presented to a venture capitalist.

2.3 Environment

Even though the project asked the team members of DiSEL to go through the whole software development cycle, DiSEL was still different than a commercial Software Engineering project. To begin with, the students came into the project knowing little about software engineering. Since most of the students in the DiSEL course did not know software engineering, lectures were given every week on Tuesdays and Thursdays. This provided the training similar to the training a professional gets once they get to their

jobs. The professors from both universities, Favela (CICESE) and Pena-Mora (MIT), took turns teaching the different areas of software engineering (project manager, quality assurance, configuration manager, analyst, designer, programmer, tester, and documentation specialist).

For the first semester, classes were given every Tuesday. On Thursday, the second meeting day, laboratories were carried out. During laboratories the students were able to show how much they had learned. Also, during laboratories, updates were given by the different management positions (project manager, quality assurance engineer, and configuration manager).

The setup for the class for lectures and laboratories can be seen in Figure 6 below. In it you will see how the microphones, computers, projectors and camera were setup. Another difference between DiSEL and a real software project was the time that was spent on the project by the students. The students would spend around ten to fifteen hours a week working on the project, which is less than the time spent by professional software engineers. Thus, the turn around for the students was slow and it took about six to eight weeks for the DiSEL members to learn their roles before they could start to work in the project's development cycles correctly.

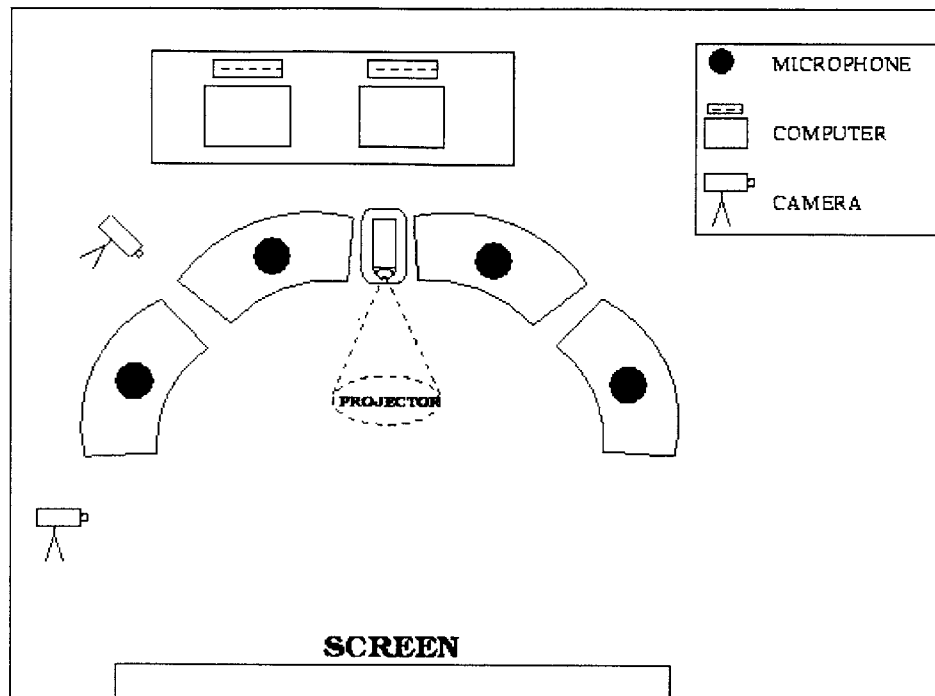


Figure 6. CLASSROOM ENVIRONMENT

2.3.1 Communications

The communication between the two universities (MIT and CICESE) was conducted over the Internet using Microsoft NetMeeting. Every Tuesday and Thursday the Teaching Assistant would setup the MIT classroom. The equipment needed for the setup of the classroom consisted of two video cameras, one projector, and four microphones (See Figure 6 above). The video cameras and microphones were used to exchange audio and video. Both, video and audio were transferred using NetMeeting. The projector was used to project the video from the opposite university along with the lectures or work that was being presented. The CICESE classroom did not need to be setup since they had a dedicated room with all the equipment needed for the communication.

2.4 Process Model

The process model use in DiSEL was a combination of the waterfall model (see Figure 7) with the spiral model (see Figure 8). The waterfall model allows engineers to have control of the project since they know the phases that follow each cycle of the project. According to Boehm, “waterfall models have much contributed to the fact that the process run has become disciplined, visual and controllable” [Boehm89, Wallmuller94].

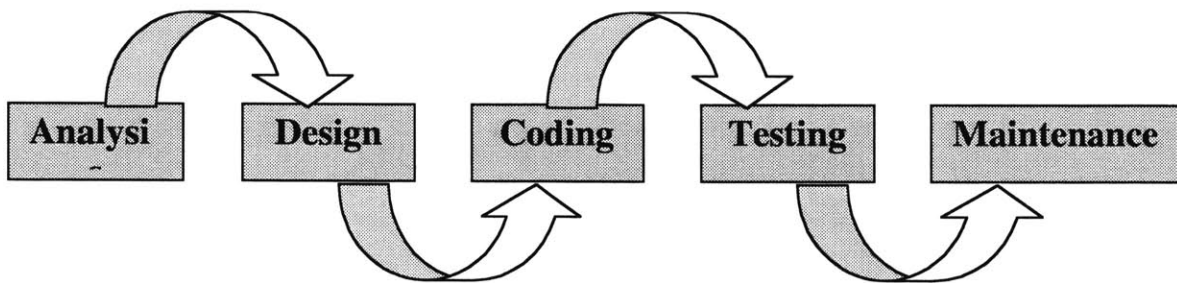


Figure 7. WATERFALL MODEL

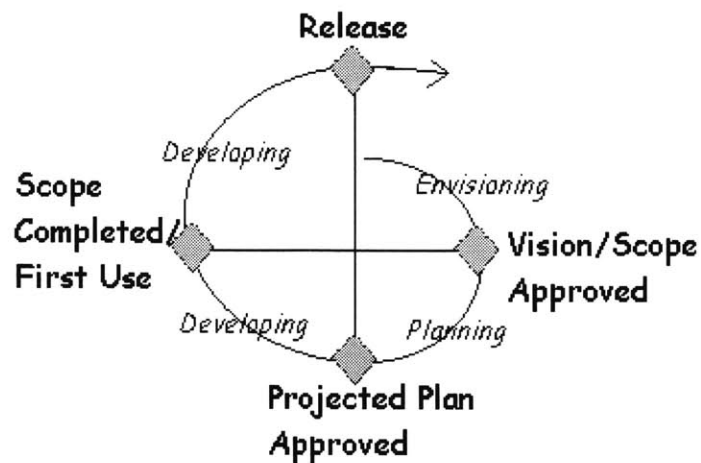


Figure 8. SPIRAL MODEL

On the other hand the spiral model is a process model that takes risks into account since it goes through several cycles before completing the final cycle. Boehm was the first person that “drew attention to the importance of being in command of risk in the development process. He considered this so important that he suggested a self-constructed process model,” the spiral model [Boehm, 1989 and Wallmuller, 1994]. By using both models, the project manager of DiSEL knew that he would have full control of the project.

2.4.1 Application

The project started by using the waterfall model as the members of DiSEL went through the Systems Engineering and Software Requirements Phases. As the project got to the design phase, the waterfall model was changed to the spiral model. Since most of the activities within DiSEL were related to the design, implementation and testing, the project manager decided to have three sequential cycles with each cycle producing one version of the CAIRO. At the end of the three cycles within the spiral model, the project went back to the waterfall model to work with the Operation ad Maintenance phase.

2.5 Conclusion

This chapter introduced the DiSEL Project by showing who the participants were, the objective of the DiSEL project, the environment in which it was taught, and the software process model that was used. All of these give a foundation for a similar project to take place in other universities.

Chapter Three

3 Applying Quality Assurance to DiSEL

There are a lot of tasks that need to be accomplished by a Quality Assurance Engineer (QAE) within Software Engineering projects. Having a distributed project such as the Distributed Software Engineering Laboratory (DiSEL) increases the amount of work that needs to be performed since the QAE needs to manage the quality process of two or more groups distributed geographically.

The QAE conducted research to find out all the literature needed for conducting quality assurance to software engineering projects. While reading through the literature, little or no research was found that showed how to manage quality assurance in distributed projects. Also, no information was found that showed how QAEs should work while they are geographically distributed. This proved challenging as in DiSEL one of the QAEs was at the Massachusetts Institute of Technology (MIT) and the other was at Centro de Investigacion Cientifica y de Educacion Superior de Ensenada (CICESE), a graduate university in Ensenada, Baja California, Mexico.

In this chapter I will focus on the implementation of quality assurance to a distributed Software Engineering project, DiSEL. To accomplish this, I will rely on the Quality Assurance Plan (QAP) that was developed by the QAE. This plan was the source that guided the QAE throughout the software development process. But before I go into the QAP, I will talk about how two members of the quality group integrated their work.

3.1 Role Integration

The two-person quality assurance team had one member at MIT and the other at CICESE. This was assigned by the professors who view this distribution of team as a form of having collaboration between the members of MIT and CICESE.

Language Barrier - All of the classes, including the laboratories, were conducted in the English Language. Communication between the two teams was slow since it took time for one of the QAEs to feel comfortable with the English language. When the Quality Assurance Plan was being created, there was little feedback from one of the QAEs. The belief is that that QAE was not comfortable with the language and it took him a while to get use to adapt to the English requirement.

Academic Barrier - An Academic Barrier occurred because one of the QAEs had already taken courses in software engineering while one QAE had not taken any. This was very frustrating for the QAE who already knew the process to follow for good quality assurance. For the QAE who had little knowledge about software engineering, it was very frustrating since he had to work twice as much to be on the same level as the other QAE.

Throughout the different stages of the project, the QAEs were always in disagreement over of what should be the quality assurance priorities. This dissention arose from the fact that one of the QAEs was better prepared.

Technology Barrier – The main problem with technology was that it was very hard to get real time communication established between the two QAEs. E-mail was not enough since one QAE was not able to check his e-mail regularly. Consequently, whenever there was feedback given through e-mail, it was too late to be useful.

In addition, the equipment necessary for carrying out real time communications with audio and video was not always accessible. All of these made the collaboration between the two members of the quality team very difficult. In the end, each member of the team started working on their own and there was no agreement about what each of the members should be doing.

3.2 Quality Assurance Plan

The work perform by the QAE is encapsulated in the Quality Assurance Plan (QAP). This QAP is the source that guides the QAE throughout the software development process. It enables the QAE to perform the tasks necessary to achieve a project of the highest quality by signaling the different functions the QAE needs to perform during the different software cycles. The sections following focus on the QAP since it was the main reference that aided the QAE when making sure that the software

process was conducted appropriately. This document will also state the problems that came up as the QAE tried to implement different aspects of the QAP.

To create the Quality Assurance Plan (QAP), the Quality Assurance Engineer (QAE) used the *IEEE Software Engineering Standards* from 1997 (IEEE, 1997). These standards were used since they are the most commonly used standards for writing software engineering documentation. These standards were also the ones used for the creation of all the documents within DiSEL.

In all, the QAP serves as the central aid for planning and checking quality assurance through out the software developing life cycle. This plan contains all the quality measurements needed to evaluate a distributed software project, and consequently, it describes how quality control should be implemented.

The first item that needed to be addressed by the QAE was to define the different phases within the software project and the functions that would be performed at each phase. Within DiSEL, there were six software phases were defined. Tables 4 below shows these phases along with the functions that were established at the beginning of the project by the QAE.

Table 4. TASKS OF THE QAE DURING THE SOFTWARE DEVELOPMENT PROCESS

PHASE	FUNCTIONS
<i>Systems Engineering</i>	<ul style="list-style-type: none"> • Define the Quality Assurance Engineer's functions and the Software Quality Assurance Plan. • Review the software development plan and audit procedures done by the Project Manager. • Participate in the meeting with the client to obtain the software requirements.
<i>Requirement Analysis</i>	<ul style="list-style-type: none"> • Review the Software Project Manager's plan and develop audit procedures. • Participate in the requirement specification review. • Implement the Software Quality Assurance Plan. • Review tools, techniques and methodologies used in the software development process.
<i>Design</i>	<ul style="list-style-type: none"> • Participate in all the preliminary and detailed design reviews. • Review the preliminary and final design document. • Review the preliminary and final operator's manual. • Review preliminary test plans.
<i>Coding</i>	<ul style="list-style-type: none"> • Participate in some code inspections. • Monitor use of defined tools, technologies and methodologies. • Review final test plans. • Audit code errors for correction. • Audit change request record tracked by the Software Configuration Manager.
<i>Integration</i>	<ul style="list-style-type: none"> • Audit baseline record established by the Software Configuration Manager. • Audit development records tracked by the Software Configuration Manager. • Witness the integration tests and acceptance test done by the Test Engineer. • Certify test reports. • Audit for correction of errors encountered in the integration tests.
<i>Operation / Maintenance</i>	<ul style="list-style-type: none"> • Review the changes proposed by the user/customer. • Audit the change record. • Audit for correction of customer problems. • Inspect updated documents: Requirements Specification, Preliminary and Detailed Design, Operator Manual, Error and Changes record

Although the QAE set out to do the functions mentioned above, problems came up that impeded the QAE from performing his work as planned. To begin with, as already stated in Chapter 2, many of the students within DiSEL had little knowledge about software engineering. This slowed the work process for most of the students. Consequently, many of the documents that were expected to be finish at certain phases were not done. Therefore, the QAE was unable to perform his work as planned. Nonetheless, creating the table above enabled the QAE to prepare for future work as it defined the functions that he would perform during the development stage.

3.2.1 Management

Management is the next task that the Quality Assurance Engineering (QAE) needed to define. This allowed the QAE to structure how he would work with the members of the team.

To manage the quality of DiSEL, the QAE set out to work closely with the documentation specialist, test engineer, and configuration manager. This group of three roles was assigned by the QAE as primary roles since they worked closely with each other. The remaining roles comprised the secondary roles.

Primary Roles - As already stated above, the primary roles were the people who worked a lot with the SQAE engineer. Figure 9 below diagrams the primary roles. All four roles are essential for the project because they manage the project from the quality view.

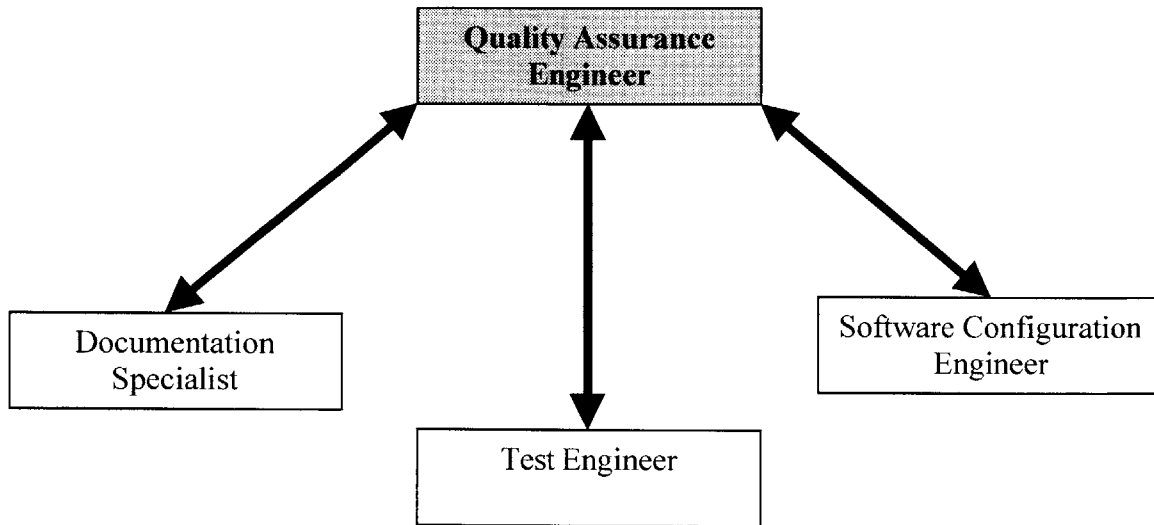


Figure 9. QUALITY ASSURANCE ORGANIZATIONAL DIAGRAM

After identifying the primary roles, the QAE stated the interaction that needed to be carried out in order to accomplish quality assurance within the project. Table 5 describes the interaction that was expected between the primary roles mentioned above and the QAE.

Table 5. INTERACTION BY QAE AND PRIMARY ROLES

ROLE	INTERACTION
Documentation Specialist	<ul style="list-style-type: none"> • QAE will work with the Documentation Specialist to make sure that the documentation created during the project meets the software standards for documentation. • The QAE will verify that the Documentation Specialist has created a repository for all the documents produced throughout the project for easy access.
Test Engineer	<ul style="list-style-type: none"> • The QAE will make sure that the Test Engineer’s Plan is adequate to the project and is being performed through all the software development phases. <p>The QAE will work closely with the Test Engineers to make sure that the software product is free of errors.</p>

Table 5. INTERACTION BY QAE AND PRIMARY ROLES (CONTINUE)

ROLE	INTERACTION
Test Engineer	<ul style="list-style-type: none"> • The TE will test the code developed by the programmers and will trace back to the software requirements document to make sure that the deliverables accord with the client's requirements. • The TE will create test reports that would be audited by the QAE.
Software Configuration Manager	<ul style="list-style-type: none"> • The QAE will make sure that the Software Configuration Manager's Plan is adequate to the project and is being performed through all the development phases. • The QAE 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.

Once again, the QAE was unable to carry out part of the interactions that he wanted from the primary roles. The major problem was that the documentation specialist was on the team at the university in Mexico, Centro de Investigacion Cientifica y de Educacion Superior de Ensenda (CICESE). Thus, it was very hard for the QAE to verify that the documentation specialist was doing the job he was expected to do.

In addition, since the Software Configuration Manager was unable to get the documents on time, he was unable to have a control of all the documents. This eventually hurt the quality of the project since the QAE was unable to manage the control versioning of documents and code.

Secondary Roles - Besides the interaction with primary roles, the QAE also needed to state the interactions that will take place with the other roles. These roles can be seen as secondary roles because even though the QAE interacts with them, it does not require

them to contribute to the assurance of the software product. Therefore, the interaction between the secondary roles can be seen as external to the Quality Assurance Group. The interaction between the QAE and the secondary roles can be seen in Table 6.

Table 6. INTERACTION BY QAE AND SECONDARY ROLES

ROLE	INTERACTION
Project Manager	<ul style="list-style-type: none"> The QAE reviews the development plan to ensure that it is created and followed. The QAE will elaborate quality assurance reports that would be included into the project manager's progress report.
Analyst	<ul style="list-style-type: none"> The QAE reviews the software requirement document to ensure that is accurate and completely represents the expectations of the customer. The software requirements documents must also be clear enough to everybody in the developer's group, especially to the designer who will be implementing the specifications of the requirements and the test engineers who will be elaborating test cases.
Designer	<ul style="list-style-type: none"> The QAE 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.
Programmer	<ul style="list-style-type: none"> The QAE 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 maintainer.

Once again, the problem of not everyone being prepared slowed the process of the project and consequently, documents were not created on time.

3.2.2 Schedule

After stating the interaction that will take place between the Quality Assurance Engineer (QAE) and the member of the team, the QAE created a schedule of events. The schedule of events contains the date of the event, the event itself, the documents that will

be reviewed, and the people involved during these events. Table 7 below shows all the events that were schedules starting from January 1999.

Table 7. *QUALITY ASSURANCE SCHEDULE*

Quality Assurance Schedule			
Date:	Event:	Documentation Involved:	Role Involved:
4-Feb-99	Presentation of QA Plan.	Quality Assurance Plan.	Quality Assurance Engineer.
9-Feb-99	Walkthrough of Project Manager's Plan.	Project Manager's Plan.	Project Manager and Quality Assurance Engineer.
11-Feb-99	Lecture on Technical Reviews. Turn in Design Document v2.1.	Technical Review Document and Design Document v2.1.	Quality Assurance Engineer and Designers.
16-Feb-99	Audit Desing Document v2.1.	Design Document v2.1.	Designers and Quality Assurance Engineer.
18-Feb-99	Update of SCM and Doc. Specialist. Turn in Code.	SCM and Doc. Specialist Update Document, and Code Turn in.	Software Configuration Manager and Documentation Specialist. Programmers.
20-Feb-99	Turn in Testing Report v2.1.	Testing Report v2.1	Test Engineer.
25-Feb-99	Audit to Test Report v2.1. Turn in Design v2.2	Test Engineer's Report v2.1 and Design Document v2.2.	Quality Assurance Engineer, Test Engineer, and Designers
2-Mar-99	Audit Design or VRML and Awareness v2.2.	Design of VRML and Awareness v2.2.	Designers and Quality Assurance
4-Mar-99	Update of SCM and Doc. Specialist Turn in Code.	SCM and Doc. Specialist Update Document, and Code Turn in.	Software Configuration Manager and Documentation Specialist Programmers.
9-Mar-99	Turn Design Document v2.3.	Design Document v2.3.	Designers and Quality Assurance Engineer
11-Mar-99	Audit Design Doc v2.3 and Test Report v2.2.	Design Document v2.3 and Test Reprot v2.2.	Quality Assurance Engineer, Test Engineers and Designers.
16-Mar-99	Audit Test Report v2.3	Test Engineer's Report v2.3.	Quality Assurance Engineer and Test Engineers
18-Mar-99	Update of SCM and Doc. Specialist Turn in Code.	SCM and Doc. Specialist update Document, and Code Turn in.	Software Configuration Manager and Documentation Specialist Programmers.
30-Mar-99	Walkthrough of Final Design Document	Final Design Document	Designers and Quality Assurance
13-Apr-99	Walkthrough of Final Test Report	Final Test Report	Test Engineers and Quality Assurance Engineer
15-Apr-99	Presentation of Manual	Manual	Documentation Specialist

3.2.3 *Software Documentation*

The next task for the Quality Assurance Engineer (QAE) was to define the documents that would be revised along with the contents it should have. The documents

produced by the different members of the team are revised in order to make sure that they are clear, understandable, comply with the *IEE Software Engineering Standards*, and are inline with what the clients or market may ask to be developed.

The QAE conducted technical reviews (see Section 3.1.3) to the documents that are stated below to assure quality.

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

Design - 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, the easier it will be for the Test Engineers to come up with test cases. Audits throughout the development of the Top Level Design/Software Specifications will be conducted by the QAE. At the final stage of the Design Document, a walkthrough of inspection will also be conducted by the QAE for verification purposes.

Project Manager's 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 QAE to check compliance with the standards in writing the Project Manager's plan.

Software Configuration Manager's Plan - The plan must state how the Software Configuration 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 QAE to check compliance with the standard in writing the Software configuration Manager's Plan.

Test Engineer's 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 QAE to verify compliance with the standards in writing the Test Engineer's Plan.

Test Engineer's Reports - The test engineers will create reports with the result from the revision made to documents as well as code. Audits will be conducted by the QAE after every revision to verify the process the Test Engineers are implementing while revising the code.

Documentation Specialist's Plan - The plan must state the standards that the team must follow to create HTML, Word and PowerPoint documents. The Documentation

Specialist must also state how he/she will be managing DiSEL's Web Repository. A walkthrough will be conducted by the QAE to verify the standards.

3.2.4 Technical Reviews

After specifying the documents that will be revised for quality assurance, the Quality Assurance Engineer (QAE) must define how the different technical reviews will be conducted. Technical Reviews are the mechanisms that aided the QAE to assure quality within the documents being produced. There are many variations to performing Technical Reviews. Most of these approaches involve a group meeting to assess a work product; however, some variations of reviews do not require a group meeting. The four major technical reviews that were used in the DiSEL project were.

- Peer Reviews
- Audits
- Walkthroughs
- Inspections

3.2.4.1 Peer Reviews

Peer Reviews are the most informal of all technical reviews because they do not involve any preparation. Peer Reviews were introduced by the Quality Assurance Engineer (QAE) to check the programmers' code for errors, prior to execution or compilation. However, Peer Reviews can also be used to revise documents developed by the development team. Issues that should be revised while doing peer reviews are correctness, misuse of variables, omitted functions, poor programming practices and redundancy.

3.2.4.2 Audits

“Audits serve to insure that the software is properly validated and that the process is producing its intended results” [Walker, 1979]. In an Audit, the review leader is responsible for validating changes in the report and to notify the Configuration Managers to establish a version control of the document.

The QAE performed audits throughout a development of a cycle to make sure that the work being done was of the highest quality. Out of all the cycles, the QAE audited the Design and Testing phase. Since the team was using the Spiral Model during those phases, every time a new version of the Design and Testing Documents was produced, the QAE audited them to make sure they were readable, understandable and aligned with what the was stated in the Requirements Document.

Preparation and Conduct - Audits need to be prepared and conducted as follows. The QAE would make the preparations of the Audit by assigning the personnel to conduct the review. An Audit notification was then sent to the participants (see Appendix 1). Within DiSEL, the reviewer was always the QAE. The QAE would review the document and fill out the preparation log (see Appendix 2). This preparation log was then given to the author of the document so he could make the appropriate changes.

3.2.4.3 Walkthroughs

A Walkthrough is an informal review that evaluates many of the documents specified in Section 3.1.3. The goals of this informal review should be determined prior

to conducting the Walkthrough and are identified in the notice announcement of a walkthrough (see Appendix 3).

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, author, documentation specialist, and two to three reviewers. While conducting a Walkthrough, some of the questions that a reviewer should have in mind are:

- Is it up to standards? In the case of DiSEL, the standard to follow are the IEEE' Software Engineering Standards.
- Does it do what it was supposed to do?

Other types of revisions that a reviewer can make are grammatical. This will include "style errors, improvement in the quality of the material, and the transfer of ideas and understanding between team members" [Evans, 1986]. Once a material has been reviewed in a Walkthrough, the end product will serve as the baseline for the next phase in the development process. For instance, once the Requirements Document has been reviewed, it will serve as the baseline for the design of a software project. The documents that were chosen to be revise using this kind of technical revision are stated above in Section 3.2.3.

From the documents stated in Section 3.2.3, the Project Manager's Plan and the Requirements Document were the only documents that got revised by the QAE. The

other documents were not revised because the process became very accelerated in which documents were not prepared with enough time for a proper review.

Following are the Planning and Conduct stages of a Walkthrough. First the planning of the review is stated followed by the steps that need to be followed to conduct the Walkthrough. The person responsible for planning the Walkthrough and conducting it will be the moderator. In DiSEL's case, the Quality Assurance Engineer played the role of the moderator.

Planning – The moderator is the person responsible for reminding the members of the team when a Walkthrough will be taking place well in advance. A Walkthrough notice (see Appendix 3) that includes all the participants, the objective of the review, the date, time and place should be sent at least one week prior to the Technical Review. The actual review should be composed of the following members:

- Moderator
- Documentation Specialist
- Two to Three reviewers
- Author
- Supervisors

The Supervisors (Professors and Teaching Assistants) are invited to Walkthroughs when their particular skill or knowledge sets are required. The supervisor will not attend as a supervisor role, rather as one member of a team reviewing another member's work. The time frame for a Walkthrough should be about 90 minutes. If additional time is required,

an additional Walkthrough should be schedule for another time. The date, time, and place for the Walkthrough should be stated in the announcement sheet.

In the DiSEL project, all of the Walkthroughs were conducted during laboratory hours on Thursday.

Conduct - At the time of the Walkthrough, the moderator will determine if all the participants are prepared for the revision. If the team is not prepared, the moderator has the power to re-schedule the walkthrough. If this Walkthrough is the second part of a previous Walkthrough that was not finish, the moderator will verify that the action items that were given in the first meeting have been resolved before continuing with the Walkthrough.

Being the most important person in the meeting, the moderator will be responsible for:

- Conducting the Walkthrough
- Creating an agenda that will be used during the meeting.
- “Ensuring that the review meets its objective in an efficient manner” [Evans, 1986].
- Opening and closing the meeting
- Soliciting comments from the reviewers
- Presenting any of the reviewers comments
- Monitoring discussion to ensure they are relevant to the subject

Overall, the moderator will be the person “responsible for arbitrating disagreements to successful conclusions” [Evans, 1986]. If there is no conclusion to an item, the Documentation Specialist writes the item in the action item list (see Appendix 4).

The Documentation Specialist is responsible for writing the minutes of the meeting. As stated above, she/he will also write down the items that were not resolved on the action item list. The Documentation Specialist is also be responsible for collecting minor errors, such as grammar, punctuation, and style errors, that the reviewers found. Once the Walkthrough is finished, the Documentation Specialist will read the items not resolved to the group to verify that they are correct. The Documentation Specialist will then give the action item list along with the minor errors collected to the author of the document.

The reviewer will be the person who will be evaluating the document presented for revision. For the DiSEL project, the reviewers were asked to verify that the documents were written in accordance with the *IEEE Software Engineering Standards*. In addition, they checked to see that the author of the document had stated their ideas in a clear and concise manner. The reviewers would review the document and would put down all errors found in a preparation log similar to the one used by Audits (see Appendix 2).

At the time of the review, the author of the document will step through the material while the reviewers give feedback by staying the errors they found. The author of the document should address questions raised by the reviewers. If no solution is found to the question raised, the Documentation Specialist will write the issue in the action item list. “No specific solution discussion shall take place during the walkthrough” [Evans, 1986].

At the end of the Walkthrough, the author will take the action item list and the minor comments made by the reviewers from the moderator. He will use this as reference for correcting the errors. The project manager will be the person who will do a follow-up on the author to make sure that all of the action items have been addressed and resolved.

At the end of the review, the moderator will see if an additional Walkthrough is needed based on whether the material is incomplete or contains too many errors. Also, the group shall decide whether to approve or not approve the material. The approved/not-approved status shall be noted on the action item list by the recorder in the "new status" space. If the document is approved, it will be frozen and used as a baseline for the next step in the developing process and it is put into the project program support library.

3.2.4.4 Inspections

Inspections should be presented as a more formal approach that can be viewed 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 stressful 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" (NASA, 1993). The major difference between Walkthroughs and Inspections is that an Inspection process involves the collection of data that can be used for feedback on the quality of the development and review process.

The documents that were schedule to be inspected were the final Design Document and the manual of the system that were written. Below you will find the steps required for the implementation on an Inspection to the documents mentioned above.

Planning - This is the “period of time [that] determine whether the product to be inspected meets the entry criteria" [NASA, 1993]. The person responsible for planning and carrying out the whole process of an Inspection is the moderator. The moderator should be a person who understands the whole process of the Inspection and is aware of the documents that are being presented. The Quality Assurance Engineer (QAE) is the person who is recommended to play the role of the moderator.

For the planning of the Inspection, the moderator should first check the document to see if it is up to standards. Based on whether or not the document is up to standards, the moderator will make a decision on whether the Inspection should take place. If not, the moderator will ask the author of the document to work more in bringing the document to standards.

After making sure that the document is ready to be inspected, the moderator will fill out an Inspection Announcement Sheet (see Appendix 5) that will be distributed to all the members of the team. This sheet will contain all the information that is needed for carrying out an Inspection. This information will contain the members of the team who will participate (moderators, author, reader, reviewers, and documentation specialist) during the Inspection. Also, it will state the time, date, and place of the reviews. Along

with the announcement of the inspection, the following material should also be distributed to the participants.

- Background on document being inspected
- Material to be inspected
- Inspector's preparation Log
- Inspection checklist (used by the moderator)

After the above material has been distributed to all the participants, the moderator will check to see if the participants of the review are familiar with the material that is being revised. If not, the moderator will ask for an overview to take place.

Moreover, the moderator will check the document's size to determine if it can be revised during one review session. If not, an additional review session should be schedule to finalize the review of the whole document. To check the whole process of the Inspection, the moderator will be using an Inspection checklist (see Appendix 6). This sheet will aid the moderator as he goes through the whole process of this technical review.

Overview - An overview is "the optimal stage in which the inspections team is provided with background information for the inspection" [NASA, 1993]. The moderator will determine if this stage of the Inspection should take place. The moderator will decide to schedule an overview if one or more of the following circumstances apply:

- The inspection team is not familiar with the product.
- The product is new or is being inspected for the first time.
- Inspections are new to the project.

This will help the team members participating in the inspection to be familiar with the document being presented for revisions. It will also help the reviewers be more productive by giving more feedback to the author since they have an understanding of the product that is being revised.

Preparation - Preparation is the "key stage in which members of the inspection team prepare individually for the inspection by reviewing and finding potential defects in the product being inspected" [NASA, 1993]. During this stage, everyone participating in the review should prepare himself or herself thoroughly.

The reviewers should use the preparation log (see Appendix 2) that was distributed by the moderator to write down all the defects found, as well as the time spent. After filling the preparation logs, the reviewers will send the logs to the moderator. While reviewing the documents, reviewers should look at the document and try to find general problems that are related with his or her specific area of expertise. For instance, Designers should look at the documents from the design side while quality assurance engineers should look at the documents from the quality side. Reviewers should also check the documents "against higher-level work products, standards, and interface documents to assure compliance and correctness" [NASA, 1993].

The moderator will collect all the logs from the reviewers to determine whether the team is adequately prepared. "If the preparation logs indicate that the team is not adequately prepared, the moderator should reschedule the inspection meeting, as a team

not fully prepared will waste time and will not be effective in finding defects” [NASA, 1993].

If the moderator finds that the logs show good preparation from the inspectors, he will decide to continue with the Inspection. First he will check, though, for “trouble spots that may need extra attention during the Inspection, common defects that can be categorized quickly, and areas of major concern” [NASA, 1993]. By doing this, the Inspection meeting will be more productive since good preparation from all the participants took place.

Meeting - This is the phase of the Inspection review "where team members as a group review the product to find, categorize, and record but not resolve defects" [NASA, 1993]. Defects that need extra time for resolution will be written as action items to be resolved by the author.

To begin with, the moderator will call on the meeting and verify that the Documentation Specialist is ready to take minutes. The moderator will also make sure that all the presentable material is on-line so that everyone would be able to access it. Especially since the DiSEL project was conducted with two sets of students who were geographically distributed. The moderator will then introduce all the participants of the review, "briefly describing their roles, and restating the purpose of the inspection and product" [NASA, 1993]. The moderator will then ask the reader to present the document.

The reader is the person who "provides a logical reading and interpretation of the product" [NASA, 1993]. It is recommended that the reader should not be the author. Instead, it should be a member of the team who is involved in the next phase of the project. For instance, if the Requirements Document is being presented for revision, then the Designer should be the person who acts as the reader.

While the reader is presenting the document, the reviewers will be allowed to "interrupt the reader...when an item containing a possible defect is read" [NASA, 1993]. The reviewer should keep the comments short and precise to the actual topic being presented. If the moderator notices that too much time is being spent in trying to come to a resolution to the problem, he will declare the issue unresolved and will ask that the problem be written as an action item to be revised at a later time. This will allow the meeting to stay within the schedule of the agenda developed by the moderator. After the discussion by the reviewer has ended, the reading is resumed.

The Documentation Specialist, as already stated, will be the person responsible for writing the minutes of the review, but more importantly, he will be the person who will write the unresolved issues into an action item sheet (see Appendix 4). "The recorder will itemize each agreed upon defect by recording its location, a brief description, its classification, and the inspector who found it" [NASA, 1993].

At the end of the meeting, the moderator will ask the Documentation Specialist to read back all the open issues that were written into the action item list. The people

involved in the review will state if they are actual problems and will also classify them by severity (major or minor). Based on the amount of open items, the moderator will determine if a third hour is needed to finalize the review. "If a third hour is needed, the action items are assigned to individual inspectors at this time" [NASA, 1993].

Also, the Documentation Specialist gives a copy of the action items to the author of the document for reference during the rework phase. The additional sheet that contains minor defects is also given to the author.

Third Hour - This is an "additional time, apart from the inspection meeting, that can be used for discussion, possible solutions, or closure of open issues raised in the inspection meeting" [NASA, 1993]. Basically, if the Inspection was not finish in the time schedule, this additional time will be used. Also, this time should be used if the author of the document would like to discuss corrections that he/she has made to his/her work.

The third hour should not be schedule right after the Inspection. Instead, it should be schedule a day or two after the Inspection. Everyone who participated in the Inspection is not required to be present during the third hour.

Rework – This is the “stage when the author corrects defects found during the inspection" [NASA, 1993]. Priority should be given to the problems that were assigned as Major defects. "Minor defects should be resolved if cost and schedule permit" [NASA, 1993].

The moderator will be responsible for making sure that the author addresses the errors found during the revisions.

Follow-up – This session of the Inspection is seen as a "short meeting between the moderator and author to determine if defects found during the inspection meeting have been corrected and to ensure that no additional defects has been introduced" [NASA, 1993]. If additional help is needed to verify this process, the moderator may include additional reviewers.

"If all the major defect have been corrected, all open issues have been resolved, and the product has satisfied the exit criteria, then the moderator 'Passes' the product by recording the completing of the inspection on the Inspection Summary Report" [NASA, 1993]. If the moderator finds that the necessary corrections have not been made, he will ask the author to do more work and correct the errors that still need to be fixed.

3.2.5 Example of Technical Reviews Performed in DiSEL

This section shows concrete examples of two Technical Reviews, a Walkthrough and an Audit, which were performed by the Quality Assurance Engineer (QAE) on the DiSEL project. The audit was performed on the Design Document v2.0, while the Walkthrough was performed on the Project Manager's Plan. No examples of Inspections are given because there were no Inspections conducted during the development process.

3.2.5.1 Audit of Design Document v2.0

The Audit to the Design Document v2.0 was schedule for February 18, 1999 during laboratory hours (see Section 3.2.2). The purpose of the review was to verify that the document was written in accordance with the *IEEE Software Engineering Standards*, which were the standards that were follow by the DiSEL team. This was the first revision of the Design Document. Two more Audits were schedule in the remaining of the project. The three revisions to the Design Document are in relation to the three cycles that DiSEL underwent while using the Spiral Model.

In order to review the document and have feedback to the Designers, the Quality Assurance Engineer (QAE) asked for the document to be submitted on February 12, 1999. This will give the QAE six days to revise the document. The member of the team responsible for the revision of this document was Gregoire Landel. Gregoire was the Analyst for the team. After writing the requirements document, he was asked to help the QAE since quality assurance was his second role. Appendix 7 shows the notification of an audit sheet that was sent to Gregoire by the QAE, Gregorio Cruz. The notification asked Gregoire to write down all the items that needed clarification on a preparation log. All of the comments written by Gregoire can be found in Appendix 8.

Since there was delay in revising the document (see Appendix 8), the QAE gave the result to the Designer personally instead of giving them during the time allocated which was on February 18, 1999.

3.2.5.2 Walkthrough on Project Manager's Plan

The Walkthrough for the project manager plan was schedule for February 9, 1999. The document was submitted to be revised so that it could be frozen and serve as the basis for the whole project. Walkthrough forms were sent to the participants of the review:

- Moderator: Gregorio Cruz (Quality Assurance Engineer)
- Author: Christian Manasseh (Project Manager)
- Reviewer1: Gregoire Landel (Analyst)
- Reviewer2: Padmanabha Vedam (Designer)
- Reviewer3: Alberto Mireles (Quality Assurance Engineer)
- Recorder: Ricardo Acosta (Documentation Specialist)

Appendix 9 shows one notification form that was sent to Alberto Mireles. The other notification forms that were sent to the other participants contained the same information that was put on Alberto's notification form. The notification sent to the recorder asked him to use the preparation log (see Appendix 2) during the review to write down all the action items.

The Walkthrough had been schedule so that only the reviewers could give feedback to Project Manager. The remaining members of the team could raise their concerns at the end of the review. While conducting the Walkthrough, the QAE found that everyone could participate in the review by giving feedback to the author of the document, as he was presenting his work. The approach of having everyone participate in the Walkthrough was found after the QAE saw that the people who were not chosen to be reviewers were as equally prepared as the reviewers.

Participation from all the team allowed the Project Manager to get more feedback from all the members as he was introducing different sections of this document. All the feedback was written into the preparation log and was given to the author of the document. The status of the document was to be determined after the Project Manager made corrections to his document. After correcting all the issues raised during the revision, the QAE verified them and if they were implemented correctly, the document would be frozen and serve as the base for the whole project.

The following comments were gather from the member of the team after the conducting the Walkthrough.

“Walkthrough is the best tool used in QA, in terms of the involvement of others in the group. I was able to fully understand the work being done by the PM, where I was left with many questions before when I simply read the document.”

“Since we all do the same, and therefore we admit, all documents are full of a lot of baloney as well as the real stuff. Although we write these to inflate the reports, we do not tell these in class, and as a result the person talking about his report in the Walkthrough gives a nice summary as well as cutting the useless parts off.”

“Since Walkthroughs are interactive, all questions are answered.”

“Because the writer of the report comments about what he wrote, new information is revealed at times, which helps to understand the situation better.”

“The Walkthrough has all the members of the group present, and since all roles are pretty much interlaced, there's always bound to be some discussions going into the domains of other roles. This ends up in the discussion of some previously unresolved points, thus contributing to the overall progress of the project.”

3.3 Conclusion

This chapter showed how the learning of quality assurance was used to manage the quality of the DiSEL project. It also showed the problems that occurred while trying to perform all quality tasks. Examples of how Technical Reviews were performed are also illustrated in this chapter.

Section 3.1 shows the integration of the roles and outlines the language, and academic barriers that the team encountered while participating in this distributed project.

Section 3.2 is the section that talks about the Quality Assurance Plan (QAP) which shows the work performed by the Quality Assurance Engineer (QAE). This section starts by stating the importance of the QAP and why it should be developed at the beginning of the project. The section follows by going over the management of the project's quality and how the primary roles (Software Configuration Manager, Test Engineer, and Documentation Specialist) were instrumental to accomplish this task. A schedule of events that were performed by the QAE with the help of the primary roles is shown in Section 3.2.2.

Section 3.2.3 states all the different documentation that was produced during the development of the project and how the QAE would verify the quality of this documents. To check the documents for quality, the QAE used the four most used software technical reviews (Peer Reviews, Audits, Walkthroughs, and Inspection). Section 3.2.4 states a

definition of these reviews and also shows how each of these software technical reviews were performed during the DiSEL project.

Section 3.2.5 gave examples of how the QAE conducted an Audit to the v2.0 of the Design Document and a Walkthrough to the Project Manager's Plan.

Many challenges arose as the QAE tried to implement the QAP. Out of these mistakes, the QAE came up with suggestions of how to improve a distributed project such as DiSEL. These recommendations are stated in Chapter 4.

Chapter Four

4 Recommendations

This chapter contains recommendations on how to structure a distributed class from the view of a Quality Assurance Engineer (QAE). It also has recommendations on how the QAE should apply quality assurance to distributed software engineering projects by making use of the Internet.

4.1 Class Structure for Distributed Software Projects

Distributed software classes such as DiSEL (Distributed Software Engineering Laboratory) need a lot of preparation from the professors. Professors must plan the structure of the class in advance so that the students are able to produce the work that is asked. Trying to produce work while learning made the students uncomfortable since they were not able to perform their software role as demanded. For instance, the Requirements Document, which was ask to be developed early in the project, was never completed because the Analyst was trying to learn about software requirements first. If the Analyst had a better understanding of how to collect and write the requirements of a system, he would have produced the document on time.

Careful preparation should be given to a plan that will bring all the students, both from the Massachusetts Institute of Technology (MIT) and Centro de Investigacion Cientifica y de Educacion Superior de Ensenada (CICESE), to a same level of understanding about software engineering. This year, students from one university came into the project more prepared than the other students since they had already taken Software Engineering courses.

Students who came into the project knowing little about Software Engineering were frustrated because the other students were already prepared to start working in a distributed software project. Students who had taken courses in software engineering were also frustrated because they could not make progress within the project since not all students were ready to build a system that required knowledge of Software Engineering. While one group of students was trying to develop a system, the other group was learning how to develop software systems in a distributed project.

Therefore, an equal understanding of software engineering from the team needs to be established before the team can start working on a distributed project such as DiSEL. If there is an equal knowledge of Software Engineering, the team will be able to progress by becoming more productive. Members of the team would produce their work on time, allowing the other members to follow a schedule they had developed at the beginning of the project. For instance, if the documents were submitted on time for revision, the Quality Assurance Engineer will be able to assure the quality of the member's work as schedule.

Table 8 below shows a schedule of events that could be used to teach the software engineering process, the roles played and how they are played to all participating students. It should be noted that the schedules of events that are shown below are taken from the view of the QAE. The schedule used is based on this year's MIT schedule where classes were conducted every Tuesday and Thursday. After the sessions have been completed, the students can start working on the project. At that point, the students should be more familiar about Software Engineering.

Table 8. OUTLINE FOR A NEW DISTRIBUTED COURSE

COURSE OUTLINE						
SESSION	DAY	DATE	TOPIC	READINGS	ASSIGNMENTS	
					OUT	DUE
1	Thursday	Sept. 10	Introduction			
2	Tuesday	Sept. 15	Software Engineering	1	AS1	
3	Thursday	Sept. 17	Project Management	2	AS2	AS1
4	Tuesday	Sept. 22	Software Configuration Manager	3	AS3	AS2
5	Thursday	Sept. 24	Quality Assurance Engineer	4	AS4	AS3
6	Tuesday	Sept. 29	Documentation Specialist	5	AS5	AS4
7	Thursday	Oct. 1	Analyst	6	AS6	AS5
8	Tuesday	Oct. 6	Design I	7	AS7	AS6
9	Thursday	Oct. 8	Programmer	8	AS9	AS7
10	Tuesday	Oct 13	Test Engineer	9	AS9	AS8
11	Thursday	Oct 15	Technical Reviews	10	AS10	AS9
12	Tuesday	Oct 20	Conclusion			AS10

As already stated before, the course outline in Table 8 was created from the view of the QAE. The first lecture should be a presentation that covers everything about software engineering. After that session, lectures about the different roles played in software engineering should be presented. The QAE finds that the management roles, Project Management, Software Configuration Management, Quality Assurance Engineer, and

Documentation Specialist should be presented first and then the roles of the Analyst, Designer, Programmer and Test Engineer. At the end, a session for Technical Reviews should be presented so that all students will have an understanding of the different technical review processes.

Assignments are given to the students after each topic is presented, starting with Software Engineering. The assignments would consist of homework that test student's understanding of the class that was presented. The homework should not be too long, they should be structured to test the full understanding of the student over the topic that was introduced. The students will bring their assignment on the following session and for the first 30 minutes the professors will answer questions that the students have about the assignment in order to clarify any concerns or misunderstandings.

After the presentation of all the sessions, the students should have a better understanding of software engineering. Students will be able to apply their learning into the project and they will be able to execute the work necessary to produce a system that is asked for any client or a market.

4.2 Application of Quality Assurance to distributed projects

This section focuses on how to make the job of the Quality Assurance Engineer (QAE) better when he/she is working in a distributed project. Since the team is composed of at least two groups who are geographically distributed, the work that is

conducted by the QAE must be accessible by all team members at all times. To make this happen, everything that the QAE does should be put into a Web site.

Before deciding what is going to be put on the Web site, a clarification of the QAE's job seems necessary. The job of the QAE is to provide quality assurance to the process of distributed Software Engineering projects, mainly to all the documents that are being written by different members of the team. In order to assure the quality of the documents, the QAE conducts technical reviews.

A Technical Review needs a lot of preparation and involves different members of the team (moderator, author, reader, reviewer, and documentation specialist). As it was presented in Chapter 3, the QAE used different forms (see Appendixes) that aided in the preparation of all Technical Reviews (Peer Reviews, Audits, Walkthroughs, and Inspections).

The recommendation would be to put all the forms that were used by the QAE in DiSEL on the Web so that the QAE will not have to create the forms every time a Technical Review takes place. After a form is filled out on-line, e-mail would be sent to the project's e-mail list so that everyone will be aware of the review. Then the participants chosen for the review (moderator, author, reader, reviewer, documentation specialist) will know of the review and they can start preparing for it. After a form is filled out, it will be saved on a database so those members of the team can access it at a later time.

For the creation of the Quality Assurance Engineer's Web Page, FrontPage from Microsoft can be used. FrontPage allows you to create the forms without knowing how to program in HTML. FrontPage also allows you to save and call any of these forms into a database without knowing SQL programming. For the database connection, FrontPage uses Microsoft Access, which is a personal database that comes along with Microsoft Office. Access control for the forms can be granted from the WinNT Operating System.

The following sections show how the different forms used by the QAE will look over the Internet. Description of the contents of the snapshot will also be provided.

Audit Notification – Figure 10 shows how the audit notification form could be structured using HTML code so that it can be viewed over the Internet. The form uses drop down menus for selecting to whom the notification will be sent and from which role the notification came from. Text boxes are used to write down the dates, time and the place where the actual Audit will take place. Scrollable text boxes are used for writing down the subject of the review and the comments that the person who is writing the notification feels are important for the Audit. Only the moderator of the review should have access to this form. Management roles such as Project Manager, and Configuration Managers should also have access to this form.

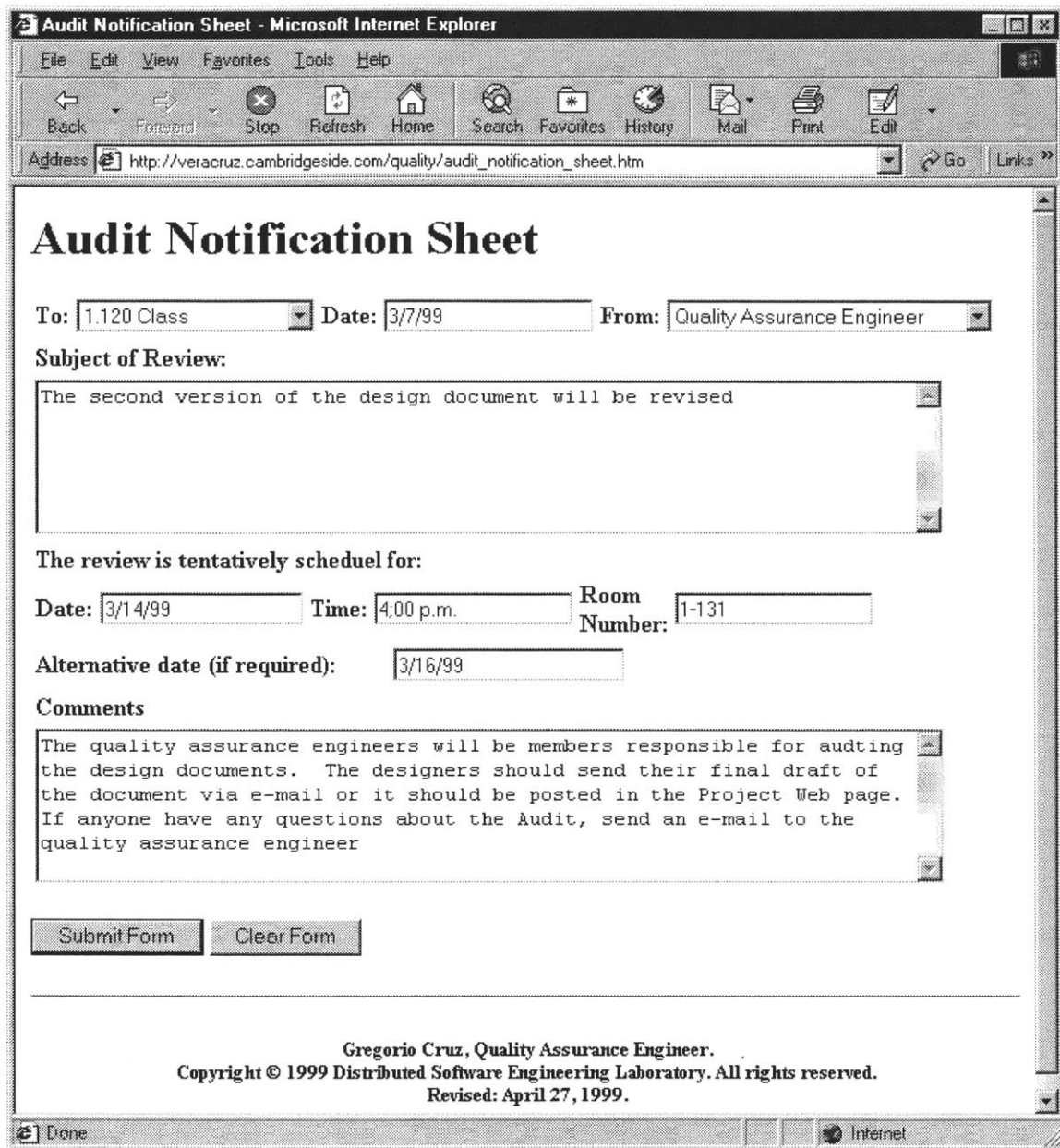


Figure 10. AUDIT NOTIFICATION SHEET

Walkthrough Announcement – Figure 11 shows how the Walkthrough Announcement would look over the Internet. The walkthrough differs from the audit in that the walkthrough announcement has more data that is necessary for the conducting the review. Similar to the audit, only management roles should have access to this form.

Walkthrough Announcement - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print Edit

Address http://veracruz.cambridgeside.com/quality/walkthrough_announcement.htm Go Links

Walkthrough Announcement

To: Date: From:

Subject of Review:

A Walkthrough will be held on:

At: In:

Reviewers, please review the material that will be posted ont he web before the walkthrough and prepare your comments. If you cannot attend the walkthrough or will not be able to review the material before time, contact the Author and the Quality Assurance Engineering so thte walkthrough can be reschedule or another memeber of the team can acat as a reviewer.

Current Status of the Material:

The objective of the material is to:

Moderator:

Recorder:

Reviewer 1:

Reviewer 2:

Done Internet

Figure 11. WALKTHROUGH ANNOUNCEMENT SHEET

Inspection Notification Sheet – Figure 12 shows the Inspection notification sheet. This form is similar to the Audit and Walkthrough notification. A difference of this review to

the others is that it asks for more reviewers to be involved in the actual review. Similar to the other notification forms, only management roles should have access to this form.

Inspection Notification Sheet

To: 1.120 Class Date: 2/23/99 From: Quality Assurance Engineer

Subject of Review: We will review the Project Manager's Plan.

Comments: Everyone who will be involved in the revision should try it's best to prepare for the revision. The members of the team who will be participating are at the bottom. If you need extra assistance, contact the Quality Assurance Engineer

The review is tentatively scheduel for:

Date: 3/1/99 Time: 4:00 p.m. Room Number: 1-131

Alternative date (if required): 3/3/99

Personnel assigned to the SQA Review Team for this review are:

Person:	Role:
Gregorio Cruz	Moderator
Christian Manasseh	Author
Jaime Solari	Reader
Gregoire Landel	Reviewer
Joon Hoor	Reviewer

All of the participants listed on the left should receive all the documentation that will be revised during the technical review. The moderator should be the person responsible for making sure that the documents are delivered.

Submit Form Clear Form

Gregorio Cruz, Quality Assurance Engineer.
 Copyright © 1997 Distributed Software Engineering Laboratory. All rights reserved.
 Revised: April 27, 1999.

Figure 12. INSPECTION NOTIFICATION SHEET

Preparation Log – Figure 13 shows the preparation log that is used by all the reviewers who are participate in the different technical reviews. The form asks for the reviewer to write down all the defects found along with the time spent. A scrollable text box is used

so that the reviewer can write down a full description of the defects found. Every member of the team should have access to this form.

Preparation Log

Name of Inspector: Date:

Subject of Review:

Number:	Defect Found:	Time Spent:
<input type="text" value="1"/>	Sec 1.1 Visualize movement + social feedback (see anotaded doc.)	<input type="text" value="5 minutes"/>
<input type="text" value="2"/>	Sec 2 (paragraph 4) Please not distinct functions of Awareness and VRML	<input type="text" value="5 minutes"/>
<input type="text" value="3"/>	Sec 2 (paragraph 5) Please make reference to figure	<input type="text" value="5 minutes"/>
<input type="text" value="4"/>	Figure 1. Social feedback is not included	<input type="text" value="5 minutes"/>
<input type="text" value="5"/>	Page 8 (bottom) unclear reference to "first environment.	<input type="text" value="5 minutes"/>

Gregorio Cruz, Quality Assurance Engineer.
Copyright © 1999 Distributed Software Engineering Laboratory. All rights reserved.
Revised: April 28, 1999.

Figure 13. REVIEWER'S PREPARATION LOG

Action Item Sheet – Figure 14 shows the form that is used by the Documentation Specialist to write down all the unresolved errors that were found during a review. Management, along with the Documentation Specialist should have access to this form.

Action Item Sheet

Material prepared by: Date:

Subject of Review:

Person:	Role:
<input type="text" value="Gregorio Cruz"/>	<input type="text" value="Moderator"/>
<input type="text" value="Gregorio Cruz"/>	<input type="text" value="Author"/>
<input type="text" value="Kiran Choudary"/>	<input type="text" value="Reviewer"/>
<input type="text" value="Vedam Padmanabha"/>	<input type="text" value="Reviewer"/>
<input type="text" value="Ricardo Acosta"/>	<input type="text" value="Documentation Specialist"/>

Current Status: New Status:

Number:	Description of Action Item:	Resolution:
<input type="text" value="1"/>	<input type="text" value="Section 1 needs more work."/>	<input type="text"/>
<input type="text" value="2"/>	<input type="text" value="Figure 1 needs to be mentioned in the body."/>	<input type="text"/>
<input type="text" value="3"/>	<input type="text" value="You need to put some relationships between the differen roles."/>	<input type="text"/>
<input type="text" value="4"/>	<input type="text" value="Explain more the process that will be followed within the project."/>	<input type="text"/>
<input type="text" value="5"/>	<input type="text" value="Change the event of 3/12/99 to 3/14/99"/>	<input type="text"/>

Figure 14. DOCUMENTATION SPECIALIST ACTION ITEM SHEET

Inspection Checklist – Figure 15 shows the form that is used by Quality Assurance Engineer (QAE) to help him create a report of the whole Inspection review. With this form, the QAE will be able to keep the whole team inform about the Inspection Review. Only the management roles should have access to this form.

Inspection Checklist

Moderator's Name: Date:

Subject of Review:

We are reviewing the Manual that will be sent to the users. It is important that the maual is carefully check for all possible errors.

Item:	Comments:	Status:
Planning:	The people participating have been chosen and notification of review has been sent	Completed
Overview:	There was no overview schedule since the people participating were familiar with the	Completed
Preparation:	The inspectors should be revising the manual carefully	Not-Complete
Meeting:		Not-Complete
Third Hour		Not-Complete
Rework		Not-Complete
Follow-up		Not-Complete

Figure 15. MODERATOR'S CHECKLIST FOR INSPECTIONS

4.3 Conclusion

This chapter presented recommendations for a distributed software engineering class and an application that would enable the Quality Assurance Engineer (QAE) to perform his/her work more systematically, faster and easier.

Section 4.1 states that in order for the students participating in geographically distributed software engineering class, they need to learn software engineering first. By learning all about software engineering, students will be able to perform better within the project.

Section 4.2 shows how the QAE how he can incorporate the forms used in technical reviews into the Web. This will allow all members of the team to have access from anywhere. Print Screens are given in this chapter to show how the forms would look once they are put into a Web Page.

Chapter Five

5 Conclusion

The DiSEL project allowed students from MIT and CICESE to learn software engineering and to work in distributed software development projects. The class was schedule two times a week, one for lecturing and one for laboratory. Students participating were assigned a primary and a secondary role. The student would always perform the primary role while being on-call for their secondary role.

The project asked the students to develop a software system that would enable better collaboration between geographically distributed teams. The students acted like a start-up company and were asked to participate in the 50K Entrepreneurship competition at MIT. This allowed them to learn how to sell an idea.

While going through the entire development software cycle, the Quality Assurance Engineer (QAE) faced many challenges that limited his performance. Having class members with different levels of expertise in software engineering was the first problems encountered. Some students had worked together before while others had never seen each other. This mix created a situation in which the students were unable to

collaborate together at the beginning of the project since they had different ideas about the job that they were asked to perform. Certain students were frustrated because, although they were eager to produce their work, they were still learning about the software development process. One solution to this problem may be to take the first two months of the project to teach about the software engineering process and the roles engineers play during the development cycles. This will prepare the students for the development of a software system in a geographically distributed atmosphere.

Not having a common repository to share all the project's data was another problem that the QAE faced. The QAE was producing different forms (see Appendixes) and carrying out Technical Reviews that needed to be stored in a place where the team could have access to it. Developing web pages that will enable the QAE to have the team up to date on quality measures would be a solution to this problem.

To create the web pages, the QAE could take advantages of FrontPage from Microsoft. FrontPage allows a user to develop web pages and forms needed for technical reviews without knowing HTML programming. FrontPage also has the capabilities of sending the forms created through e-mail once they have been completed and storing them in a database. Sending e-mail notifications will kept the team informed while having the forms in a database will allow members of the team to revisit the QAE's work (forms or technical reviews) at any time.

In all, having a better-prepared team, and a place where the team members can share their work, will increase the success and quality of a distributed software development project. Additionally, having a well-prepared classroom, with all the technology (e.g., video cameras, projectors, and audio) available, will enable the collaboration between distributed teams to increase as communication tools will be easier to access.

Bibliography

- [1] Boehm, B., *A Spiral Model of Software Development and Enhancement*, ACM SIGSOFT Software Engineering Notes, Vol. 11, No. 4, 1986.
- [2] Evans, M. W., Marciniak, J., *Software Quality Assurance and Management*, John New York, Wiley & Sons, 1986.
- [3] Gibson, C.F., *Lifeline Systems, Inc. – The Caresystem Project*, MIT Sloan School of Management, MIT, 1999.
- [4] IEEE, *Software Engineering Standards*, IEEE Press, 1997.
- [5] Hussein, Karim., *Computer supported interaction in distributed design teams*, Ph.D. Thesis, Civil & Environmental Engineering, MIT, 1998.
- [6] McCall, J. A., Richards, P.K., Walters, G.F., *Factors in Software Quality*, Vols I-III, Rome Air Development Centre, 1977.
- [7] MIT 50K, *MIT \$50K Entrepreneurship Competition*, MIT, <http://50k.mit.edu/> Last Visited: May 13, 1998.
- [8] Mizumo, Y., *Software Quality Improvement*, IEEE computer, March 1983, pp. 66-72.
- [9] Nenz, H., *Management der Software-Qualitatssicherung*, Softwaretechnik-Trends, Vol. 3-1, 1983.
- [10] NASA, *Software Formal Inspections Guidebook*, Office of Safety and Mission Assurance at NASA, Washington D.C., August 1993.
- [11] Pressman, R. S., *Software Engineering: A Practitioner's Approach*. McGraw-Hill, New York, 1997.
- [12] Vincent, J., Waters, A., Sinclair, J., *Software Quality Assurance*, Prentice Hall, New York, 1988.
- [13] Wallace, Dolores R., Wendy W. Peng, and Laura M. Ippolito, *Software Quality Assurance: Documentation and Reviews*, NISTIR 4909, National Institute of Standards and Technology, Gaithersburg, MD 20899, September, 1992.
- [14] Wallmuller, Ernest., *Software Quality Assurance: A practical approach*, Prentice Hall, Englewood Cliffs, NJ, 1994.

- [15] Walker, M., *Auditing Software Development Projects: A Control Methodology*, *Proceeding*, COMPCON Spring, Silver Spring, MD: IEEE Computer Society Press, 1979.
- [16] Yang, Bob., *Managing a Distributed Software Engineering Team*, M.Eng Thesis, Electrical Engineering and Computer Science, MIT, May 1998.

Appendix 3: Walkthrough Announcement

From: _____

Date: _____

To: _____

Subject of Review: _____

A walkthrough will be held on _____

at _____ in _____

Please review the material that will be posted on the web before the walkthrough and prepare your comments. If you cannot attend the walkthrough or will not be able to review the material before that time, contact the author and Quality Assurance Engineer so the walkthrough can be reschedule or another member of the team can act as a reviewer.

Current status of the material: _____

The objective of the material is to: _____

Moderator: _____

Recorder: _____

Material: _____

Appendix 5: Inspection Notification Sheet

To: _____

From: _____ Date: _____

Subject of Review: _____

Comments: _____

A(n) _____ Review has been schedule.

The review is tentatively schedule for:

Date: _____ Time: _____ Room Number: _____

Alternative Date (if required): _____

Please send a copy of the phase/review documentation to: _____

Personnel assigned to the SQA Review Team for this review are:

Person	Role
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Appendix 6: Inspection Checklist

Moderator's Name: _____

Date: _____

Subject of Review: _____

ITEM	COMMENTS	STATUS
Planning		
Overview		
Preparation		
Inspection Meeting		
Third Hour		
Rework		
follow-up		

Appendix 7: Audit Notification to Revise Design Doc. v2.0

To: Gregoire Landel

From: Gregorio Cruz Date: Feb 12, 1999

Subject of Review: Review the Design Document to make sure that the standards for Writing a design document were followed.

The review is tentatively schedule for:

Date: Feb 18, 1999 Time: 4:00 p.m. Room Number: 1-131

Alternative Date (if required): _____

Please send a copy of the phase/review documentation to: Documentation Specialist, Designers and Quality Assurance Engineer.

Comments: Gregoire, please verify that the design document is incorporating the Requirements specified in the Requirements Document. Also, verify that The document is satisfying the IEEE Software Engineering Standards. Finally, check for grammatical errors. Try to spent at least 3 to 4 hours On this (the whole process). E-Mail me if you have any questions. Attach Is an action item list for you to write the items that need clarification.

Appendix 8: Preparation Log to revise Design Doc. v2.0

Name of Inspector: Gregoire Landel

Date: 2/21/99 (Reading done 2/14/99)

Subject of Review: Software design document v2.0 (Feb 12, 1999) for VRML team

Total time spent (reading + writing : 1hr 15 min)

Number	Defect Found	Time Spent
1	Section 1.1: Visualize movement + social feedback. See annotated doc.	
2	Section 2 (3 rd paragraph): 120 deg. vision	
3	Section 2 (4 th paragraph): Please not distinct functions of AWARENESS of VRML environment.	
4	Section 2 (5 th paragraph): Please make references to figure.	
5	Figure 1: Social feedback is not included.	
6	Page 8 (bottom): Unclear reference to "first environment"	
7	Section 3.3 (Page 9): Requirements for head movement and of avatar is not met.	
8	Section 3.3 (Page 9, bottom): Consider including several areas for avatars.	
9	Minor editorial suggestions throughout the document.	
	GOOD DOCUMENT!!!!	

Appendix 9: Walkthrough Announcement of PM Plan

From: Gregorio Cruz (Quality Assurance Engineer)

Date: February 3, 1999

To: Alberto Mireles (Quality Assurance Engineer)

You will be acting as the reviewer in the Walkthrough of the Project Manager's
Plan. If you have any questions, please let me know by e-mail or in person.

Subject of Review: Freeze the Project Manager's Plan

A walkthrough will be held on February 9, 1999

at 4:00 P.M. (Eastern) in 1-131 (Design Studio of the Future)

Please review the material that will be posted on the web before the walkthrough and prepare your comments. If you cannot attend the walkthrough or will not be able to review the material before that time, contact the author and Quality Assurance Engineer so the walkthrough can be reschedule or another member of the team can act as a reviewer.

Current status of the material: Not Frozen (needs revision)

The objective of the material is to: Review the document to be sure that it is written

In accordance with the IEEE Soft. Eng. Standards.

Moderator: Gregorio Cruz

Recorder: Ricardo Acosta

Material: The material to be reviewed will be the Project Manager's Plan. The PM

Would send an e-mail very soon letting everyone know where this doc. can

Be found so you can start reviewing it. As for the IEEE standards, I'll try to

Get a copy for every as soon as possible.