

TEAM4OIES

Team Project Report

Version 3.0

Signature Block

COSC 4351	Name	Signature	Date
SE Javier Rivera	Javier Rivera	[Signature]	3-9-15
SE Linh Tong	Linh Tong	[Signature]	3-9-15
SE Paul Miller	Paul Miller	[Signature]	3-9-15
SE Team Leader	[Signature]		
SE SQA			
SE SQA			
SE Team Leader			
SE SQA			
SE SQA			

Contents

SIGNATURE BLOCK	1
1. CHAPTER I: INTRODUCTION TO TEAM4OIES	4
1.1 INTRODUCTION	4
1.2 RESEARCH METHODOLOGY	4
1.3 REPORT ORGANIZATION	4
2. CHAPTER II: REQUIREMENTS	4
2.1 PROBLEM STATEMENT	4
2.1.1 Historical aspect	4
2.1.2 Economical Aspect	5
2.1.3 Maintenance Aspect	5
2.2 EXPANDING THE BORDERS	5
2.3 EXPANDING VS. REPLACING THE SYSTEM	6
2.4 CONCLUDING REMARKS	6
3. CHAPTER III: EDUCATING THE USER	6
3.1 THE SOFTWARE PROCESS	6
3.2 PROBLEMS WITH SOFTWARE COMPLEXITY	7
3.3 PROBLEMS WITH SOFTWARE CONFORMITY	7
3.4 PROBLEMS WITH SOFTWARE CHANGEABILITY	7
3.5 PROBLEMS WITH SOFTWARE INVISIBILITY	7
3.6 CONCLUDING REMARKS	7
4. CHAPTER IV: CHOOSING THE RIGHT TEAM	7
4.1 PROBLEM STATEMENT	8
4.2 ASSEMBLING THE TEAM	8
4.2.1 The Democratic Approach	8
4.2.2 The Classical Approach (Chief Programmer)	8
4.2.3 The Modern Approach (Chief Programmer & Business Manager)	9
4.3 RECOMMENDATION FOR TEAM4OIES	9
4.4 CONCLUDING REMARKS	9
5. CHAPTER V: CHOOSING THE RIGHT MODEL	9
5.1 PROBLEM STATEMENT	9
5.2 COMPARING MODELS	10
5.2.1 Waterfall Model	10
5.2.2 Incremental Model	10
5.2.3 Extreme Programming	10
5.2.4 Synchronize and Stabilize	11
5.2.5 Spiral Model	11
5.2.6 Object-Oriented Model	11
5.3 RECOMMENDATION FOR TEAM4OIES	12
5.4 CONCLUDING REMARKS	12
6. CHAPTER VI: ANALYSIS AND DEVELOPMENT METHODS	12
6.1 PROBLEM STATEMENT	12
6.2 CONVENTIONAL ENGINEERING METHODS	12
6.1.1 Stepwise Refinement	12
6.1.2 Cost Benefit Analysis	12
6.1.3 Software Metrics	13
6.1.4 CASE	13

6.1.5	Software Versions	13
6.1.6	Configuration Management	13
6.1.7	Problem Statement	13
6.1.8	Other Applicable Engineering Tools	13
6.3	RECOMMENDATION FOR TEAM4OIES	13
6.4	CONCLUDING REMARKS	13
7.	CHAPTER VII: TESTING	14
A.	PROBLEM STATEMENT	14
B.	QUALITY ASSURANCE	14
	NON-EXECUTION-BASED TESTING	14
	EXECUTION-BASED TESTING	14
C.	RECOMMENDATION FOR TEAM4OIES	15
D.	CONCLUDING REMARKS	15
8.	CHAPTER VIII: DEVELOPING TEAM4OIES PAGE MASTER AND HOME PAGE	15
8.1	PROBLEM STATEMENT	15
8.2	TEAM4OIES PAGE MASTER AND HOME PAGE	15
8.3	CONCLUDING REMARKS	15
	DOCUMENT CONTROL	16

1. CHAPTER I: INTRODUCTION TO TEAM4OIES

1.1 INTRODUCTION

TEAM4OIES is an international online database system to be used to store, manage, visualize, and query EVAR (EndoVascular Aneurysm Repair) data in the form of medical images).

TEAM4OIES will allow Surgeons, Technicians, and Computational Scientists to collaborate in collecting data on the patients that have had this surgery. The goal will be to have a large patient database with metrics on the distention of the aorta before and years after the endovascular aneurysm repair.

1.2 RESEARCH METHODOLOGY

TEAM4OIES will be the first of its kind. Doctors will be able to upload anonymous patient information. Technicians will be able to upload patient data. Computational Scientist will be able to download, run metrics on the data, and upload the data. Although this type of program will be the first of its kind, the technology to accomplish this is accessible.

1.3 REPORT ORGANIZATION

We have organized the report with the user in mind. Chapter II will offer a better understanding to the on the Historical, Economical, and Maintenance Aspect that TEAM4OIES have taken into account and addressed in TEAM4OIES.

Chapter III will offer a better understanding to the layperson on what variables will go into creating a software product. Anyone can hire a few programmers to "hack" a product together, but it takes meticulous design to create a great product that will be future proof.

W.V.V. 7 -10 Red Feedback

2. CHAPTER II: REQUIREMENTS

2.1 PROBLEM STATEMENT

2.1.1 Historical aspect

The majority of software products are not delivered on time, exceed budget limits, lack functions that were promised or have several bugs and faults. In a study performed in 2006, approximately 46% of projects fit into this category. Although these projects were completed and installed, they faced one of the problems listed. To prevent our project from experiencing any of these problems, TEAM4OIES will need to take several precautions.

To ensure that TEAM4OIES's project would be delivered on time, deadlines have been set for our deliverables and work has been delegated equally among the members of the

team. Having the work split evenly among members assures a higher probability of having the work completed on time. TEAM4OIES also wants to make sure our project will function the way the client needs it to. During the requirements and analysis phase, TEAM4OIES will determine what the client needs and will research ways to satisfy these requirements. For our project to be bug-free as possible, TEAM4OIES will rely on our SQA's to continuously test and check our deliverables to make sure that everything is functioning properly without errors. Because this is a class project, TEAM4OIES will not need to worry about exceeding the budget.

2.1.2 Economical Aspect

The economic principle of software development dictates that as the software developers, TEAM4OIES must take into consideration the short and long-term impact of our software on our client. The product that TEAM4OIES create needs to work according to our client's specification, and at the same time, it also needs to be easy to maintain. TEAM4OIES would like develop the product using a simple user interface that the client can easily navigate. By doing so, it eliminates the extra cost and time incurred to train the employees on how to use the software. This will also help the client in the long run if they decide to hire new employees that will need training for the software.

2.1.3 Maintenance Aspect

As software developers, TEAM4OIES will need to develop the product using the waterfall model. The waterfall model consists of six different stages. During the first stage, the requirements phase, TEAM4OIES will identify the client's requirements and the concept of the software. Then the analysis phase requires that we produce the specification document and the software project management plan. The specification document explains what the product should do and the software project management plan describes the software development process in detail. The design phase contains two design procedures: the architectural design and the detailed design. The architectural design is where the project will be broken into parts called modules and will then be designed in the detailed design procedure. The implementation phase involves two steps, coding and testing. When our product has been coded, it will then be sent to the client for testing. Once the client is satisfied with the product, this marks the end of the implementation phase and the beginning of postdelivery maintenance. In this phase, we perform corrective maintenance. Any errors or faults that happen after the software is installed will need to be corrected. TEAM4OIES may also need to enhance or update the software by making changes to the specifications and having these changes implemented. The final stage is retirement in which the product is no longer useful to the client and will need to be removed. If our software is deemed inadequate by the client, TEAM4OIES will remove the software.

2.2 EXPANDING THE BORDERS

The postdelivery maintenance stage in the software life cycle is very important. This phase incurs the majority of software costs. Our product serves as a model of the real world, but the real world is constantly shifting. Therefore, TEAM4OIES would expect

that our software product would also change to reflect the real world. Such changes can include a change in sales tax or a change in company policy. For our software project, TEAM4OIES should employ techniques that would help to decrease the overall cost of postdelivery maintenance. The client may need to make changes to the software. For example, the client's company decides to change the formatting of the patient IDs. Therefore, TEAM4OIES will need to code our software so that it will allow for easy changes to be made.

2.3 EXPANDING VS. REPLACING THE SYSTEM

During postdelivery maintenance, TEAM4OIES may need to perform two types of enhancement: perfective maintenance and adaptive maintenance. Perfective maintenance are changes made to the software that will boost its effectiveness where adaptive maintenance is changes that are made when the environment fluctuates. Under perfective maintenance, our client may request for additional functionality. At the moment, only certain users (i.e. surgeons) are allowed to upload EVAR CT scans and download CFD flow simulations. In the future, our client may want other users to have that ability, so as the software developers, TEAM4OIES should be able to implement this additional function for them. Changes in the environment may necessitate adaptive maintenance. Our website should run on Chrome and Firefox browsers. Updates to the browsers may cause our website not to function properly. Therefore, TEAM4OIES need to remain up-to-date and make sure that our software will always be up and running.

2.4 CONCLUDING REMARKS

TEAM4OIES will develop a software product that will meet the requirements and contain the functionalities requested by the client. It will need to be bug-free, delivered on time, and easy to maintain. In order to accomplish this, TEAM4OIES will follow the stages of the software life cycle and incorporate techniques that follow the software development process. TEAM4OIES will try to minimize future maintenance costs and produce a completed software product that will fulfill the client's needs.

3. CHAPTER III: EDUCATING THE USER

3.1 THE SOFTWARE PROCESS

The Unified Process consists of Inception, Elaboration, Construction, and Transition. During the Inception phase, there is emphasis on the business modeling and the requirements of the project. During this phase, TEAM4OIES will discuss risks and specifications of the projects. After the Inception phase, TEAM4OIES will get into the Elaboration phase which is where we really get an idea of how to do our project. TEAM4OIES will also discuss the requirements in more detail along with the risk factors. During this phase, we will build use case diagrams and basic class diagrams. Finally we will get a cost and schedule to develop the project. Next, TEAM4OIES will transition into the Construction Phase where TEAM4OIES will implement and build our software. TEAM4OIES will prioritize the foundation of the project using more in depth MVC UML diagrams. Every iteration of this phase will result in more software changes

to eventually complete the foundation. Finally the last phase is the transition phase where we will turn the project in to the client and receive feedback. If the client wants something to change TEAM4OIES will give him an estimate on how long and how much and get to work to meet specifications. As TEAM4OIES works on this project, we will add compatibility maturity models to assist us in having as few bugs as possible.

3.2 PROBLEMS WITH SOFTWARE COMPLEXITY

Software is the most complex thing built by human kind. Because of this, great care must be taken into building the software. Our group will take great care into designing the software with models and UML to ensure that TEAM4OIES fix most potential bugs in planning as opposed to during development.

3.3 PROBLEMS WITH SOFTWARE CONFORMITY

An important part of software is to work in unison with existing software. As a team, we will carefully plan out the software we use to design the web app/database and make sure that our software meshes well with software already developed by the client.

3.4 PROBLEMS WITH SOFTWARE CHANGEABILITY

A large problem with software changeability is that the software must be constantly updated if the software doesn't adapt, it WILL die. If the software is well designed, there will be pressure from happy customers to constantly improve the product. Despite these things, software is very easy to change and modify. In our group, we hope to make the best software and keep the customer satisfied.

3.5 PROBLEMS WITH SOFTWARE INVISIBILITY

A huge problem with software development is the fact that it is hard to see the product as we are making it and hard to imagine it during development. A good way to counteract invisibility is with UML Models and flowcharts. Even with good flowcharts, it is impossible to get a grasp of the product fully. In our group, we plan to have a very detailed model and if we miss something during planning we will quickly modify our charts and models.

3.6 CONCLUDING REMARKS

This project requires that TEAM4OIES have great models and UML. By planning the project appropriately, we will hopefully avoid the five issues faced above and build a successful database for our client. As long as our group follows the Unified Process and our models, TEAM4OIES are sure to create great software for our client.

4. CHAPTER IV: CHOOSING THE RIGHT TEAM

4.1 PROBLEM STATEMENT

Most products are too large to be completed by a single software professional within the given time constraints. As a result of this, the products are given to a group of developers. These developers form a team.

4.2 ASSEMBLING THE TEAM

4.2.1 The Democratic Approach

The democratic team organization's basic concept is a team of egoless programmers. Every programmer must encourage the other members of the team to find faults in his or her code. If not, ego creeps in and developers see their written code as an extension of themselves. A group of up to ten egoless programmers constitutes a democratic team.

4.2.2 The Classical Approach (Chief Programmer)

There are two key aspects of a chief programmer and his/her team.

First: Each member of the team carries out only those tasks for which he or she has been trained.

Second: The chief programmer directs the actions of all the other members of the team and is responsible for every aspect of the operation. There is a hierarchy.

This is the way the team is formed. The chief programmer was both a successful manager and a highly skilled programmer who did the architectural design and any critical or complex sections of the code. The other team members worked on the detailed design and the coding, under the direction of the chief programmer. No lines of communication existed between the programmers; all interfacing issues were handled by the chief programmer. Finally, the chief programmer reviewed the work of the other team members, because the chief programmer was personally responsible for every line of code.

The position of backup programmer was necessary only because the chief programmer was human and could therefore become ill, fall under a bus, or change jobs. Therefore, the backup programmer had to be as competent as the chief programmer in every respect and had to know as much about the project as the chief programmer. In addition, to free the chief programmer to concentrate on the architectural design, the backup programmer did black-box test case planning and other tasks independent of the design process.

The word secretary has a number of meanings. A secretary can be a person who assists a busy executive by answering the telephone, typing correspondence, and so on. But when we talk about the American Secretary of State or the British Foreign Secretary, we refer to one of the most senior members of the Cabinet. The programming secretary was not a part-time clerical assistant but a highly skilled, well-paid, central member of a chief programmer team. The programming secretary was responsible for maintaining the project production library, the documentation of the project. This included source code

listings, JCL, and test data. The programmers handed their source code to the secretary, who was responsible for its conversion to machine-readable form, compilation, linking, loading, execution, and running test cases. Programmers therefore did nothing but program. All other aspects of their work were handled by the programming secretary.

4.2.3 The Modern Approach (Chief Programmer & Business Manager)

The modern approach is to remove much of the managerial role from the chief programmer. After all, the difficulty of finding one individual who is both a highly skilled programmer and successful manager has been pointed out. Instead, the chief programmer should be replaced by two individuals: a team leader in charge of the technical aspects of the team's activities and a team manager responsible for all nontechnical managerial decisions. The team leader is responsible for only technical management. Consequently, budgetary and legal issues are not handled by the team leader nor are performance appraisals. On the other hand, the team leader has sole responsibility on technical issues. The team manager therefore has no right to promise or say that the product will be delivered within 4 weeks; promises of that sort have to be made by the team leader. The team leader naturally participates in all code reviews; after all, he or she is personally responsible for every aspect of the code. At the same time, the team manager is not permitted at a review, because programmer performance appraisal is a function of the team manager. Instead, the team manager acquires knowledge of the technical skills of each programmer in the team during regularly scheduled team meetings.

4.3 RECOMMENDATION FOR TEAM4OIES

TEAM4OIES will utilize a mix of Modern and Democratic Team. We want all the team members to write code and be egoless about the way it is reviewed. TEAM4OIES will use OOP low coupling modules and high cohesion.

4.4 CONCLUDING REMARKS

Choosing the appropriate team organization is vital to any project. TEAM4OIES will use the mistakes from history to choose the correct Team Organization.

5. CHAPTER V: CHOOSING THE RIGHT MODEL

5.1 PROBLEM STATEMENT

There are many software life-cycle models, each with their own differences and qualities. When selecting the correct model to follow, many aspects need to be taken into consideration. These aspects may include the size of the project, the number of features required, the number of team members working on the project, and if postdelivery maintenance is needed. The right model can only be chosen after careful evaluation of the product being developed.

5.2 COMPARING MODELS

5.2.1 Waterfall Model

There are many variations of the waterfall life cycle model. The typical waterfall model consists of six phases: requirements, analysis, design, implementation, postdelivery maintenance and retirement. This model is heavily document driven.

In the requirements phase, the client's requirements are determined. These requirements are analyzed and drafted in the form of a specification document in the analysis phase. Next, the specifications go through the architectural design and detailed design procedures to produce a design document in the design phase. The design document describes "how the software product does it." In the implementation phase, the team develops the code and perform testing on the code to make sure that it runs correctly. This phase ends when the client tests the product and approves of it. Postdelivery maintenance is performed on the product to make sure the product is effective and running when the environment changes. The last phase, retirement, occurs when the client deems the product is no longer of use to them and is removed from service.

5.2.2 Incremental Model

Instead of phases, the incremental model contains workflows performed over the entire life cycle. The life of the product is divided up into parts or increments. The workflows include the requirements, analysis, design, implementation, and test workflow. The workflows are done throughout the whole life cycle, but there will be times when one workflow is dominate over the other workflows. For example, at the beginning of the life cycle, the requirements workflow is the main workflow, but its importance lessens as the life cycle progresses. Near the end, the implementation and test workflows are more important and will consume more of the team's time.

Iteration occurs during each increment and involves all workflows in varying degrees. All planning, documentation and testing tasks are done at the beginning of product development and will continue until the product is completed and delivered to the client.

5.2.3 Extreme Programming

Extreme programming is an approach to software development based on the iterative-and-incremental model and is often referred to as agile processes. It begins with the team determining what features the client wants in the product and a successive build will be created containing these features. The build will then be broken down to smaller parts called tasks. Then it goes through test driven development in which the programmers draw up test cases for each task.

Extreme programming also incorporates pair programming where two programmers implement the task and ensure that all test cases work correctly on a single computer. The task will then be integrated into the current product. All members of the team work on the requirements, analysis, design, coding and testing.

This model utilizes a time management technique called timeboxing, in which time is set aside for each task and team members do as much as possible to complete the task within that time frame. XP also holds short stand-up meetings to raise issues or concerns and hold follow up meetings to address solutions to these problems.

5.2.4 Synchronize and Stabilize

The Synchronize-and-Stabilize model is also based on the iterative-and-incremental model. The model starts with determining a list of features with the highest priority by interviewing several possible clients. The work is divided into three or four builds. The first build usually contains the most essential features whereas the next build will contain the next most important features, and so on.

Each build has a team of developers and all teams work in parallel. At the end, the teams will combine their parts together while testing and checking for errors. This process is called synchronize. After fixing the faults, the product will undergo stabilization where the developers will freeze the build and no other changes will be made to the specifications.

The advantage of this model is that the process of synchronization ensures that all components will work together. Even if the specification is incomplete, the Synchronize-and-Stabilize model could still be used. This model also provides developers with early understanding of how the product operates and therefore will allow for adjustment of requirements if needed.

5.2.5 Spiral Model

The Spiral Life Cycle model uses prototypes and other means to minimize risks. A prototype is an incomplete version of the software product used to minimize risks as well as give information about classes of risk. If the prototype closely resembles the actual product, it can help control risks. One associated risk may be time constraint.

Before each phase starts, a risk analysis is conducted to mitigate the risks. A phase begins by first defining the objectives of that phase and ways to achieve the objectives. A strategy will then be developed for accomplishing these objectives. This strategy will then be analyzed with the intention of reducing risks. The project is terminated if developers find that it is impossible to mitigate all major risks. Otherwise, the project will continue onto the next phase. This model is typically used for large scale projects and requires developers to be knowledgeable in risk analysis.

5.2.6 Object-Oriented Model

The Object-Oriented Model is versatile as it allows for reusability. With an object-oriented model, objects are independent from one another. This allows for parts of the codes to be used for future projects as well as borrowing from previous projects. This also reduces complexity of the modules and allows for developers to efficiently see how they fit together. Because the code is broken up into smaller parts, detection of futures faults and debugging is made easier.

Similar to the classical model, this model contains the requirements workflow, the analysis workflow, the design workflow, the implementation workflow, postdelivery maintenance, and retirement.

5.3 RECOMMENDATION FOR TEAM4OIES

We recommend a combination of the waterfall and iterative-and-incremental models for TEAM4OIES. In the initial stages of our software development, TEAM4OIES was required to identify our client's requirements. Then we produced documentation showing what TEAM4OIES analyzed similar to what is done in the waterfall model. We also recommends the iterative-and-incremental model as it follows the agile software development process.

5.4 CONCLUDING REMARKS

Choosing the right software life-cycle model is vital when developing software. Each model has its own advantages and disadvantages, so it is important to first understand which benefits each model offers and which will best fit the team and the product that is being developed. Using the wrong model or one that is a poor fit will hinder progress and ultimately, may lead to failure.

6. CHAPTER VI: ANALYSIS AND DEVELOPMENT METHODS

6.1 PROBLEM STATEMENT

A software engineer must use various tools to increase efficiency, cost, and time for a project. Which of these tools decrease time in a way that does not affect cost or efficiency? Which tool increases efficiency without adding too much cost and time? Which of these minimize time but stay efficient?

6.2 CONVENTIONAL ENGINEERING METHODS

6.1.1 Stepwise Refinement

Stepwise Refinement is development of a project by first describing general functions and breaking each of these functions into 7 plus or minus tasks until the entire project is entirely defined. 7 plus or minus comes from Miller's Law which states that a human can only remember 7 plus or minus things at one time.

6.1.2 Cost Benefit Analysis

Cost benefit analysis is the process of estimating strengths and weaknesses or giving other approaches to satisfy the client if the current approach is too costly. Decisions are based in terms of time and cost saving usually.

6.1.3 Software Metrics

The 5 software metrics include: size, cost, duration, effort, and quality. Software metrics are used to analyze software quantitatively and can be helpful in determining faults or finding solutions.

6.1.4 CASE

Case tools or computer-aided software engineering are tools that help manage workflow. Using CASE tools is very important in developing high-quality, maintainable, and code with very few bugs. CASE tools are either used for Upper CASE (front end) or Lower CASE (backend). These tools can also be divided into 6 categories, business and analysis modeling, development, verification and validation, configuration management, metrics, and project management.

6.1.5 Software Versions

As a product is developed, the product will get updated and modified. When this occurs, the developers need a way to keep track of these changes and this is done by using software version. Versions are usually kept track of in a versioning system but TEAM4OIES are keeping track of it ourselves by modifying it in our documentation.

6.1.6 Configuration Management

This tool controls the check-in and check-out of repository objects and files. TEAM4OIES are using SVN to accomplish this.

6.1.7 Problem Statement

Please see above 6.1 Problem Statement.

6.1.8 Other Applicable Engineering Tools

Other tools that are easily useable, include compilers, data modeling tools, and text editors. These are usually very expensive and therefore TEAM4OIES cannot use these in our product due to constraints of budget and time.

6.3 RECOMMENDATION FOR TEAM4OIES

To complete this project, TEAM4OIES will definitely make use of our repository. We also strongly recommend TEAM4OIES use Visual Studio 2010 to develop our project due to it having a built in DBMS having an easy to use C# compiler. Other useful things will include MYSQL to code queries to our database. Also, we recommend using Microsoft Word to handle documentation due to it being easy to get, and being a very good text editor.

Use svn
& URLs
connect

6.4 CONCLUDING REMARKS

There are many potential and powerful tools that are available to use on our Software Product. Even though tools exist, TEAM4OIES must remember that TEAM4OIES can only

use free software due to an issue in cost, and simple software due to a constraint in time. Finding the right software for our team is essential to building a great software product.

7. CHAPTER VII: TESTING

A. PROBLEM STATEMENT

Testing is important part of software development and development team must carry out this process throughout. There are two types of testing, first one is execution-based testing and second one is non-execution testing.

B. QUALITY ASSURANCE

The job of SQAs is to check and test the work of developers once they have finished a workflow. SQA group must assure quality of product and this is done by assuring quality of the software process.

Non-Execution-Based Testing

Non-Execution-Based testing does not include any code, but it includes careful review of software and mathematical analyzing of software. It is important that person who worked on the document is not the one testing it. There are two types of reviews: Walkthroughs and Inspections. Walkthrough team is made up of four to six members and they typically include: manage responsible for workflow, client rep, member of team that will be performing next workflow, SQA rep. SQA rep should carry out the walkthrough because they have the most at stake. Walkthrough is an interactive process and should include many questions and discussion. As mentioned above walkthroughs are less formal, where on the other hand Inspection is more formal and has five steps.

1. Overview: This is an overview of the document to be inspected.
2. Preparation: Understanding of document in detail and finding faults.
3. Moderator: Leader of the team will produce report of the faults.
4. Rework: Faults and problems documented above are resolved.
5. Follow-up: Moderator makes sure all the faults and problems are corrected and no new faults were introduced.

Execution-Based Testing

In execution based testing, the program is tested using selected input in the environment that the program will be used in. Sometimes this type of testing is not the best because even if the program passes the selected input test it does not mean there exist no faults in the software. Execution-based testing should include the following: utility, reliability, and correctness.

C. Recommendation for TEAM4OIES

For our project we would recommend TEAM4OIES use both type of testing as explained above. Even though SQAs play active roles in testing, we believe it is everyone's job to make sure software correctness through above testing methods.

D. Concluding Remarks

TEAM4OIES agrees with both types of testing and will implement them through the life of this project. We think by using the testing methods mentioned above, it will make our final product bug free and post-delivery maintenance will be easy.

8. CHAPTER VIII: DEVELOPING TEAM4OIES PAGE MASTER AND HOME PAGE

8.1 PROBLEM STATEMENT

As with most projects that are being developed for a client, many variables can get in the way, and cause a product to not meet one, or more, of the three main requirements. These shortcomings can be seen clearly when the deadline is missed, the cost of the product goes over budget, or the product doesn't meet the reliability that was expected of it. This is why communication is a vital connection that the client and developer need to have if these problems are to be avoided. By communicating the developers can better understand what the clients are looking for, and the clients can give better details as to what they want to see in the product. It is through the implementation of the requirements workflow that the developers are able to understand the domain that the clients are after.

8.2 TEAM4OIES PAGE MASTER AND HOME PAGE

-to where are they?

The page master and the home page are starting blocks for the site design and will influence how the rest of the site is built. However, TEAM4OIES, the developers, are also responsible for making a site that matches the client's requirements. The main page will contain a header, footer, navigation section with links to different parts of the website, as well as different other sections that will make up the site. This content is mostly dictated by the requirements specified in the textual analysis, and a goal to make the experience for any users on the site as simple and straight forward as possible. Through the use of design tools, proper layout, and proper coding, TEAM4OIES hopes the website can properly convey the client's ideas, and at the same time give an appealing look to properly sell their idea.

8.3 CONCLUDING REMARKS

It is with the implementation of the requirements workflow that will better understand the client's needs, expectations, and goals for this product. TEAM4OIES will be able to understand the domain more, and better plan ahead for other additions the client may have. To understand the domain the developers will perform textual analysis on all the information they are given, and being sure to ask more questions if a need arises. With

proper planning in place, the proper tools to utilize, and a firm understanding of the client's domain, TEAM4OIES will be able to handle, and even predict, problems that may arrive in the future. That is how TEAM4OIES, as the developers, will be able to properly maintain the deadline, cost, and reliability.

↓ new page

DOCUMENT CONTROL

CHANGE HISTORY

Table 1: TLs entries (assigned work and due dates) before releasing to the team (all SQAs)

Revision	Name	Due Date	Description
1.A	TM Daniel Gonzalez	02/08/2015	Add your work to Document
1.B	TM Sarah Moore	02/08/2015	Add your work to Document

1.C	DBA Jainesh Mehta	02/08/2015	Add your work to Document
1.X	Linh Tong	02/09/2015	Review Document
1.Y	Paul Miller	02/10/2015	Review Document

Table 2: Entries when work completed (SVN Commit Comment matches Description)

Revision	Name	Completed Date	Description
1.A	TM Sarah Moore	02/10/2015	Added third chapter to the document.
1.B	TM Daniel Gonzalez	02/10/2015	Added chapter one to the report
1.C	DBA Jainesh Mehta	02/10/2015	I added chapter two to the report.
1.X	SQA Paul Miller	02/10/2015	I reviewed Document
1.Y	SQA Linh Tong	02/10/2015	I reviewed Document

Table 3: TL entry for RED DELIVERABLES (SVN Commit Comment matches Description)

Revision	Name	Due Date	Description
2.0	TL Javier Rivera	02/11/2015	I changed Version to 2.0 ✓

CHANGE HISTORY

Table 1: TLs entries (assigned work and due dates) before releasing to the team (all SQAs)

Revision	Name	Due Date	Description
2.A	TL Javier Rivera	03/06/2015	Add your work to Document
2.B	DBA Jainesh Mehta	03/06/2015	Add your work to Document
2.C	DBA Logan Stark	03/06/2015	Add your work to Document
2.X	SQA Linh Tong	03/07/2015	Review Document
2.Y	SQA Paul Miller	03/08/2015	Review Document

Table 2: Entries when work completed (SVN Commit Comment matches Description)

Revision	Name	Completed Date	Description
2.A	TL Javier Rivera	3/04/2015	I Completed the Assigned work and now waiting for review.
2.B	DBA Jainesh Mehta	3/06/2015	I added CHAPTER VII: TESTING.
2.C	DBA Logan Stark	03/06/2015	I added Chapter VIII
2.X	SQA Linh Tong	03/07/2015	I reviewed Document
2.Y	SQA Paul Miller	03/07/2015	I reviewed Document

Table 3: TL entry for PURPLE DELIVERABLES (SVN Commit Comment matches Description)

Revision	Name	Due Date	Description
3.0	TL Javier Rivera	03/08/2015	I changed Version to 3.0 ✓

DOCUMENT STORAGE

This file is stored in SVN at [https://svn.cs.uh.edu/svn/cosc4351/team4/TEAM PROJECT DELIVERABLES/Team Project Report.doc](https://svn.cs.uh.edu/svn/cosc4351/team4/TEAM_PROJECT_DELIVERABLES/Team%20Project%20Report.doc).

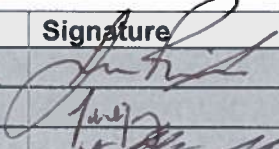
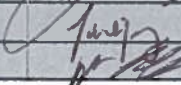

No Content Web Site
pages

TEAM4OIES

Web Site Design

Version 2.0

Signature Block

COSC 4351	Name	Signature	Date
SE Team Leader	Javier Rivera		3-9-15
SQA	Linh Tong		3-9-15
SQA	pavl Miller		3-9-15

Content Outline

Master Page / wireframe

- Links
 - Doesn't change and controls the home page view
 - Links will include:
 - Home
 - Testimonials
 - About Us
 - Contact Us
- Log in
 - When you first enter it will display as a log in with username and password
 - After signing in will display some links for account info
- Chat box
 - Its own section.
 - Contains the chat box window. One chat for whole site.
 - This could be upgraded later for better support options.
- Home Page
 - Center of Page
 - This is what changes when different links are pressed.
 - Home:
 - Stating page
 - General information
 - Testimonials
 - About Us
 - Contact Us

TEAM4OIES

Search

Links

Home Page

Photo

news

Snag line statement

Footer

TEAM4OIES

Search



Links

About Us

Mission Statement

Position

img

Position

Position

Footer

1

TEAM4OIES

2

3

Search

Links

Contact Us

Address,
Phone number
Other information

Comment
Submission
Form

Footer

TEAM4OIES

Search

Links

Testimonial

Search

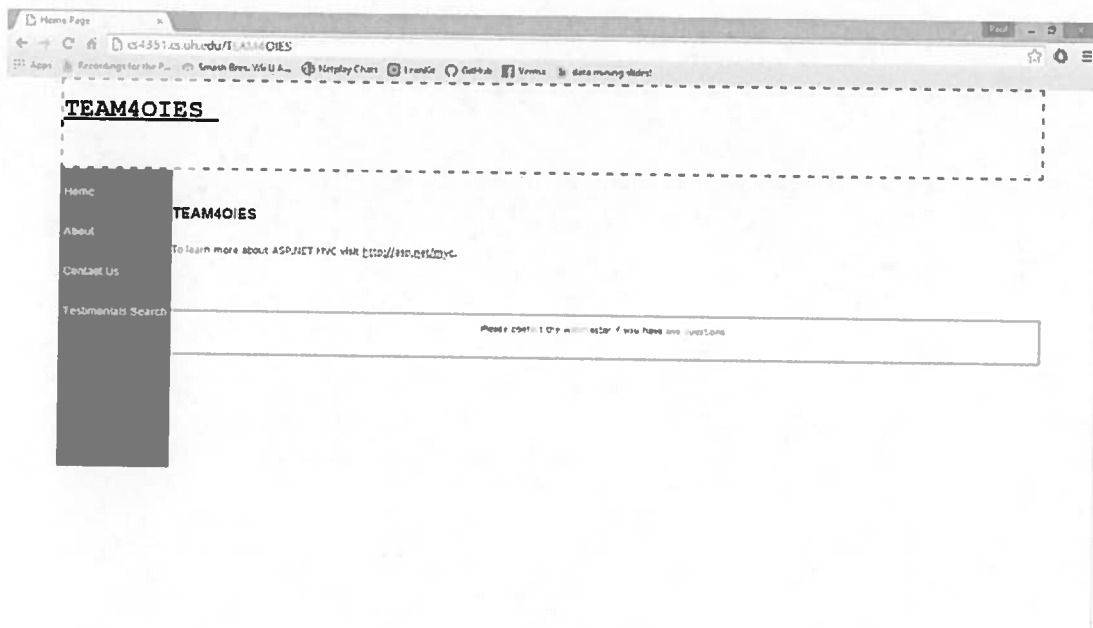
Testimonial 1

Testimonial 2

Testimonial 3

Testimonial 4

Footer



Document Control

CHANGE HISTORY

Table 1: TLs entries (assigned work and due dates) before releasing to the team (all SQAs)

Revision	Name	Due Date	Description
1.A	TM Janaye Maggart	03/06/2015	Complete XXX
1.B	TM Sarah Moore	03/06/2015	Complete YYY
1.C	DBA Logan Stark	03/06/2015	Complete ZZZ
1.X	SQA Linh Tong	03/07/2015	Review Document
1.Y	SQA Paul Miller	03/08/2015	Review Document

Table 2: Entries when work completed (SVN Commit Comment matches Description)

Revision	Name	Completed Date	Description
1.A	TM Janaye Maggart	03/07/2015	I aided in planning the rough draft of the website design
1.B	TM Sarah Moore	02/26/2015	I completed the rough draft of the website design
1.C	DBA Logan Stark	03/06/2015	I completed the content outline and rendition of master page wireframe
1.X	SQA Linh Tong	03/07/2015	I reviewed Document
1.Y	SQA Paul Miller	03/08/2015	I reviewed Document

Table 3: TL entry for RED DELIVERABLES (SVN Commit Comment matches Description)

Revision	Name	Due Date	Description
2.0	TL Javier Rivera	03/08/2015	I changed Version to 2.0

DOCUMENT STORAGE

This file is stored in SVN at <https://svn.cs.u.edu/svn/cosc4351/team4/TEAM PROJECT DELIVERABLES/Web Site Design.doc>.

Iterated 2

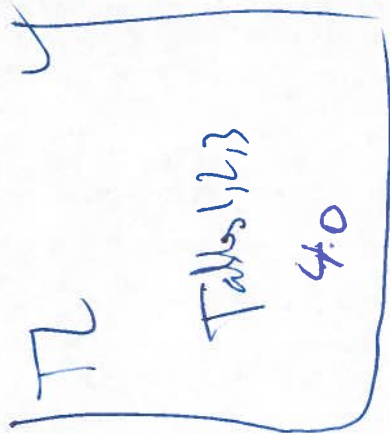
✓ Used

PURPLE SE
Mice (For Actors)
Not for UCs

TEAM4OIES

(100)

SE Team Project with Line Numbers TEXTUAL ANALYSIS for Requirements Workflow UML USE CASE DIAGRAM



15

12

6

5

Version 3.0 ✓

20

Input Forms and Output Reports

↓
These are coming
from the Requirement
IF 21

Version 8.0

Signature Block

COSC 4351	Name	Signature	Date
SE Team Leader	Javier Rivera		
SE SQA	Linh Tong		
SE SQA	Paul Miller		
SE Team Leader			
SE SQA			
SE SQA			
SE Team Leader			
SE SQA			
SE SQA			