Software Quality Assurance Plan

I. Table of Contents

1.0 Introduction

This section gives a general overview of the Software Quality Assurance Plan (SQA) for the **W**aste **M**anagement **I**nspection **T**racking **S**ystem (WMITS) version 2.
SQA will focus on the management issues and the process specific activities that enable a software organization to ensure that it does "the right things at the right time in the right way."

SQA plan provides a road map for instituting software quality assurance. Developed by the SQA group and the project team, the plan serves as a template for SQA activities that are instituted for each software project.

1.1 Scope and Intent of SQA Activities

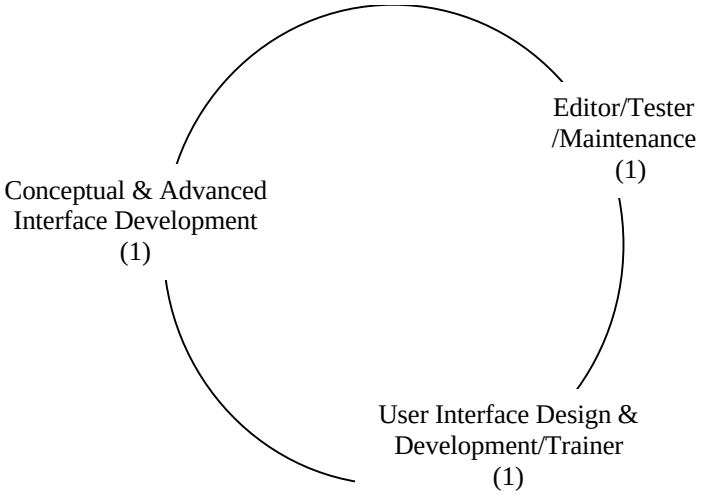
The objectives of SQA are:

- A quality management approach
- Effective software engineering technology (methods and tools)
- Formal technical reviews that are applied throughout the software process
- A multi testing strategy is draw
- Control of software documentation and the changes made to it
- A procedure to assure compliance with software development standards when applicable
- Measurement and reporting mechanisms

1.2 SQA Organizational Role

We have a relatively small team, only 3 members. So the team will use the *Ego-less* structure.

Ego-Less Structure

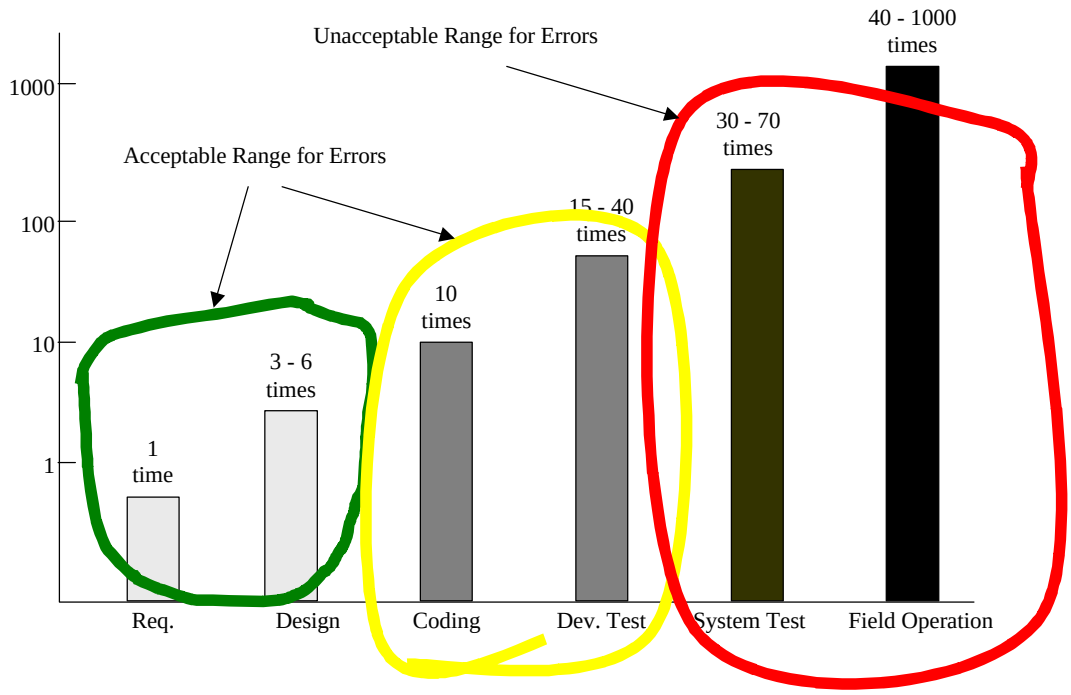


- Conceptual & Advanced Interface Development: Ray Yu
- User Interface Design & Development/Trainer: John Deng
- Editor/Tester/Maintenance: Bhavin Patel

A small team has its advantages and disadvantages. Communication is easier among other team members. The team has a very close contact with each other. We hold regular meeting, daily update using e-mails, phone calls, etc. The disadvantage of a small team is between each team member, we have many overlapping job functions. As good example, the project tester, for a regular team might be dedicated to this task. For our team, we do not have a separate member as the tester (he/she may also write the help menu, or even doing documentations.), but since we don't have any extra personnel to spare, so we have to test our part while we are writing it. And for the final testing, we will do it all together. To prevent the case that we might miss any test case, we will invite a member from our client to monitor our testing process.

Using the chart in the Pressman book [SEPA, 4/e], the cost of correcting an error stated as below:

Relative Cost of Correcting an Error



If goes according to plan, Quality Assurance Plan will minimize the errors the team made. Of course, eliminate all errors will be out of question. Discover errors during the Reg. & Design stage is our goal; During Coding & Dev. Testing is still expectable; we certainly can't make (discover) any errors during the System Test & Field Operation phase. Because the result will be disaster for the team, we are running out of time, to make it simple.

A wag once said, "Every program does something right, it just may not be the thing that we (clients) want it to do." We have a very close contact with our client; a regular meeting schedule has been set up; we keep constant contact with them through e-mail; an online help desk will set up soon.

2.0 SQA Tasks

Here are the tasks we have for the SQA:

- Voting system
- Close contact with the client
- Extensive detail design
- Research on the subject

2.1 Task Overview

Tasks that described above will cover the product quality control, Saving Design Time & Cost, Minimize Errors, Problem Tracking & Data Flow.

2.2 Standard, Practices and Conventions (SPC)

Voting system

The Cyber Rovers team has only three members, so there is no leader or supervisor, although the voting system works out fine for us. Whenever we make some decision for the change of the project, we vote for it. Each team member either supports it or opposes it.

Comment: This team organization is democratic, decentralized and is appropriate for this project.

Close Contact with the Client

So far, we already have 3 meeting with the client. We have meet with the Environmental Quality Manager and two Environmental Quality Analysts. They have been very helpful and cooperative. Also, we have had contact with the network administrator of DEQ, although not much help there. (It's either he didn't get our mail or didn't know what we talking about.)

During each meeting, we discuss the problems, options we have come up with, with their experience on the field, and they provide solutions for the team.

The below is the table for the meeting. We have 3 meetings with our client so far. Which average 2 – 3 weeks per meeting. Each meeting lasted about 2+ hours.

Comment: Good mechanism for improving overall product quality.

Meeting Schedule Table

Meeting Date	Discussions/Subjects	Time
January 10, 2000	- Project background - Basic software function - Gather project information - Client request	8:00am – 9:56am
January 21, 2000	- Present draft designs of the interface - Present different design options - More client request	2:00pm – 4:11pm
February 15, 2000	- Present detail designs of the interfaces - Confirm client request - Client more request	2:00pm – 4:00pm
March 14, 2000	- Present the basic prototype - Confirm client request - Client more request	
To be scheduled	- Present a semi-working software - Minor function changes - Cosmetic changes	
To be scheduled	- Present a nearly-working software - More office testing (by inspectors) - Cosmetic changes	
To be scheduled	- Present the final version	

Extensive Detail Design

The last team who work on the **W**aste **M**anagement **I**nspection **T**racking **S**ystem (WMITS) version 1 use Visual Basic to implement the software. Even the client request we don't have to use the same implementation method, but because limited time we have, the idea of putting the software in browser base is out of question.

After we define the method we will use to design the project, we will start to design the interfaces for the software. Based on the Version 1 design, & the request from the client, we come up with 9 new design interfaces. Each interface will include all the elements that the client wanted, and the new elements that we want. But the functions for each element wouldn't be implemented. We present the draft interface to the client (in the 2nd & 3rd meeting). Base on the input of the client, we do more modification on the interface, add/delete elements. And also come up with 6 more new meetings

The process described above wouldn't involve any "actual" coding. Instead of clearly separate the design and coding phase, we do the simple interface (element layout design) coding in the design phase. The advantage for this is to refresh the project background &

provide a good exercise for two of the team members who never have any VB programming experience.

Research on the Subject

Version 1 has lot bugs or unknown errors. We were able to get a hold of the copy of the program. After an inspection and test on the software, we estimated Version 1 has complete about 30% - 40% of the tasks that required by the client. Many of functions that sit behind the interfaces can be reused, even though we are going to redesign the interface or rearrange the elements on the old interface.

For the extended enhancement, palm pilot integration part, although we have some idea how to approach the problem, we still haven't got any feedback from the client for the specific brand or price range they want. So we can't do any actual design on that part.

Comment: This represents a significant project risk and also a potential quality problem. In addition to the obvious issues of learning how to program the device that is finally selected, interfacing issues and other matters will ultimately have to be addressed.

2.3 SQA Resources

No external SQA resources are defined for this project.

3.0 Reviews and Audits

A formal technical review (FTR) is a software quality assurance activity that is performed by software engineers. The objectives of the FTR are:

(1) to uncover errors in function, logic, or implementation for any representation of the software;
(2) To verify that the software under review meets its requirements;
(3) To ensure that the software has been represented according to predefined standards;
(4) To achieve software that is developed in a uniform manner;
(5) To make projects more manageable.

3.1 Generic Review Guidelines

3.1.1 Conducting a Review

There are two kinds of reviews we'll do, review cases with the client, and review cases with other teammates.

For the changes that will affect the clients' performance when they use the software, we have to consult them first. But before taking the cases to the client, the entire team member has to agree with the change. And keep a good record of the project before and after changes.

3.1.2 Roles and Responsibilities

As stated in 1.2, SQA Organizational Role, the role of each team member will multi-dimensional since we have a relatively small team.

- Conceptual & Advanced Interface Development: Ray Yu
- User Interface Design & Development/Trainer: John Deng
- Editor/Tester/Maintenance: Bhavin Patel

3.1.3 Review Work Product

For each period (weekly, in our case), we'll generate a work report from each member. In the work report, we will state each member's work for the past week, problems encountered, problem that can't be solved, and any cautions. This report will be extremely helpful when comes to documentation and writing the help menu.

When we meet with our client, we will have minutes for the record. In the minutes, we will have the questions that we make before we go see the client, answered got from the clients, clients' request, opinions on the current work, self reminder, schedule for next meeting, etc.

3.2 Formal Technical Reviews

Here are the FTR that we will conduct during the software process:

- Walkthroughs
- Inspections

After each form (interface) we design, we'll do a test on the interface using block box testing method. And each week, we will ask the team members to do a inspection on the interface, then hook up the other's work, do a walkthroughs of all the interfaces.

3.2.1 Description of review *Walkthroughs*

3.2.1.1 Description and Focus of Review

This review mainly focuses on the integration of the parts that we design (such as interfaces, form, database.) We will ask other team members to do the walkthroughs with the presence of the coder.

3.2.*2* Description of Review *Inspection*

3.2.2.1 Description and Focus of Review

This review mainly focuses on the correctness of the parts that we designed. Usually allow the other two team-members to do a private test, without the designer's interruption. The idea is try to allow other team-members bring out the test cases.

3.2.1 System Specification Review

The system specification is usually changed after each weekly meeting, and after each meeting with our client. As of this moment, most of the system designs have stablized. And since we are actually doing a "bug fixing" originally, so the system specification is pretty much there. Even though we have extended the project into a higher level, the basic mapping of the project is still the same. For more information, please see the document titled "System Specification".

3.2.2 Software Project Plan Review

The purpose of Software Project Plan is over look of the whole project. For more information, please see the document titled "Software Project Plan".

3.2.3 RMMM Review

RMMM, Risk Mitigation, Monitoring & Management, is use to prevent, monitor, and manage the risk. For more information, please see the document titled "RMMM".

3.2.4 Requirements Reviews (Models, Specification)

Software Requirements stated the data requirement, specifications. For more information, please see the document titled "Requirement Plan".

3.2.5 Data Design Review

The Data Design document is about the data flow between each form (interface), and forms to the database. For Architectural Design, please see the document titled "Architectural Design".

3.2.6 Architectural Design Review

The Architectural Design document is about the whole project design, layout, and data flow. For Architectural Design, please see the document titled "Architectural Design".

3.2.7 Interface (GUI)

Up on the request of the client, we will redesign the interfaces from the previous version. We have 15 interfaces so far, and maybe more will be added later on. They all are Visual Basic front-end.

3.2.8 Component Design Review

Besides the 15 interfaces, we also doing a palm pilot integration, and online help desk for a period of time. We also doing some database re-engineering for the client's database to make it more efficient and suit our project need.

3.2.9 Code Review

3.2.10 Test Specification Review

3.2.11 Change Control Reviews and Audits

Critique: Sections 3.2.x require substantial rework. The intent of these subsections is to discuss the mechanisms used for conducting reviews, the frequency of reviews, the people who will participate, the questions to be asked (or references to appropriate checklists), and the criteria used to deem the work product acceptable. None of these issues is addressed above. All should be.

3.3 SQA Audits

- Team members will have a weekly report on their individual performance for the past week. Any problems, question regardless on the performance of other team members will also noted there.
- Members will write part of the help menu that relates to their design work. And they also share between members.
- Any changes that will affect the project will be presented to other team members before doing any changes. These are the changes that are minor or require little code change, but still are different from the original architectural design.
- The client should be notified of all changes made to the. For minor changes, we will just notify a reprehensive from the client instead of the whole team from the client. This rule only applies to the minor changes or cosmetic changes, or minor functional changes. Any major functional change will still require the agreement from the whole team from the client side (two inspectors and one division manager).
- A changed version will also been carefully recorded. That included the copy of the project before and after the changes. And noted on the document that what kind of change will be make on where and by who. A version system always helpful. See software project plan 6.2 for more details.

Critique: The discussion above would be better placed in the SCM Plan, where a focus on change control dominates. It seems out of place here. The intent of the SQA audit is to assess the teams approach to QA and determine if there are any weaknesses. In general, an SQA audit would not be performed for a project of the size and duration of for the **W**aste **M**anagement **I**nspection **T**racking **S**ystem (WMITS) version 2.

4.0 Problem Reporting and Corrective Action/Follow-up

This section will describe problem reporting mechanisms that occur as a consequence of the FTRs that are conducted and the means for corrective action and follow-up.

4.1 Reporting Mechanisms

We will use the “tiny tool” that we designed on CIS 375 to do problem reporting, it is browser-based program, which is very convenient for the team members. Site name is

http://www.cyberrovers.com/crr. It not only used by the team members to report problems to other team members, we can also train the clients to use it during the testing phase. When they submit the form, they will receive a copy of the request. The copy also "carbon copies" itself to the DEQ manager. The request will send to all three members of the cyber rovers' team.

An online help desk will also put online for the client ask any question regard on the operation of the software. This service will be provide during the training period, or up on the request of the clients.

4.2 Responsibilities

Since we use the egoless team model, we won't select a team leader. But since Ray has a lot of real world experience on software development and has great knowledge on the project, so he's the de facto leader for the team. But as far as decision making, no changes will be make unless all three members agree on it.

- Conceptual & Advanced Interface Development: Ray Yu
- User Interface Design & Development/Trainer: John Deng
- Editor/Tester/Maintenance: Bhavin Patel

4.3 Data Collection and Valuation

To properly conduct software quality assurance, data about the software engineering process should be collected, evaluated, and disseminated. Statistical SQA helps to improve the quality of the project and the software process itself.

During each meeting, we will present a client a prototype of the project; ask the clients' opinion on the design, present different option for them to choice in many cases. Even during the two weeks that between two meetings, if we try to keep the client have the latest and 1st hand knowledge on the process of the project, we occasionally posted the project online, so letting them exam the process. Or let them download a prototype version, let them run it in the real world enlivenments, and then ask for feedback.

4.4 Statistical SQA

Statistical quality assurance reflects a growing trend throughout industry to become more quantitative about quality. For software, statistical quality assurance implies the following steps:

- Information about software defects is collected and categorized.
- An attempt is made to trace each defect to its underlying cause (e.g., nonconformance to specification, design error, violation of standards, poor communication with customer).
- Using the Pareto principle (80% of the defects can be traced to 20% of all possible causes), isolate the 20% (the "vital few").
- Once the vital few causes have been identified, move to correct the problems that have caused the defects.

Although hundreds of different errors can be uncovered, even for a "small scale" project like this, all can be tracked to one (or more) of the following causes:

- Incomplete or erroneous specification (IES)
- Misinterpretation of customer communication (MCC)
- Intentional deviation from specification (IDS)
- Violation of programming standards (VPS)
- Error in data representation (EDR)
- Inconsistent module interface (IMI)
- Error in design logic (EDL)
- Incomplete or erroneous testing (IET)
- Inaccurate or incomplete documentation (IID)
- Error in programming language translation of design (PLT)
- Ambiguous or inconsistent human-computer interface (HCI)
- Miscellaneous (MIS)

	Total		Serious		Moderate		Minor	
Error	No.	%	No.	%	No.	%	No.	%
IES								
MCC								
IDS								
VPS								
EDR								
IMI								
EDL								
IET								
IID								
PLT								
HCI								
MIS								
Total		100%		100%		100%		100%

E*i* = the total number of errors uncovered during the *i*th step in the software engineering process
Si = the number of serious errors
M*i* = the number of moderate errors
Ti = the number of minor errors
PS = size of the product (LOC, design statements, pages of documentation) at the *i*th step

At each step in the software engineering process, a phase index, PI_i, is computed.
$PI_i = w_s (S_i/E_i) + w_m(M_i/E_i) + w_t(T_i/E_i)$

The error index (EI) is computed by calculation the cumulative effect or each $PI_{i,.}$ Weighting errors encountered later in the software engineering process more heavily than those encounter earlier.

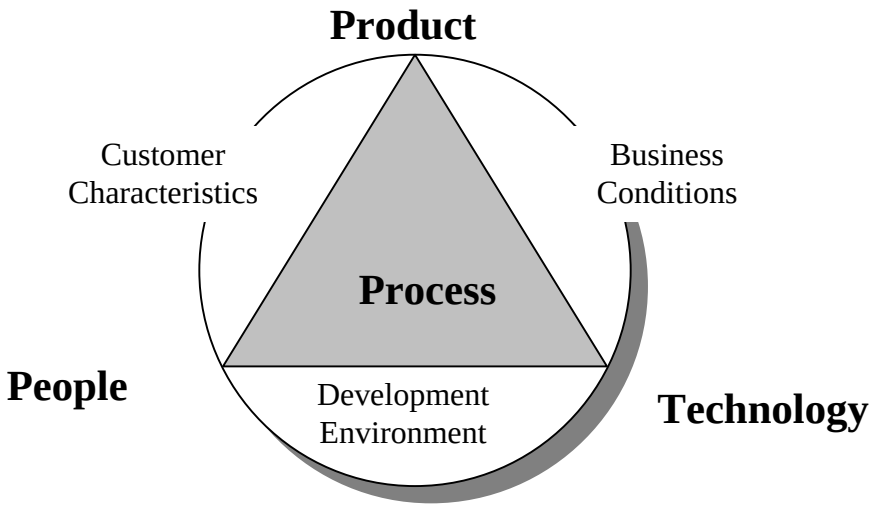
EI $= \sum (i \times Pi)/PS$

$= (PI_1 + 2PI_2 + 3PI_3 + iPI_i) / PS$

The above table and functions will be used in future, since we don't have enough data to support it yet.

5.0 Software Process Improvement Activities

Determinants for Software Quality and Organizational Effectiveness



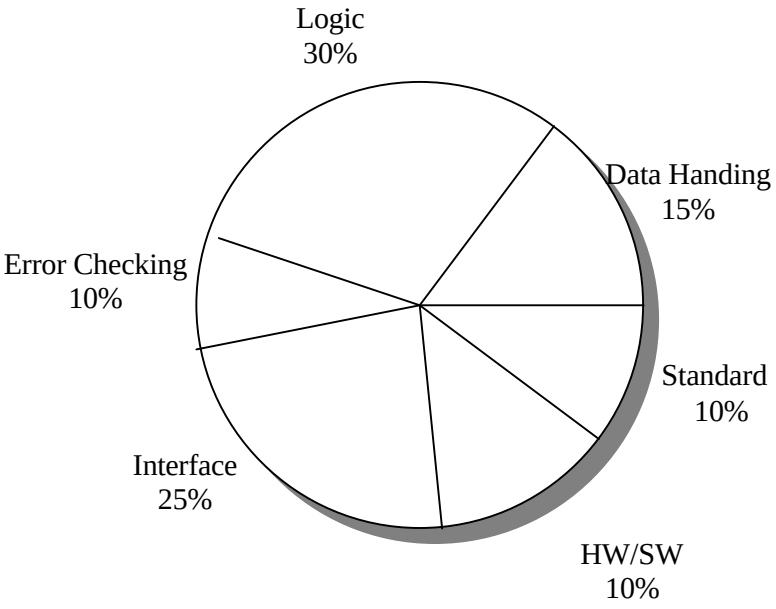
5.1 Goal and Object of SPI

Here are some of the goals of SPI:

1. All errors and defects are categorized by origin (e.g. flaw in specification, flaw in logic, nonconformance to standards).
2. Te cost to correct each error and defect is recorded.
3. The number of errors and defects in each category are counted and ordered in descending order.
4. The overall cost of errors and defects in each category is computed.
5. Resultant data are analyzed to uncover the categories that result in highest cost to the organization.
6. Plans are developed to modify the process with the intent of eliminating (or reducing the frequency of occurrence of) the class of errors and defects that is most costly.

The graph below illustrated the error that we expected for the project. They categorized in six fields:

Field	%	Reason
Logic	30	None of the member have any experience on doing a project in this size, nor the experience on the project in this degree of difficulties
Interface	25	We have 15+ interfaces that contain more than 400+ elements combined. And each element has average of 3+ functions behind it.
Data Handing	15	This project involved a lot of data accessing/storing, data flow between each interfaces. It's not easy to keep track of them. And the queries that the database used are pretty much outdated.
Error Checking	10	We practically proud of this field because for a job in this size and the experience we had in this field, we should get a higher error percentage. Since we have a very close contact with the client, and have done a exceptional work on research and study the old version. We believe we can keep the field under 10%
HW/SW	10	This mainly for the palm pilot integration section. We still haven't got any news from the client yet, regardless on the brand, price, model that want for the palm pilot. So it's very difficult to do any further research. And the palm pilot that we choice might be replace in few years or even shorter, or the government decide to go with different brand, or model.
Standard	10	Again, since we have no experience on such degree of project, so other errors might be uncover, but we been optimist here since we did a pretty good research on the project, and spent many times on designing phase before any actual coding


Logic
30%
Data Handing
15%
Error Checking
10%
Standard
10%
Interface
25%
HW/SW
10%

5.2 SPI Tasks and Responsibilities

Again, since we don't have a large team, the responsibilities of each team member will do doing SPI activities.

6.0 Software Configuration Management and Overview

During the time of the software development we will be making changes to our original plans. Software Configuration Management Plan is developed so that we can identify the changes, control the changes, making sure that the plan is implemented correctly and to making sure that we report the change to other team members and the clients. For further information on this topic, please go to the document titled "Software Configuration Management Plan".

7.0 SQA Tools, Techniques, Methods

We have described a lot tools, techniques, methods for SQA. Using voting system, close contact with the client, extensive detail design, and research on the subject to minimize the errors.
Different tools had been use for the SQA for this project. Using software review, problem tracking,
We try to follow the ISO 9001 Standard as our organizational structure, responsibilities, procedures, processes, and resources for implementing quality management. But the truth is, limited by the time and the size of the team; we are overlapping many of the functions to the same team member.

III Appendix

Book:

Pressman, Roger. Software Engineering, A Practitioner's Approach, 4th edition. 1997. McGraw-Hill, Inc.

URL

General Information on SQA:

1. http://www.asqc.org
2. http://www.quality.org/qc/homepage.html
3. http://hissa.ncsl.nist.giv/sw_assurance

Palm Pilot Research:

1. http://www.palmpilot.com
2. http://www.palmpilotware.com