**Quality Plan for Software Development Project**

**Prepared by:** Yaroslav Kechkin **Date:** 23 February 2012

The purpose of this Quality Assurance Plan is to define the techniques, procedures, and methodologies that will be used to assure timely delivery of the software that meets specified requirements within project resources. The use of this plan will help assure the following:

1. Software development, evaluation and acceptance standards are developed, documented and followed
2. Results of software quality reviews and audits will be given to appropriate management. This provides feedback as to how well the development effort is conforming to various development standards.
3. Test results adhere to acceptance standards.

No training of SQA personnel is anticipated as the person tasked is the author of the SQA plan. Each member of the software development team will be given a copy of the Software Development Plan (SDP) and Software Management Plan (SMP). SQA ensures this by interviewing each software developer within one month of their beginning to work on the project.

SQA will generate and maintain an Audit Schedule. Audits will occur at the end of each development phase as indicated in the SMP. The results of audits will be discussed with the individual most responsible for the production of the deliverable. SQA will perform random and unannounced audits to ensure the corrective actions agreed to during the Scheduled Audits are being followed. The results of audits will be discussed with the individual most responsible for the production of the deliverable. Audits performed shall include examination of both internal and external software deliverables. The SQA function will be audited by the Project Manager prior to any project review in which software is involved. The results of the audit will be available at the project review. Checklists will be used. Audit reports, and recommended corrective actions generated by SQA will be brought to the attention of the individual most responsible for producing the software deliverable. Corrective action will be recommended and reviewed with the individual and SPM. The results of audits of the SQA function will be kept by the Project Manager.

**Software Development**

SQA will review all deliverable software documentation including software plans. Review checklists will be used to review these documents. These reviews will help ensure that documentation is in compliance with applicable plans and procedures. The essential software documentation should include:

1. Software requirements
2. User documentation
3. Analysis documentation

Software documentation must be based on some published convention such as found in IEEE Software Engineering Standards.

1. A high level language shall be used except when approved by SPM.
2. Each method, function and class will be identified with their own comment header. The contents of the header should identify the purpose and any assumptions the user or caller must be aware of.
3. Coding documentation will, at a minimum, describe reasons for code branching and a description of each variable name at their point of memory allocation.
4. Naming conventions shall be used that clearly distinguish literal constants, variables, methods and class/object names. Class/object names should be nouns, methods should be verbs, and variables shall not be re-used for different purposes, except in trivial cases such as loop counts and indices. In addition, all names will contain at least 2 (two) characters to facilitate global pattern searches.
5. Dispatcher logic shall include a default clause, and loops shall include an escape clause except in forever loops.

(IEEE.org, 2012)

Released documents will be audited to ensure that published conventions were followed and that appropriate review comments were incorporated. SQA will audit deliverables between each software development life cycle phase until the software or subsystem is permanently incorporated in to the DPA hardware. The software life cycle phases will be defined in the SMP and the software products will be identified in the Software Development Schedule.

**Software Testing**

All code will be unit tested to ensure that the individual unit (class) performs the required functions and outputs the proper results and data. Proper results are determined by using the design limits of the calling (client) function as specified in the design specification defining the called (server) function. Unit testing is typically white box testing and may require the use of software stubs and symbolic debuggers. This testing helps ensure proper operation of a module because tests are generated with knowledge of the internal workings of the module.

There are two levels of integration testing. One level is the process of testing a software capability e.g. being able to encrypt a text. During this level, each module is treated as a black box, while conflicts between functions or classes are resolved. A second level of integration testing occurs when sufficient modules have been integrated to demonstrate a scenario e.g. encrypt and decrypt the text file. During this phase software is evaluated for performance for each function. Software documentation is reworked if necessary.

Testing Checklist (it.toolbox.com, 2008)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Unit Testing** | | | | |
| The unit test environment has been equipped with utilities for backup and recovery of multiple versions of the database. |  |  |  |  |
| The unit test environment has been kept up-to-date with the most current version of each release of the software and the database. |  |  |  |  |
| The test cases documented in the Unit Test Plan have been executed. |  |  |  |  |
| If a code error was detected, the demotion/promotion procedures documented in the Standards and Procedures Manual were followed. |  |  |  |  |
| A Promotion Request form has been completed for each module which has achieved the Unit Test Plan expected results. |  |  |  |  |
| The Build Status in the automated software item directory has been set to "Unit Tested" for each module which has passed unit testing. |  |  |  |  |
| **Integration Testing** | | | | |
| The Integration Test Environment has been established as defined in the Standards and Procedures Manual and in the Integration Test Plan. It is a separate test environment. |  |  |  |  |
| The integration test environment has been equipped with utilities for backup and recovery of multiple versions of the database. |  |  |  |  |
| The integration test environment has been kept up-to-date with the most current version of each release of the software and the database. |  |  |  |  |
| The personnel who conducted Integration Test are different from those who coded and unit tested the modules. |  |  |  |  |
| The test cases documented in the Integration Test Plan have been executed. |  |  |  |  |
| Documentation of test results has been maintained as stipulated in the test plan. |  |  |  |  |
| Fault Reports have been generated and classified for detected errors. |  |  |  |  |
| An Integration Test Execution Log has been maintained. |  |  |  |  |
| Corrected modules have been re-unit tested before being promoted to the Integration Test environment. |  |  |  |  |
| A Promotion Request form has been completed for each module which has achieved the Integration Test Plan expected results. |  |  |  |  |
| The Build Status in the automated software item directory has been set to "Integration Tested" for each module which has passed Integration Testing. |  |  |  |  |

**Six Sigma: DMAIC**

“DMAIC is the five-step approach that makes up the Six Sigma tool kit, and its sole objective is to drive costly variation from manufacturing and business processes.  The five steps in DMAIC are   Define, Measure, Analyze, Improve, and Control” (DMAICTools.com, 2012). Each step in software development life cycle can be related to a step in the DMAIC process.

1. Customer Requirements: get customer input, define objectives, measurable, and delivery dates. This can be viewed as the Define step of the DMAIC process.
2. Project Plan: assign resources, create checkpoints and test plans, update release date. This step corresponds to the Measure step in DMAIC process.
3. Execute plan: create software, perform testing and debugging, adjust plan and release the product. This corresponds to the Measure and Analyze steps.
4. Review Project: what was done right and wrong, and what needs to be improved. This is the Improve Control of the DMAIC process.

The process goes in the circle creating constant quality improvement for the software development project.

**References**

DMAICTools.com (2012). Six Sigma Tools: The DMAIC Process. Retrieved from <http://www.dmaictools.com/>

IEEE.org (2012). IEEE Software Engineering Standards. Retrieved from <http://www.ieee.org/portal/innovate/products/standard/ieee_soft_eng.html>

IT.Toolbox.com (February 9, 2008). Development Testing. Retrieved from <http://it.toolbox.com/blogs/enterprise-solutions/development-checklist-22370>