Software Project Audit Process

Version 1.2

Information and Communication Technology Agency of Sri Lanka

July, 2013

Revision History

Date	Version	Description	Author
14/12/2010	0.1	Initial Documentation	Erandi C Hettiarachchi
16/12/2010	0.1.1	Updated	Erandi C Hettiarachchi
02/09/11	0.1.2	Updated	Erandi C Hettiarachchi
15/09/2011	0.2	Updated with Payment milestones	Erandi C Hettiarachchi
15/11/2011	1.0	Updated with SPA process flow, guideline for review of artifacts and project progress calculation	Erandi C Hettiarachchi
23/03/2012	1.1	Updated payment milestones with OAT, Changed the documentation name from MSQA to Software Project Audit Process	Erandi C Hettiarachchi
08/08/2012	1.1.1	Changed the terminology - “SQA” to “SPA”	Erandi C Hettiarachchi
13/09/2012	1.2	Updated Section 1.1,1.2, Modified Section 2.1- Definition of RCI Index Added new section - Section 4.3 explains guidelines for increase defect severity Update Figure-1.0, Figure-3.0	Erandi C Hettiarachchi
23/07/2013	1.2	Reviewed version	Erandi C Hettiarachchi

Table of Contents

1. Introduction

1.1 Purpose

Purpose of this document is to describe the Software Project Audit Process which is capable of auditing and ensuring the quality of different activities carried out throughout a software project life-cycle. The main purpose of this process is to provide much higher level of confidence in the quality of the deliverables received by the client from the developer. The quality level of the audited activity is presented using a measurement technique called metrics.

The process should be followed by both the development team and the Software Project Audit team to derive their own metrics to measure the quality status of a software product in its life cycle. Eventually, the trend analysis of such metrics can be used to identify any potential project issues or failures and to come up with solutions.

This document explains several guidelines which can be used within the audit process for project progress calculation and mapping payment milestones with project deliverables or and project artifact reviews to effectively manage the project.

Further, the document contrasts the Software Project Audit process from typical software development life cycle and illustrates how it has been automated by integrating several testing tools and testing methodologies as well as embedding best industry standards.

1.2 Scope

Scope of this document is to provide an insight about the Software Project Audit Process, importance of metrics, analysis of metrics, automated process of metric generation, skills required to generate certain metrics, guideline for project progress calculation, guideline for mapping payment milestones with deliverables and guideline for Review of Project artifacts.

1.3 Definitions, Acronyms and Abbreviations

Acronym	Definition
AQI	Architecture Quality Index
AD	Architectural Design
CQI	Code Quality Index
DD	Defect Density
DQI	Design Quality Index
DSI	Defect Severity Index
ISI	Issue Severity Index
PERI	Project Execution Readiness Index
RCI	Requirement Clarity Index
SPA	Software Project Audit
SR	Software Requirement
TTEI	Tasks Tracking Efficiency Index
TR	Transfer
UAT	User Acceptance Test
OAT	Operational Acceptance Test

2. Process Overview

It is often said that if something cannot be measured, it cannot be managed or improved. There is immense value in measurement, but you should always make sure that you get some value out of any measurement that you are doing.

What is a Metric?

It is a standard of measurement which can be used to measure the software quality. It gives a confidence in the software product. They are typically the providers of the visibility of the software product you need.

Why Measure?

When used appropriately, metrics can aid in software development process improvement by providing pragmatic, objective evidence of process change initiatives. Although metrics are gathered during the test effort, they can provide measurements of many different activities performed throughout a project. In conjunction with root cause analysis, test metrics can be used to quantitatively track issues from points of occurrence throughout the development process. In addition, when metrics information is accumulated, updated and reported on a consistent and regular basis, it ensures that trends can be promptly captured and evaluated.

What to Measure?

When considering the metric driven process, it can be divided into two parts. The first part is to collect data, and the second is to prepare metrics/charts and analyze them to get the valuable insight which might help in decision making. Information collected during the software development process can help in:

- Finding the relation between data points
- Correlating cause and effect
- Input for future planning

Normally, the metric driven process involves certain steps which are repeated over a period of time. It starts with identifying what to measure. After the purpose is known, data can be collected and converted into the metrics. Based on the analysis of these metrics appropriate action can be taken, and if necessary metrics can be refined and measurement goals can be adjusted for the better. Data presented by Development/testing team, together with their opinion, normally decides whether a product will go into client or not. So it becomes very important for Development team/test teams to present data and opinion in such a way that data looks meaningful to everyone, and decision can be taken based on the data presented. Every software project should be measured for its schedule and the quality requirement for its release. There are lots of charts and metrics that we can use to track progress and measure the quality requirements of the release. In Figure 1.0 shows some of main metrics which can be derived at specific level of the software development life-cycle.

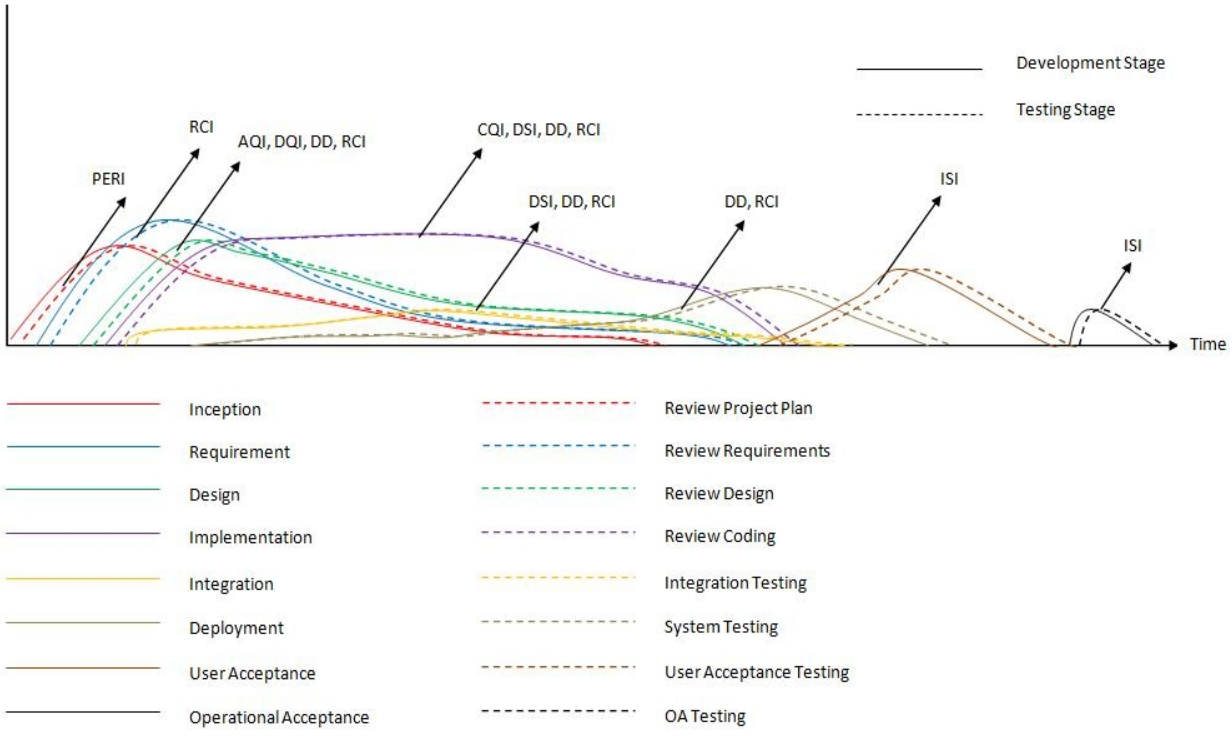


Figure 1.0 - Various Metrics derived at different levels of SD process

2.1 Metrics in Brief

Metric	Purpose
Project Execution Readiness Index (PERI)	This Proposed index at requirements stage is derived based on quality of the documents involve with this phase. The main Documents involve in this phase are; * User Requirements Document * Acceptance test plans * Project management plan for the SR phase * Configuration management plan for the SR phase * Verification and validation plan for the SR phase * Quality assurance plan for the SR phase When reviewing , reviewers can verify the document by checking its content with a checklist. Each of these content in a checklist is categorized under their Severity to the System. All defects in those contents should be logged in a defect tracking system. Finally, index can be derived as; Weighted average of the total number of Open Issues in the product detected till date against all categories (Blocker (B), Critical (C), Major (Ma), Normal (N), Minor (Mi), Trivial(T)). Metric: $\frac{(B*162 + C*54 + Ma*18 + N*6 + Mi*2+ T)*10}{\text{Total weight } (162+54+18+6+2+1)}$ Note: Can be calculated based on the review cycles
Requirements Clarity/Change Index(RCI)	This index measures following two criteria relevant to requirements 1. Requirements Clarity This is the proposed index is at Specification Stage which should indicate how well each member of the Software development team comprehend the requirements and also indicates How well the requirements are cleared for Software Development Team.

	2. Requirement Changes Requirement changes may be arisen at any stage of a project. Therefore, this index should be continued till UAT phase of a project and all the requirement changes arisen during that period should be captured under this index. The index indicates, weighted average of the total number of Open Issues in the product detected till date against all categories (Blocker (B), Critical (C), Major (Ma), Normal (N), Minor (Mi), Trivial(T)). Metric: $\frac{(B*162 + C*54 + Ma*18 + N*6 + Mi*2+ T)*10}{\text{Total weight } (162+54+18+6+2+1)}$ Note: Can be calculated based on the review cycles.
Architectural Quality Index (AQI)	Testing indicator for Architectural design level. The main documents of the AD phase are; *Architectural Design Document (ADD); *Software Project Management Plan for the DD phase (SPMP/DD) *Software Configuration Management Plan for the DD phase (SCMP/DD) *Software Verification and Validation Plan for the DD Phase (SVVP/DD) *Software Quality Assurance Plan for the DD phase (SQAP/DD) *Integration Test Plan (SVVP/IT) When reviewing , reviewers can verify the document by checking its content with a checklist. Each of these content in a checklist is categorized under their Severity to the System. All defects in those contents should be logged in a defect tracking system. Finally, index can be derived as; Weighted average of the total number of Open Issues in the product detected till date against all categories (Blocker (B), Critical (C), Major (Ma), Normal (N), Minor (Mi), Trivial(T)). Metric: $\frac{(B*162 + C*54 + Ma*18 + N*6 + Mi*2+ T)*10}{\text{Total weight } (162+54+18+6+2+1)}$

	Note: Can be calculated based on the review cycles
Design Quality Index (DQI)	This is the Index proposed at Detailed Design Level. Should define a quality index (DQI) to measure and evaluate the quality of the Detailed Design based on the quality of the documents involve with the Detailed Design phase. The main documents of the AD phase are the; *Detailed Design Document (DDD) *Software User Manual (SUM) *Software Project Management Plan for the TR phase (SPMP/TR) *Software Configuration Management Plan for the TR phase (SCMP/TR) *Software Quality Assurance Plan for the TR phase (SQAP/TR) *Acceptance Test specification (SVVP/AT) When reviewing , reviewers can verify the document by checking its content with a checklist. Each of these content in a checklist is categorized under their Severity to the System. All defects in those contents should be logged in a defect tracking system. Finally, index can be derived as; Weighted average of the total number of Open Issues in the product detected till date against all categories (Blocker (B), Critical (C), Major (Ma), Normal (N), Minor (Mi), Trivial(T)). Metric: $\frac{(B*162 + C*54 + Ma*18 + N*6 + Mi*2+ T)*10}{\text{Total weight } (162+54+18+6+2+1)}$ Note: Can be calculated based on the review cycles
Code Quality index (CQI)	- Indicates how well the software codes are written and maintained. - To be derived using considering multiple aspects. This will be decided in project execution. - Index can be derived as; Weighted average of the total number of Open Issues in the product detected till date against all categories (Blocker (B), Critical (C), Major (Ma), Normal (N), Minor (Mi), Trivial(T)).

	Metric: $\frac{(B*162 + C*54 + Ma*18 + N*6 + Mi*2+ T)*10}{\text{Total weight } (162+54+18+6+2+1)}$
Defect Density (DD)	- Number of defects per unit size of the application (KLOC) - Calculated end of each drop cycle. - The Number of Known Defects is the count of total defects identified against a particular software entity, during a particular time period - Size is a normalizer that allows comparisons between different software entities (i.e modules, releases, products). Size is typically counted either in Lines of Code or Function Points.
Defect Severity Index (DSI)	- Indicates application stability - Weighted average of the total number of Open Defects in the product detected till date against all categories (Blocker (B), Critical (C), Major (Ma), Normal (N), Minor (Mi), Trivial(T)). Metric: $\frac{(B*162 + C*54 + Ma*18 + N*6 + Mi*2+ T)*10}{\text{Total weight } (162+54+18+6+2+1)}$ Note: Calculated weekly and delivered by drop
Issue Severity Index (ISI)	During the User Acceptance Test(UAT) time issues can be arisen. All those issues should be logged in UAT documentation as well as in the bug tracking System. - Weighted average of the total number of Open issues in the product arisen during the UAT period against all categories (Blocker (B), Critical (C), Major (Ma), Normal (N), Minor (Mi), Trivial(T)). Metric: $\frac{(B*162 + C*54 + Ma*18 + N*6 + Mi*2+ T)*10}{\text{Total weight } (162+54+18+6+2+1)}$
Defect Category	An attribute of the defect in relation to the quality attributes of the product. Quality attributes of a product include functionality, usability, documentation, performance, installation, stability ,compatibility , internationalization etc. This metric can provide insight into the different

	quality attributes of the product. This metric can be computed by dividing the defects that belong to a particular category by the total number of defects.
Defect Cause Distribution Chart	This chart gives information on the cause of defects.
Defect Distribution Across Components	This chart gives information on how defects are distributed across various components of the system.
Defect Finding Rate	This chart gives information on how many defects are found across a given period. This can be tracked on a daily or weekly basis.
Defect Removal Efficiency	The number of defects that are removed per time unit (hours/days/weeks). Indicates the efficiency of defect removal methods, as well as indirect measurement of the quality of the product. Computed by dividing the effort required for defect detection, defect resolution time and retesting time by the number of defects. This is calculated per test type, during and across test phases.
Effort Adherence	As % of what is committed in contract. Provides a measure of what was estimated at the beginning of the project vs. the actual effort taken. Useful to understand the variance (if any) and for estimating future similar projects.
Number of Defects	The total number of defects found in a given time period/phase/test type that resulted in software or documentation modifications. Only accepted defects that resulted in modifying the software or the documentation are counted.
Review Efficiency	# of defects detected /LOC or pages reviewed per day
Test Case Effectiveness	The extent to which test cases are able to find defects. This metric provides an indication of the effectiveness of the test cases and the stability of the software. Ratio of the number of test cases that resulted in logging defects vs. the total number of test cases.
Test Case Execution Statistics	This metric provides an overall summary of test execution activities. This can be categorized by build or release, module, by platform (OS, browser, locale etc.).
Test Coverage	Defined as the extent to which testing covers the product's complete

	functionality. This metric is an indication of the completeness of the testing. It does not indicate any thing about the effectiveness of the testing. This can be used as a criterion to stop testing. Coverage could be with respect to requirements, functional topic list, business flows, use cases, etc. It can be calculated based on the number of items that were covered vs. the total number of items.
Test Effort Percentage	The effort spent in testing, in relation to the effort spent in the development activities, will give us an indication of the level of investment in testing. This information can also be used to estimate similar projects in the future. This metric can be computed by dividing the overall test effort by the total project effort.
Traceability Metric	Traceability is the ability to determine that each feature has a source in requirements and each requirement has a corresponding implemented feature. This is useful in assessing the test coverage details.
Scope Changes	The number of changes that were made to the test scope (scope creep). indicates requirements stability or volatility, as well as process stability. Ratio of the number of changed items in the test scope to the total number of items
Task Tracking Efficiency Index (TTEI)	This index indicates the average time taken to attend to general project tasks. TTEI = $\frac{\Sigma \text{ Time taken to attend task}}{\Sigma \text{ open task}}$

Table 1.0 – Metrics

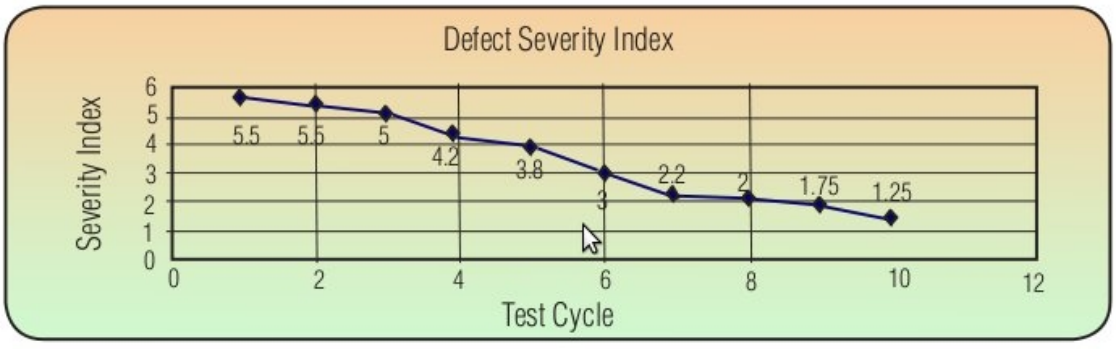
2.2 Metrics Analysis

Much as the time is spent gathering or maintaining metrics, enough time should be spent to review and interpret on a regular basis throughout the test effort, particularly after the application is released into production. During review meetings, the project team should closely examine all available data and use that information to determine the root cause of identified problems. It is important to look at several metrics, as this will allow the project team to have a more complete picture of what took place during a test.

Let's assume that as part of the SPA Process, the following metrics are collected by the SPA team.

Metric	Purpose
Defect Severity Index	Weighted average index of the Severity of defects. A higher severity defect gets a higher weight. S1 is a show stopper, S2 is high severity, S3 is medium & S4 is low. Ideally, this should slope down as test cycles progress.

For instance, if the test team has generated the following metrics:



Looking at the graphs one can safely deduce the followings;

Defect Severity Index Trend:

What does the graph indicate? The defect severity index is sloping down consistently. This indicates an increasingly favorable trend. As the test cycle progresses (from cycle 1 to cycle 10), the severity index is sloping which suggests increasing quality of the application (as lesser number of critical and high severity defects are being reported).

This is what it could mean: While a fall in the defect severity index is definitely a good trend, looking at this index in isolation could be misleading. Following factors need to be considered in order to have a meaningful analysis.

Number of defects logged - let us consider an example where the test team executed two cycles of testing (assuming other things as constant). The number of defects logged against each of these cycles along with the calculated severity index is shown below.

Number of Defects		
Defect Severity	**Cycle 1(# of defects)**	**Cycle 2(# of defects)**
s1	5	5
s2	10	15
s3	50	30
s4	100	100
Severity Index	1.52	1.50

At first thoughts, when we compare cycle 1's Severity Index with cycle 2's Severity Index, cycle 2 looks to be favorable (as the severity index is lower). If you go into the details of the number of defects logged and their severity, the picture turns out to be the opposite. While the total number of Severity 1 and Severity 2 defects for cycle 1 is 15, the number of Severity 1 and Severity 2 defects for cycle 2 is 20. In terms of quality, cycle 1 is better than cycle 2 as cycle 1 has lesser number of high severity defects (though the total number of defects logged in cycle 1 is more than cycle 2 defects and the severity index is greater than cycle 2 severity index). Test coverage has a similar impact. A lower test coverage coupled with reducing severity index would not be a healthy trend.

Severity of Defects		
Defect Severity	**Cycle 1(# of defects)**	**Cycle 2(# of defects)**
s1	4	0
s2	4	0
s3	42	75
s4	27	2
Severity Index	1.81	2.03

- Defect Severity - let's consider another example where the test team executed two cycles of testing (assuming other things as constant). The severity of defects logged against each of these cycles along with the calculated severity index is shown below.

Looking at the severity index, it looks like cycle 1 is better than cycle 2 (as the severity index is low for cycle 1 compared to cycle 2). However, cycle 2 is better than cycle 1 as total number of Severity 1 and Severity 2 defects is zero compared to a total of 8 severity 1 and severity 2 defects of cycle 1. Just because the severity index is low, do not believe the quality of the application is better than the earlier cycle.

3. Process Automation

In following section describes about the testing methodologies, process and tools to be used while automating the typical software development life-cycle in order to deriving the metrics.

3.1 Testing Methodology

According to the Automated testing process, every development activity is mirrored by a test activity. The testing process follows a well-proven testing methodology called W-model. Following Figure-2.0 explains, the way of testing activities of W-model involve with the standard software development life-cycle.

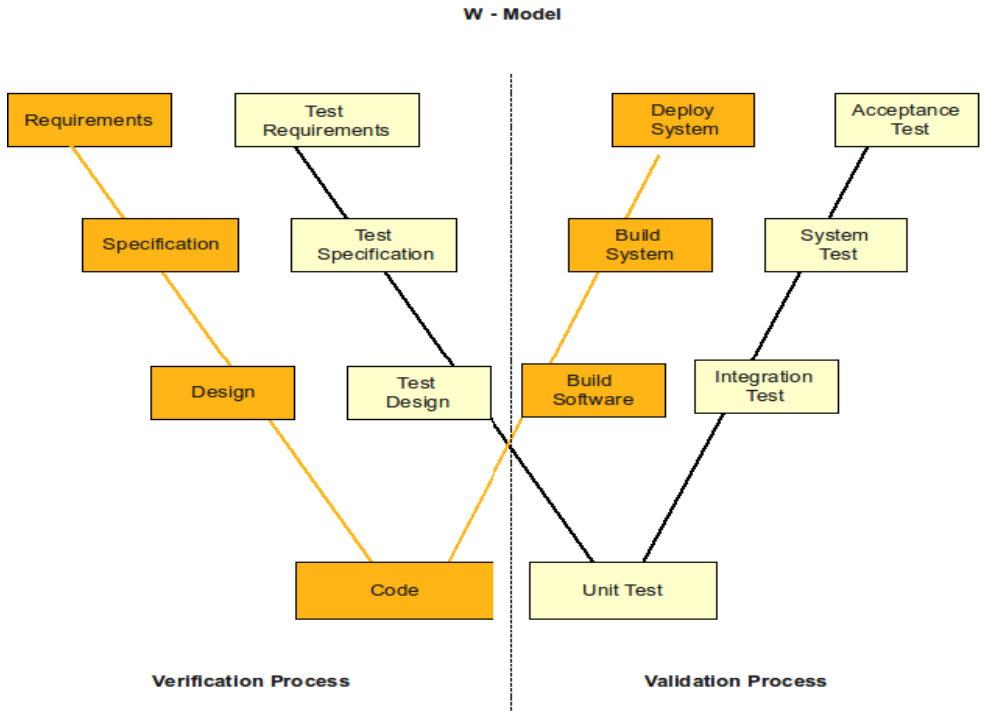


Figure 2.0 – The W-model

While the execution of the project, either developers or SPA team can generate the related metrics.

3.2 Skills required to generate Metrics

During the different stages of a software project, several roles and parties will be involve with development, reviewing and testing activities. In Figure 3.0 shows the different stages of a software project, the main activities which should perform during those stages, the roles/parties should involve and the metrics which derive and maintain in those stages.

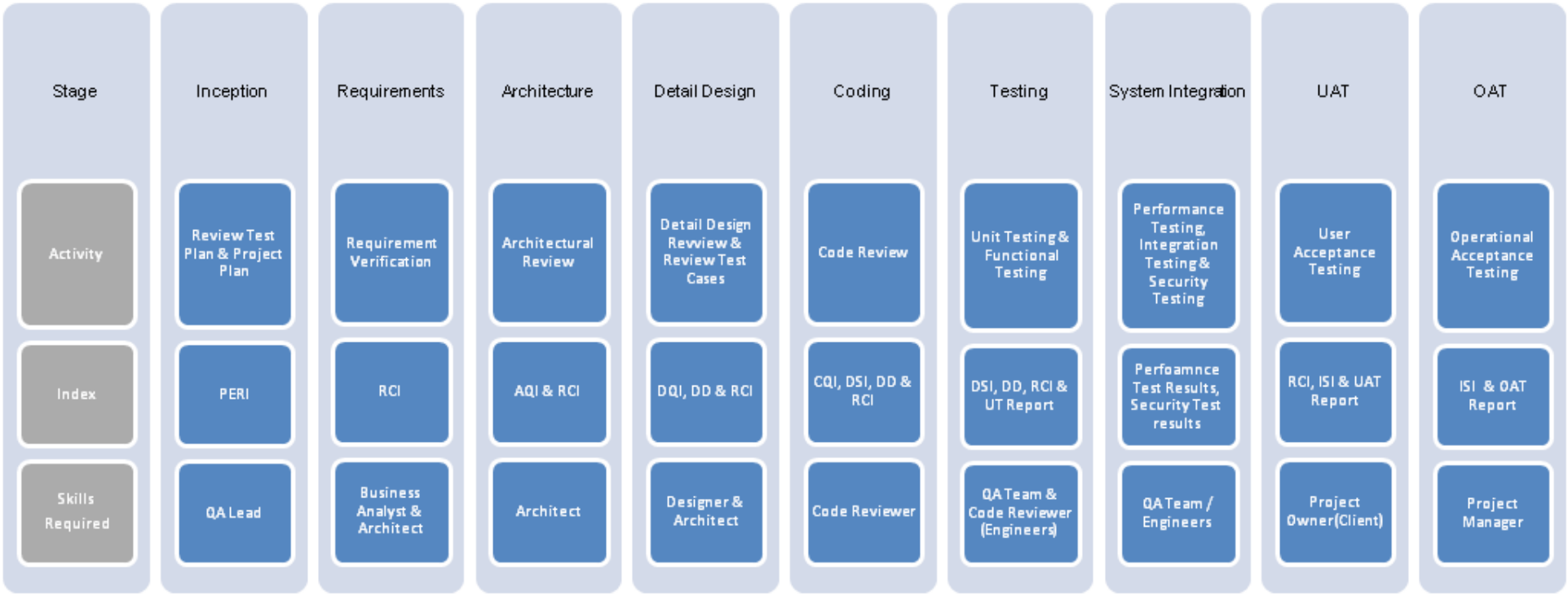


Figure 3.0 - Skills required to generate Metrics

3.3 Process of Setting-up a Metric

The Figure-4.0 explains the life-cycle of a Metric or the process involved in setting up the metrics:

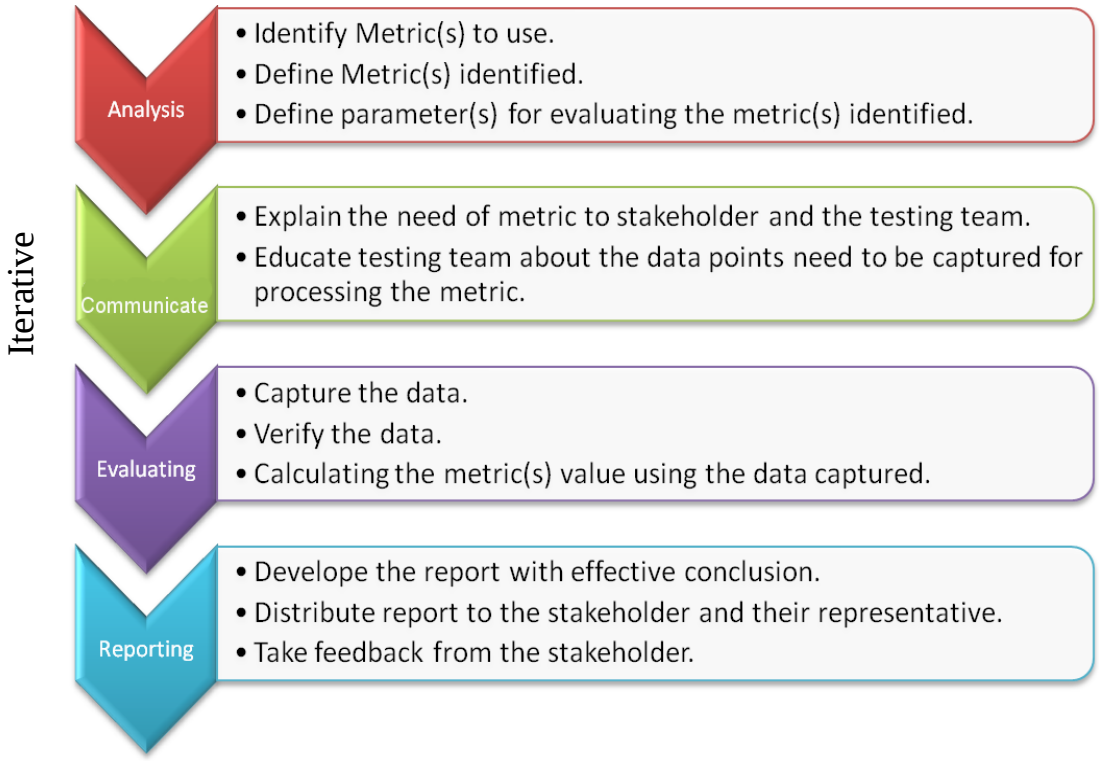


Figure 4.0 - Metrics Life-Cycle

When implementing this process, several testing tools and techniques will be used along with the automated testing process in order to generating, maintaining and evaluating the metrics derived at specific level of the Software development life-cycle.

3.4 Integration of testing tools/process

Below you find a list of tools /process which will be used when automating the typical SD life-cycle suits to the Software Project Audit Process.

- **Fagan inspection -** Fagan Inspection defines a process as a certain activity with a pre-specified entry and exit criteria. Activities for which Fagan Inspection can be used are:
    - Requirement specification
    - Software/Information System architecture (for example DYA)
    - Programming (for example for iterations in XP or DSDM)
    - Software testing (for example when creating test scripts)
- **Cruise Control –** It is both a continuous integration tool and an extensible framework for creating a custom continuous build process. It includes dozens of plug-ins for a variety of source controls, build technologies, and notifications schemes including email and instant messaging. A web interface provides details of the current and previous builds.
- **Bug-zilla -** It is a Web-based general-purpose defect tracking and testing tool.
- **SVN -** It is a revision control system which use Subversion to maintain current and historical versions of files such as source code, web pages, and documentation.
- **Git -** Git is a free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- **SCM -** For Configuration identification and Identifying configurations, configuration items and baselines. Also for Configuration control ,Configuration status accounting and Configuration auditing

3.5 Displaying Metrics –The Dashboard

The Dashboard is the interface to help project teams to visualize their project statuses by several indexes. And also it could be used to displaying the test results of specific tests carried by the SPA team who responsible for the given project. As an example; in Figure 5.0 displays the current status of the project with its estimated effort against the predicted effort.

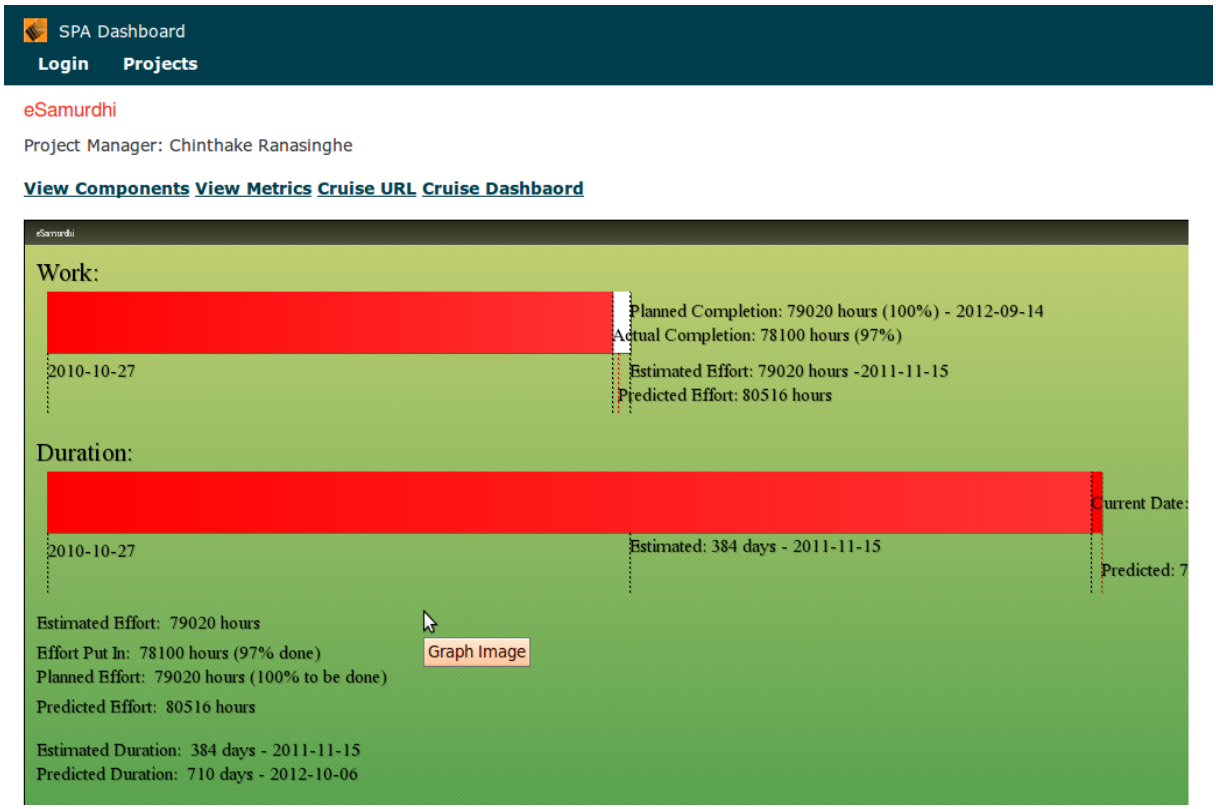


Figure 5.0 – SPA Dashboard view

4. Guideline for review of Artifacts

"SPA Process is not an QA process. It is an Audit of the quality of the deliverables at the client site."

4.1 Industry guideline on Defects

Deliverable Type		Issues per 1 KL
Documentation	**Source code**	
After writing	After coding	20
After self reviewing	After unit testing	8
After peer review	After QA	3-4 or less

4.2 Acceptance / Rejection of Deliverables

An acceptable client delivery should be 3-4 defects without any blocker, critical or major issues. If there are any known issues with the delivery, the vendor should prior inform to ICTA & the SPA team.

Any client deliverable will be prone to rejection under following scenarios.

In a client deliverable if there is;

- One blocker issue
- Two critical issues
- One critical and 4 major issues
- Eight major issues

In such situation, the SPA team should stop the review and bring that to the attention of ICTA.

4.3 Increasing Defect Severity

It is required to increase the severity of a defect by the SPA team, if the defect was kept open for more than 1 month. The Defects with "Differed" status should not be changed.

NOTE: Automatic update of defect severity is to be added to the SPA dashboard in near future. Till then, SPA team needs to update the defect severity manually.

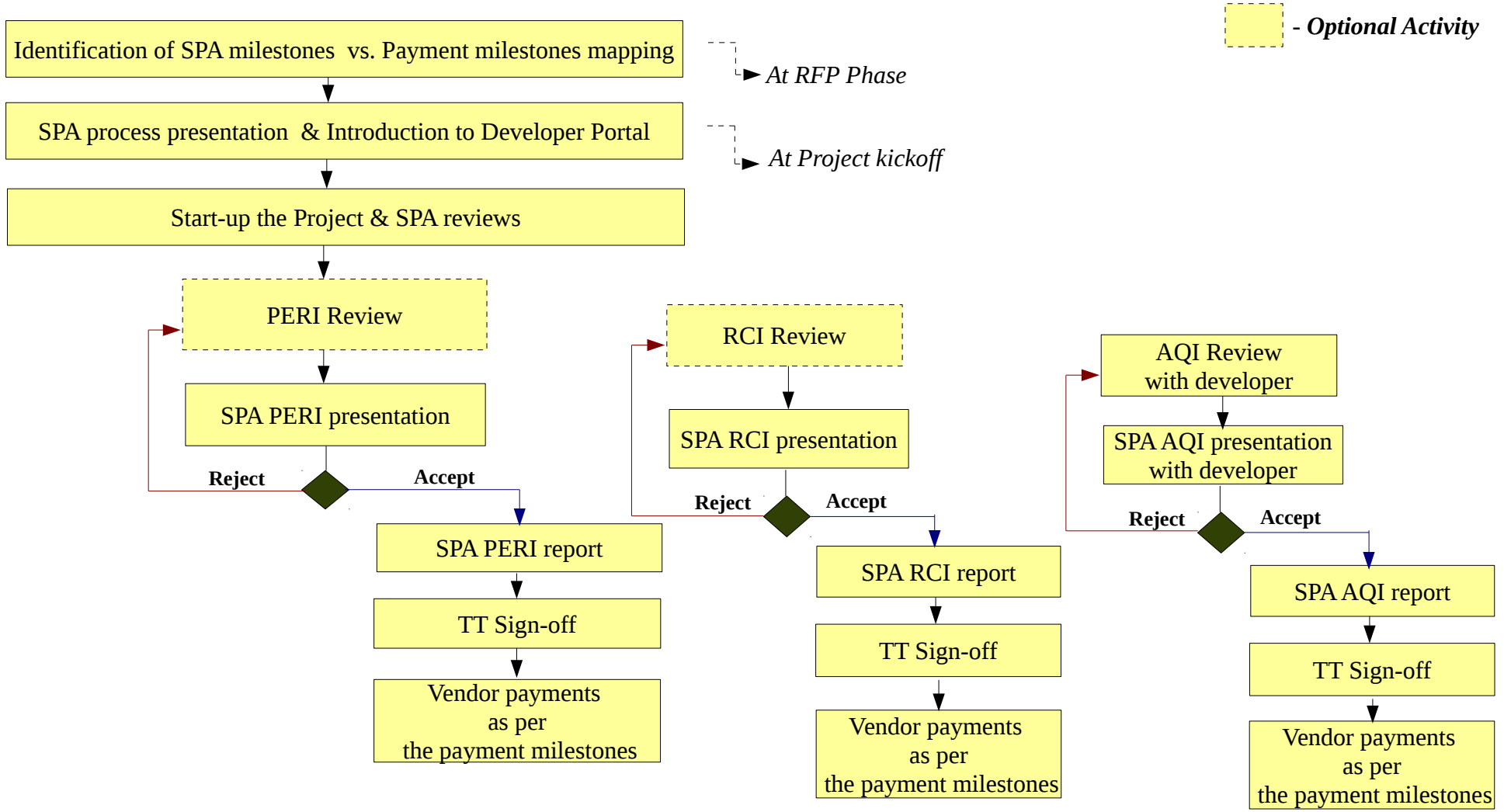
5. Payment Milestones

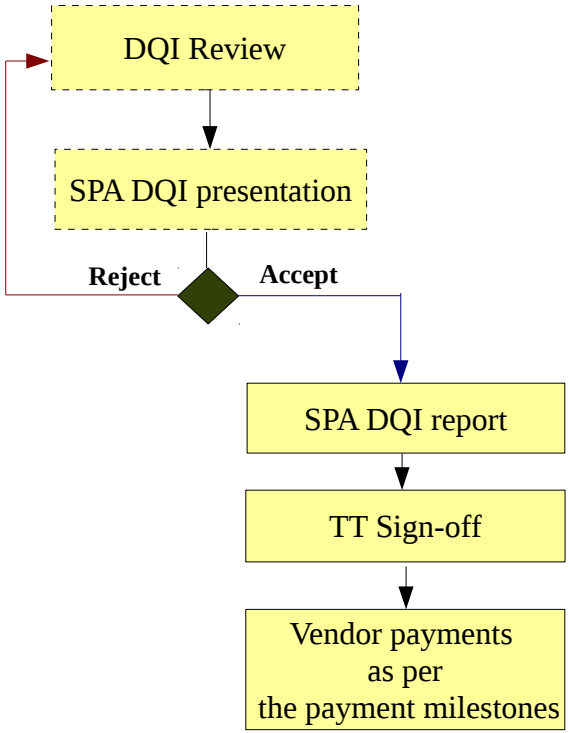
The metrics discussed above are directly linked with payment milestones of a project. In Table 2.0 describes the several output documentations needed to be provide by both vendor and the SPA team at such payment sign off point.

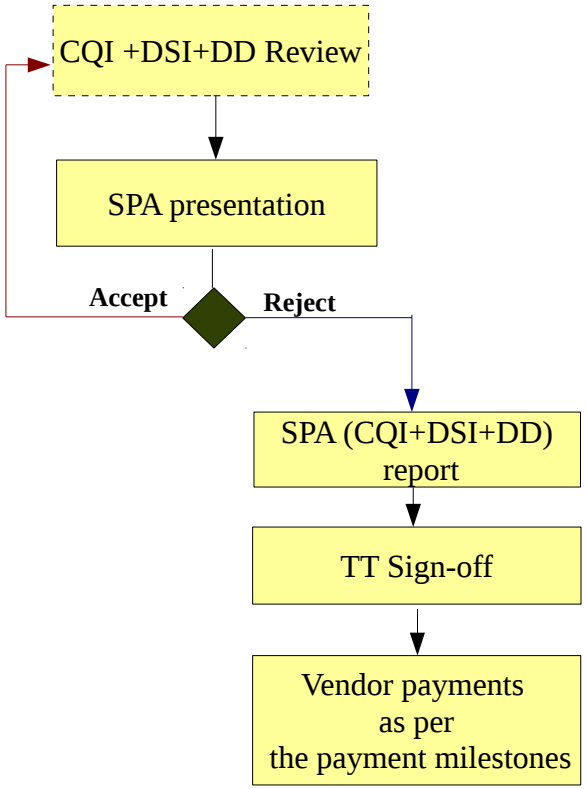
Payment Milestone	Index	Vendor Output	SPA Team Output
#1	PERI	• Project plan , Initial project schedule ,Test plan • Release management plan(this may can be covered in project schedule or test plan) • Project governance and the communication structure	• Project Execution Readiness Report
#2	RCI & AQI	• Software Requirement Specification report (RCI) • System Architecture document (AQI)– In this report should include the deployment architecture also.	• Requirement Clarity report • Architectural Quality report
#3 or Core Module wise	DQI & CQI	• Detailed design document for the Core modules (optional for the small projects)	• Design quality report • Code quality report
#4 Each iteration(code drop)	CQI,DSI,DD		• Software Quality report - In this report SPA team should include the CQI,DSI and DD observations.
#5	ISI	• User Acceptance Test (UAT) report - All the issues arisen at the UAT period should be logged under new chapter in UAT documentation.	• All the issues rescheduled in all above phases should be transferred to the UAT phase
#6 Final Payment (Successful run during the OAT period)	ISI	• Operational Acceptance Test (OAT) Report • All the issues raised during the UAT phase should be rescheduled to the OAT phase.	• No SPA team involvement. All the issues raised during the OAT phase + the rescheduled issues from UAT should be logged by the ICTA Project Manager.

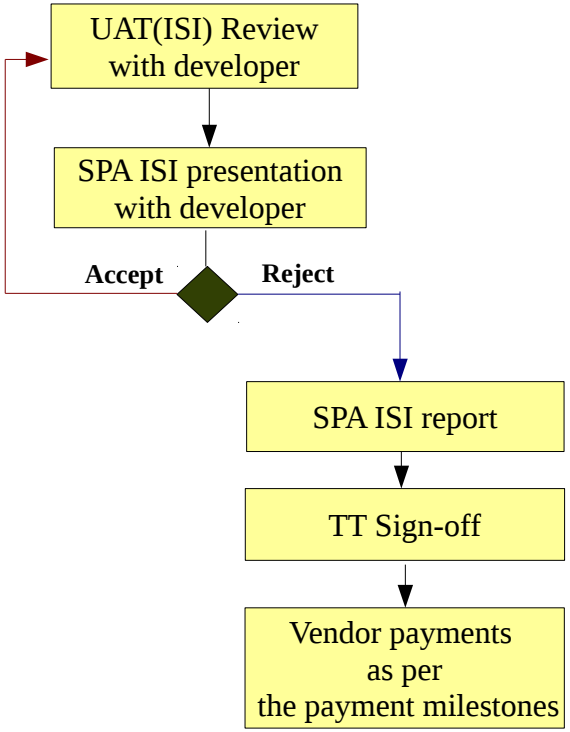
Table 2.0 – Payment milestones

6. SPA Process Flow









7. Guideline for Project progress calculation

In SPA process following methodology is used in the calculation of project progress.

- Any software project consists of several payment milestones. Each payment milestone consists of one or more deliverables.
- A payment milestone is an indication of some percentage of the project progress.
- Each deliverable indicates equal percentage completion of relevant payment milestone.
- At the beginning of a project, all the possible payment milestones and their deliverables should be identified and updated to the SPA Dash board.
- For monitoring purposes, progress of a deliverable is considered as the basic element of measurement when calculating project progress.
- The SPA process defines 05 steps life-cycle for a deliverable. Figure 6.0 illustrates those steps and a percentage completion for each steps.

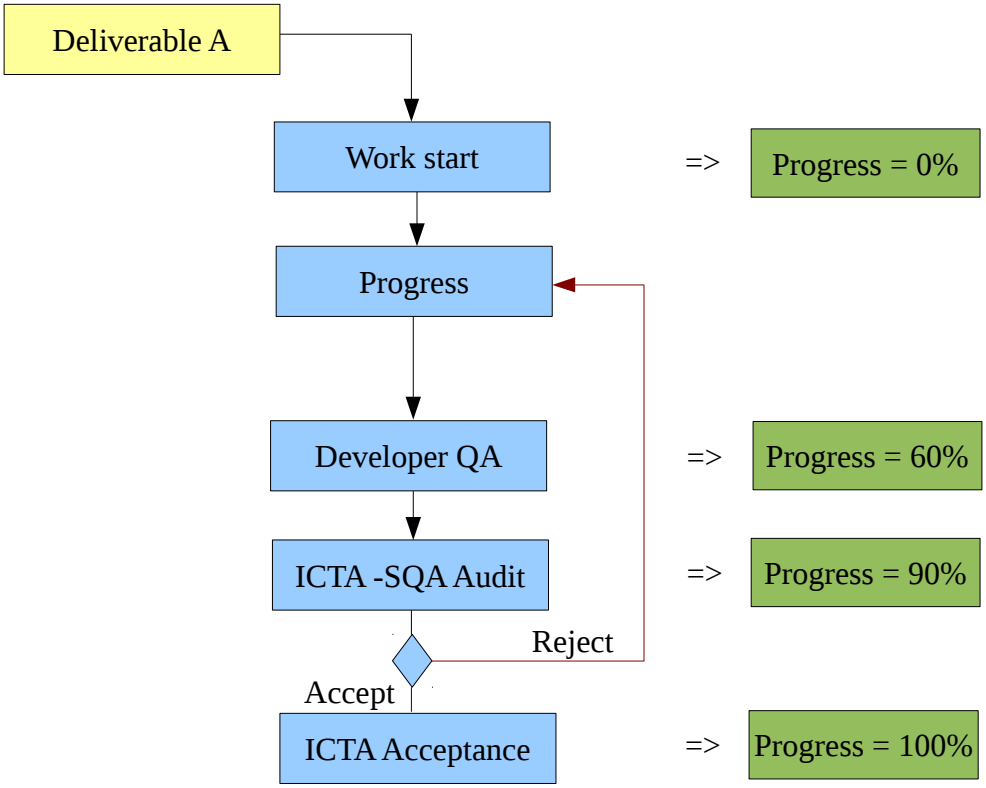


Figure 6.0 – Deliverable Life-Cycle

- Depending on the stage of a deliverable, its progress should be updated to the SPA Dashboard by the project manager.
- Finally, the progress of a payment milestone and the total project progress will be calculated by the system as follows;

Payment milestone Progress

Suppose a payment milestone which consists with **n** no. of deliverables called $\mathbf{D_1}$, $\mathbf{D_2}$, $\mathbf{D_3}$, … , $\mathbf{D_n}$. If the respective progresses of those deliverable are as $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$ **… ,** $\mathbf{P_n}$, then the total progress of the payment milestone can be calculated as follows;

Payment milestone Progress % = $\left(\sum_{i=1}^{n} \mathbf{P_i} \right) / \mathbf{n}$ **%**

Assumption : Each deliverable indicates equal percentage completion of a relevant payment milestone.

E. g:

Assume a project with 2 deliverables called D_1 and D_2 . The current statuses of those deliverables are as follows;

Deliverable	Current Status	Progress %
D_1	ICTA SPA Review	90 ($\mathbf{P_1}$)
D_2	Developer QA	60 ($\mathbf{P_2}$)

Then the progress of the payment milestone can be calculated as follows;

Total progress of the payment milestone = (90 + 60) / 2 %
= 75%

So it can be concluded that the payment milestone is 75% completed.

Total Project Progress

Suppose a project consists with **n** no. of payment milestones called $\mathbf{M_1}$, $\mathbf{M_2}$, $\mathbf{M_3}$, … , $\mathbf{M_n}$ and their respective weightages are as $\mathbf{W_1}$, $\mathbf{W_2}$, $\mathbf{W_3}$, … , $\mathbf{W_n}$. If the respective progresses of those payment milestones are as $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$, …, $\mathbf{P_n}$, then the total progress of the project can be calculated as follows;

Total project progress = $\mathbf{((P_1 * W_1) + (P_2 * W_2) + (P_3 * W_3) + \ldots + (P_n * W_n)) / 100\ \%}$

$$= \left(\sum_{i=1}^{n} P_i * W_i \right) / 100 \quad \%$$

E. g:

Consider a project with 5 payment milestones called M_1 , M_2 , M_3 , M_4 and M_5.

Payment Milestones	Weightage %	Progress %
M_1	10 (**W1**)	90 ($\mathbf{P_a}$)
M_2	20 (**W2**)	80 ($\mathbf{P_b}$)
M_3	40 (**W3**)	90 ($\mathbf{P_c}$)
M_4	25 (**W4**)	75 ($\mathbf{P_d}$)
M_5	5 (**W5**)	40 ($\mathbf{P_e}$)

then the total progress of the project can be calculated as follows;

Total progress of the project = (90*10 + 80*20 + 90*40 + 75*25 + 40*5)/100 %

= 81.75%

So it can be concluded that the project is 81.75% completed.

8. References

[1] . Fagan inspection , from Wikipedia, http://en.wikipedia.org/wiki/Fagan_inspection

[2] . W-model , http://sqa.fyicenter.com/FAQ/Software-Development-Models/

[3] . CruiseControl, http://cruisecontrol.sourceforge.net/