Title: TTAT Quality Management Plan	Authors: Treacy, Kern	Date: 3/19/2020
Document No: 688-TTAT-0012-MGMT		Version: 1.0

Telescope Time Allocation Tools

Quality Management Plan

Project: 688

PREPARED BY	ORGANIZATION
Bob Treacy, Jeff Kern	NRAO, PMD

APPROVALS	ORGANIZATION	SIGNATURE
Jeff Kern	NRAO SRDP, Project Director	
	NRAO SRDP AD	
	NRAO, PMD AD	
Tony Beasley	NRAO Director	

CHANGE RECORD

VERSION	DATE	SECTIONS	CHANGE DESCRIPTION
0.01	1/27/2020		Initial Draft
0.02	2/12/2020		Added Sec **Error! Reference source not found.** on Defect Prevention
0.03	3/10/2020	All	Major revision to all sections.
1.0	3/19/2020	All	Incorporated feedback from internal review.

Table of Contents

1 INTRODUCTION

In defining an actionable quality management plan for the development of the Telescope Time Allocation Tools project we are influenced by the definition given by Feigenbaum [RD01]:

> "Quality is a customer determination, not an engineer's determination, not a marketing determination, nor a general management determination. It is based on the customer's actual experience with the product or service, measured against his or her requirements -- stated or unstated, conscious or merely sensed, technically operational or entirely subjective."

and by Juran [RD02]:

> "The word quality has multiple meanings. Two of these meanings dominate the use of the word:
> 1. Quality consists of those product features which meet the need of customers and thereby provide product satisfaction.
> 2. Quality consists of freedom from deficiencies."

Based on these concepts we qualitatively define the total software quality as: Feature Completion minus defects identified by customers and incurred technical debt. With this qualitative measure of Software Quality, the tradeoffs between functionality, deficiencies (which we equate with defects) and technical debt are readily apparent.

This plan defines the metrics we will use to measure each of the terms in our quality equation and the cadence at which these metrics are measured. This plan is not prescriptive either in terms of absolute value targets for the metrics nor in the correct balance among the terms, instead it provides a framework to measure the current software quality, drive data-based decisions on improving software quality, and allow trend analysis of the quality of software products.

This plan is comprised of four Software Quality Management (SQM) categories:

- Quality Management Planning
- Software Quality Assurance (SQA)
- Software Quality Control (SQC)
- Software Process Improvement (SPI)

1.1 Applicable Documents

[AD01] SRDP Program Management Plan 530-SRDP-044-MGMT
[AD02] TTA Tools Project Management Plan 688-TTAT-003-MGMT
[AD03] TTA Tools Stakeholder Register 688-TTAT-0016-MGMT
[AD04] TTA Tools Concept 688-TTAT-002-MGMT
[AD05] DMS Software Development Processes

1.2 Reference Documents

[RD01] A. V. Feigenbaum, "Total Quality Control", McGraw-Hill, 1983.
[RD02] J.M. Juran, "Juran's Quality Control Handbook", McGraw-Hill, 1988.
[RD03] P. Bourque, R. Farley "Software Engineering Body of Knowledge (SWEBoK)", Ver 3, IEEE 2014.

[RD04] C. Jones, "Software Defect Removal Efficiency", www.ppi-int.com.

2 QUALITY MANAGEMENT PLANNING

The overall objectives for the TTA Tools Quality Management Plan are to assess and improve the quality and reliability of delivered software products. All software will have defects. A primary objective of this plan is to minimize defects that reach the production environment, therefore maximizing the user experience. A secondary goal is to limit and closely manage the technical debt incurred in order to complete and deliver a maintainable and extensible software system. The desired result is higher confidence and increased use of the radio astronomy software tools provided by NRAO to the astronomy community.

Specific Goals:

- Identify, measure, manage, and reduce the number of defects in developed code, server environments, and configuration of tools and deployments.
- Develop metrics to measure defects and track rate of occurrence
- Establish metrics and processes to manage technical debt
- Develop quality processes to collect and assess metrics
- Develop quality processes to mitigate defects (monitor, control, and minimize defects)
- Actively plan and manage process improvements across all quality processes
- Identify and assign roles and responsibilities associated with executing this plan

Defects and technical debt are costly and could be quantified as the "Cost of Poor Quality (COPQ)", incurred in the absence of quality processes. COPQ is frequently used to justify the investment in quality processes, but is a "catch 22" because unless the quality processes are in place and the COPQ is quantified, one cannot know in advance if the process cost can be justified.

3 SOFTWARE QUALITY ASSURANCE (SQA)

SQA includes both Product Assurance and Process Assurance. Process Assurance for software is much like other disciplines, ensuring that the stated processes are executed correctly. Product assurance refers to measuring and controlling the quality of the software systems under construction. SQA Processes include verification, validation, reviews, and audits; all of which apply to the product and the processes that produce the product. Metrics are the output of the SQA Processes. Terms that define the TTA Tools quality framework follow.

3.1 Feature Completion

Fundamentally users expect software systems to provide functionality that matches the problem the system is designed for. For the TTA Tools we define feature completion as the ratio of the number of completed capabilities during a specified period to the number that were planned to be completed during that period.

Software systems can fail to meet user expectations for reasons other than missed delivery milestones. In order to ensure that the software being delivered will address the intended problem the following steps are taken within the TTA Tools project:

- Stakeholder identification and engagement - Stakeholder groups were broadly identified in the Project Charter, individuals were identified within those groups, and included in the project Stakeholder Register [AD02]. Stakeholder engagement is managed under the TTA Tools Project Management Plan [SAD01] in the Communications Section.

- Requirements elicitation – A committee was convened and charged to develop the TTA Tools Concept [AD03] for redesign that would preserve current functionality and add needed capability to support future needs. Committee members represented all instruments to be supported with the redesigned TTA Tools, taking input from the User community, managers, scientists, and developers. Requirements decomposition and analysis has and will be performed while consulting key stakeholders.
- Design – A computer-based modeling environment is used to capture the design, allocate requirements to design elements, to facilitate validation of the architectural model, and to track verification and validation of the final implementation.
- Traceability – Requirements and traceability are maintained in the model as the "record of authority" on three levels Conceptual, System, and Implementation. Requirements are allocated to architectural components to assure all requirements are captured to the design. Verification and Validation matrices are built within the computer model to provide traceability between design, implementation, and test. A computer-based issue tracking tool is used to manage test procedures and progress, which is integrated with the model to complete the traceability.
- Enablers of Defects and Technical Debt – Defects frequently occur due to factors external to the developers and even to the project. Such factors include schedule pressure, unrealistic expectations, inconsistent or nonexistent management and engineering processes, poor communication regarding organizational vision and project objectives, and other factors. Managing expectations of all stakeholders and application of properly scaled management and engineering processes is necessary to minimize the impact of these external enablers. The TTA Tools project conducts a review following each development phase to capture lessons learned which includes the impact of external factors on the quality of deliverables.

In spite of the efforts above defects stemming from misunderstanding or overlooked scope may occur. In the SQA framework these errors will be accounted for either through the identification of technical debt (partially fulfilled functionality) or defects (usually identified as requirement deficiencies).

3.2 Identification of Defects

A defect is an unintended or undesired behavior that adversely affects the user experience. Nearly all software has defects. For the purposes of this quality process errors in design or concept that are detected and corrected prior to implementation are assumed to not impact the user experience and are excluded from the quality definitions. The following aspects of defects are defined in order to establish useful metrics for quantifying and controlling the defect rate experienced by the end user.

3.2.1 Defect Origins

Defects can be introduced anywhere in the product lifecycle from Requirements Capture through Divestment. For the purposes of measuring the rate of defect introduction all defects are defined to originate from one of six categories.

- **Requirements**: Unclear, unstated, or incorrect requirements lead the system to not perform as desired.
- **Architecture / Design:** Defects stemming from the structure or design of the system.
- **Implementation:** Defects originating in the implementation of the system, errors in coding leading to incorrect or undesired behaviors.
- **Configuration:** Defects originating in the configuration of the system in deployment (incorrect paths, parameters, etc).
- **Performance**: Defects encountered as quality of service defects, insufficiently responsive systems, or other origins relating to computing resource utilization.
- **Documentation:** Defects stemming from incorrect or missing documentation.

3.2.2 Detection Points

The "escape rate" or fraction of total defects found in production is a key indicator of overall software quality, and the efficacy of the testing process. Tracking the environment in which defects are detected allows determination of the total defect volume and the effectiveness of testing programs at each level. Although the perceived software quality from the user is only dependent on defects detected in production the cost of defects is strongly dependent upon where in the development cycle they are first identified. We will track the detection environment at four levels:

- **Implementation:** Identified during implementation.
- **Integration and Verification:** Identified during integration and verification activities.
- **Validation:** Identified during the software validation phase prior to production deployment.
- **Production:** Identified while the software is in production use.

It is worth noting that the determining factor is not the role of the individual identifying the defect but the point in the software lifecycle that the defect is first identified. Prior to the deployment of the MVP, there is no production version available, during this phase the Validation environment will be used to determine an "internal escape rate" to monitor software quality.

3.2.3 Severity

For the purposes of measuring software quality, the severity of defects will either be classified as blocker or non-blocker.

- **Blocker:** True blocker defects are rare to be detected in the production environment. These are defects which prevent the user from accomplishing their goal, and for which no reasonable workaround exists. Blocker level defects when detected in production should result in either revision to previous versions or an emergency patch deployment.
- **Non-Blocker:** Everything else**.**

Additional levels of severity as subjective at some level and are difficult to apply uniformly by all users, making interpretation of the resulting metrics more nuanced. Defects will still be categorized at various levels of severity for the purposes of prioritization and planning but will be consolidated to these two levels for reporting purposes.

3.3 Technical Debt

In software-intensive systems, technical debt consists of design or implementation constructs that are expedient in the short term but that set up a technical context that can make a future change more costly or impossible. Technical debt is a contingent liability whose impact is limited to internal system qualities- primarily, but not only, -maintainability and evolvability.

Technical debt may be incurred intentionally, but is often unintentional and only identified as debt in retrospect. In all cases capturing the cause of the debt, the impact, and the cost to address facilitates proper management and decision making. DMS is developing a process for capturing technical debt [AD05] which the TTA tools will leverages as part of the software quality management process.

3.4 Methodology

Total Quality Management includes many factors which cannot be quantified, and at the basest level reflects how users feel about the software product. Although some unquantifiable factors have a significant impact on the introduction of defects, realistic metrics are constrained to what can be measured. The objective for the TTA Tools SQA plan is to re-use existing processes to impose the minimum reporting burden on the project team while enabling the measurement and management of the overall software quality.

3.4.1 Feature Completion Metrics

Software development effort at NRAO is tracked through an issue reporting system. For planning purposes work within a development phase will be defined as a set of features to be delivered. Within DMS these features are further decomposed to tasks, and an estimated effort level associated with the tasks. Thus, two measurements of feature completion are readily available:

- **Fractional Feature Completion**: The number of features that were delivered (including validation) divided by the planned number of features. No credit is claimed for partially delivered features, although a feature that is delivered but incurs technical debt (unsupported use cases or development short cuts) should be included. This metric has the advantage of measuring what the user wants (features) but the disadvantage that all features are assumed to be of uniform importance and cost.
- **Delivered Schedule**: A second metric that accounts for the variance in size of the delivery features is the delivered schedule. Defined as the total estimated effort required for the delivered features, this provides a measure of the amount of effort actually delivered. Again, no credit should be claimed for partially completed features. The Normalized Delivered Schedule is calculated by divided the delivered schedule by the total effort estimated for all features[1].

3.4.2 Defect Metrics

The same tracking system used for task tracking is used to record identified defects. It will be modified to support the recording of the detection environment (implementation, verification, validation, or production) for all defects.

Each defect identified must either be repaired, mitigated, or rejected. A repair operation addresses the underlying cause of the issue in software and a modified version of the software removing the defect is released. Defects that are mitigated may require change of process or documentation to avoid the defect either on a temporary or permanent basis. If the mitigation is temporary the defect should be entered into the technical debt register for the project, if the mitigation is permanent no further action is necessary. Defects that are rejected require no further action.

Following Project Performance International [RD04] the **Defect Removal Efficiency** (DRE) is defined as the percentage of defects found and in each of the identified pre-production environments. Rejected defects should not be included in this calculation.

The number of defects found in production is assumed to depend on the duration of production use only weakly. To verify this assumption, and provide a normalized metric to compare across multiple releases the **Production Defect Rate** is defined by the number of defects identified per unit time.

3.4.3 Technical Debt Metrics

Technical debt is often invisible to the end user; thus, the project will rely on the DMS procedures to gauge the technical debt included in delivered software. DMS will maintain a technical debt register [AD0] for the TTA project and several metrics will be tracked:

- The **Total Technical Debt** in the system will be quantified by the number of entries in the technical debt register.
- The **Total Technical Debt Effort** is defined as the sum of the estimated effort to address all technical debt in the system. For debt which does not have an effort estimate it will be assumed to be equal to the average of those which do have effort estimates.

[1] A related metric is the ratio of the delivered schedule to the actual effort. Although useful for planning and scheduling this metric is not used as part of software quality management.

- The **Marginal Technical Debt** is the net increase or decrease in Total Technical Debt of the system from one delivery to the next.
- The **Marginal Technical Debt Effort** is the net increase or decrease in Total Technical Debt Effort of the system from one delivery to the next.

Technical debt is often not identified until well after the fact, in these cases on a best effort basis the technical debt will be assigned to the delivery in which the debt was incurred rather than when it is detected.

4 SOFTWARE QUALITY CONTROL (SQC)

Software quality metrics for current and previously released software versions are calculated as part of the lessons learned for each phase by the SQA Manager. All defects are reviewed, the origin is agreed by stakeholders and any defects not assigned a resolution are assigned one. The lessons learned meeting also serves as an audit of the process execution, checking with all stakeholders that the processes defined in the Project Management Plan [AD01] and Software Development Processes [AD05] are followed.

Metrics for previously released software is updated to include any defects identified in production or technical debt that as uncovered. The SQA Manager will produce a Software Quality Report describing the quality of current and previous releases. The Project Director is the primary audience for this report although it will be distributed to the Sponsor and Assistant Director of DMS as well.

No prescriptive actions are defined at this time based on the Software Quality Report, the objective is to improve software quality over time. Strategies and tactics for accomplishing this will be developed based on the trends and patterns revealed by the metrics.

4.1 ROLES AND RESPONSIBILITIES

Table 1 enumerates the roles within the quality management processes. The first column specifies the roles within the quality management process and the third column allocates that role to an individual in the TTA Project.

QM ROLE	RESPONSIBILITIES	Project Role
SQA Manager	QA process execution, use metrics for M&C, reporting	TTA Project Manager
SQA Oversight	Oversight of quality management, Audit QA processes, coordinate solutions, communications	Project Director
Project Director	Resolve conflicts, final decision authority,	Project Director
Technical Debt Manager	Track and manage technical debt	DMS Architect
Development Team	Record defects per definition in metrics Propose and implement solutions Identify and report technical debt Provide effort estimates	Science Software and Archive Group
Development Team Lead	Schedule and report functionality deliveries Assist categorization of technical debt origin	SSA Group Lead
Defect Identifiers	Record defects per definition in metrics	DMS Architect SSA Group Lead TTA Project Scientist Data Analysts
Users	Report on defects via help desk system	Users – Data Analysts or Project Scientist should enter defect in tracking system.

Table 1 Roles and Responsibilities in the TTA Quality Management Process