

Quality plan of mcs

prepared by: Dr. Safi-ur-rahman & Anwar Shamim

Mineral Cadastre System

2014

Table of Contents

[Acronyms and Abbreviations 11](#_Toc388268700)

[INTRODUCTION 13](#_Toc388268701)

[1.1. Purpose 13](#_Toc388268702)

[1.2. Objectives 13](#_Toc388268703)

[1.3. Starting Points 13](#_Toc388268704)

[1.4. Scope 13](#_Toc388268705)

[1.5. Reference Documents 13](#_Toc388268706)

[MANAGEMENT 15](#_Toc388268707)

[2.1. Methodology Overview 15](#_Toc388268710)

[2.2. Meetings 15](#_Toc388268711)

[2.2.1. Sprint Planning Meeting 16](#_Toc388268712)

[2.2.2. Daily Scrum 16](#_Toc388268713)

[2.2.3. Sprint Review Meeting 16](#_Toc388268714)

[2.2.4. Sprint Retrospective 17](#_Toc388268715)

[2.3. Definition of Done 17](#_Toc388268716)

[2.3.1. Story Definition of Done 17](#_Toc388268717)

[2.3.2. Sprint Definition of Done 17](#_Toc388268718)

[2.3.3. Release Definition of Done 17](#_Toc388268719)

[2.4. Release Planning 18](#_Toc388268720)

[2.5. Project Indicators 18](#_Toc388268721)

[2.5.1. Product Metrics 19](#_Toc388268722)

[2.5.2. Project Monitoring Statistics 19](#_Toc388268723)

[2.5.3. Quality Metrics 19](#_Toc388268724)

[2.6. Risk Management 20](#_Toc388268725)

[DOCUMENTATION 21](#_Toc388268726)

[3.1. Provided Documents 21](#_Toc388268728)

[3.2. User Documentation 21](#_Toc388268729)

[3.3. Development documentation 21](#_Toc388268730)

[3.4. Location 21](#_Toc388268731)

[STANDARDS, PRACTICES AND CONVENTIONS 23](#_Toc388268732)

[4.1. Conventions 23](#_Toc388268734)

[4.2. Quality Attribute Profiles 23](#_Toc388268735)

[PRODUCT REVIEWS 25](#_Toc388268736)

[VALIDATION, VERIFICATION AND TESTING 27](#_Toc388268737)

[6.1. Unit Testing 27](#_Toc388268740)

[6.2. Load/Performance Testing 27](#_Toc388268741)

[6.3. Security Testing 27](#_Toc388268742)

[6.4. Usability testing 27](#_Toc388268743)

[6.5. Compatibility Testing 28](#_Toc388268744)

[6.6. Backup and Recovery 28](#_Toc388268745)

[PROBLEM REPORTING AND CORRECTIVE ACTION 29](#_Toc388268746)

[7.1. Code errors 29](#_Toc388268748)

[7.2. Documentation errors 29](#_Toc388268749)

[TOOLS AND TECHNIQUES 31](#_Toc388268750)

[8.1. Testing 31](#_Toc388268752)

[8.1.1. Unit Testing 31](#_Toc388268753)

[8.1.2. Load/Performance Testing 31](#_Toc388268754)

[8.2. Planning 31](#_Toc388268755)

[8.3. Verifying 31](#_Toc388268756)

Table of Figures

**No table of figures entries found.**

List of Table

[Table 1: OpenM4S Project Team 16](#_Toc388268757)

[Table 2: Quality Attributes Profiles 23](#_Toc388268758)

Review History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Acronyms and Abbreviations

|  |  |
| --- | --- |
| Word | Definition / Description |
| DoD | Definition of Done |
| IDE | Integrated Development Environment |
| SQA | Software Quality Assurance |
| SQL | Structured Query Language |
| WoW | Way of Working |

# INTRODUCTION

This document represents the Software Quality Plan for OpenM4S Project Development.

## Purpose

The purpose of this Software Quality Plan is to define the techniques, procedures, and methodologies that will be used to assure timely delivery of the software and that the development system meets the specified requirements within project resources.

## Objectives

Software Quality Assurance is a process for evaluating and documenting the quality of the work products produced during each stage of the software development lifecycle.

The primary objective of the SQA process is to ensure the production of high-quality work products according to stated requirements and established standards.

## Starting Points

The present Quality Plan is mainly based on Non-functional Requirements section of Statement of Requirements document and Methodology part of Way of Working document.

## Scope

This plan discusses the procedures and software engineering processes which are covered by quality assurance controls such as:

* Monitoring quality assurance of the management team throughout the software engineering process
* Development and documentation of software development, evaluation and acceptance standards and conventions
* Verifying the test results
* Tracking problem solving and corrective actions needed as a result of testing

## Reference Documents

The following documents have been used as requirements and references for the development of OpenM4S Quality Plan:

* Statement of Requirements: Initial Solutions for OpenM4S



# MANAGEMENT

In order to create software in compliance with its requirements, starting points, set of restrictions and quality concerns, certain development method should be used. In the case of OpenM4S project it has been suggested to put into practice a Scrum-based development methodology. In the sections below basic issues concerning roles, meetings, main quality aspects in Scrum and risks are briefly introduced.



## Methodology Overview

Scrum is a set of rules and practices to optimize the development environment, manage team organization, and synchronize requirements (and their priorities) with time-boxed iterative prototypes. Each iteration, known as sprint, typically lasts from 2 to 4 weeks. OpenM4S with 2 weeks sprint duration throughout the project. Software features to be implemented in the next sprint are determined in the beginning of the sprint (i.e. planning stage) and cannot be changed throughout its duration. This strictly fixed small sprint duration gives the development process predictability and flexibility.

Main roles in Scrum methodology are:

* Scrum Master

The one who leads Scrum meetings and makes sure that the process respects all principles of Scrum. In other words, the Scrum Master takes care of quality aspects of the development environment and team.

* Product Owner

Person who represents the interests of end users and other interested in the product parties. The Product Owner takes care of quality aspects related to user requirements.

* Scrum Team

This is cross-functional team consisting of developers, testers, architects, analysts, etc. The team is the only fully involved development stakeholder, and is responsible for the project results as a whole. The Scrum Team takes care of developing features according to functional and quality requirements.

From a Scrum perspective, OpenM4S project team is organized according to the Table 1.

## Meetings

Scrum methodology is based on regular meetings where Scrum Team, Scrum Master, Product Owner and third parties discuss different issues concerning development process. From a quality perspective, the purpose of the meetings is to ensure smooth running of the project and to preserve the overall quality aspects of the developed software. OpenM4S Team are doing meeting and they discuss following.

* What we did yesterday?
* What we will do today?
* What issues we are facing during development.

Table 1: OpenM4S Project Team

|  |  |  |
| --- | --- | --- |
| Product Owner: | Dr. Safi Ur Rehman (PI) | [mine.safi@gmail.com](mailto:mine.safi@gmail.com) |
| Scrum Master: | Muhammad Anwar Shamim(Team Lead-1) | itprofessional81@gmail.com |
| Scrum Master: | Qasim Jamal (Team Lead-2) | engrqasimjamal@gmail.com |
| Scrum Team: | Zia Ur Rehman | [ziajustonline@gmail.com](mailto:ziajustonline@gmail.com) |
|  | Saleem Ullah | saleem.ullah.88@gmail.com |
|  | Nahyan Bin Khalid | nahyan.bin.khalid@gmail.com |
|  | Yahya | myb.xmenn@gmail.com |
|  | Iftikhar Mashwani | iftikhar.mashwani@gmail.com |
|  | Zahid Ali | engineerzahidali@gmail.com |
|  | Imdad Ul Haq | [imdad.openm4s@gmail.com](mailto:imdad.openm4s@gmail.com) |

### Sprint Planning Meeting

Sprint Planning Meeting is held at the beginning of every sprint cycle.

During the meeting, Scrum Team and the Product Owner define a sprint goal, which is a short description of what the sprint will attempt to achieve. The success of the sprint will later be assessed during the Sprint Review Meeting against the sprint goal. Product Owner describes the highest priority features to the team. It is important to identify and communicate work estimates and what exactly is likely to be done during the current sprint. Finally, Scrum Team needs to prepare the Sprint plan and Sprint Backlog that detail the time it takes to do the work.

### Daily Scrum

In order to ensure that tasks are being implemented according to their planned schedule, a Daily Scrum meeting occurs during each sprint.

During the meeting, each team member answers three questions:

* What have you done since yesterday?
* What are you planning to do today?
* Do you have any problems that would prevent you from accomplishing your goal?

In case of problems, it is the role of the Scrum Master to ease the resolution of these obstacles. The impediments should be discussed outside the Daily Scrum meeting, not to unnecessarily involve team members who are not concerned with the issues discussed. It is suggested to have the Scrum meeting at the same time, in the same place, and for the same short time duration.

### Sprint Review Meeting

Sprint Review Meeting or Demo Meeting is held at the end of a sprint cycle.

Throughout this meeting the Scrum Team shows what they have accomplished during the sprint. Typically this takes the form of a demo of the newly developed features. During the Sprint Review the project is assessed against the sprint goal determined during the Sprint Planning Meeting. Ideally the team has completed each product backlog item brought into the sprint, but it is more important that the team achieves the overall goal of the sprint. Every feature that has met the “Done” criteria is accepted and marked as completed.

### Sprint Retrospective

Sprint Retrospective meeting is held at the end of a sprint after the Review Meeting.

Sprint Retrospective involves reviewing the way team works, interacts, behavioral aspects, improving technical skills so that the subsequent sprint is faster, etc. Scrum Team and Scrum Master discuss what went well during the sprint and what could be improved.

Although theoretically Scrum intends to hold Sprint Planning Meeting, Sprint Review Meeting and Sprint Retrospective at the separate time, it is recommended to bring all these meetings together into one Post Sprint meeting.

Post Sprint meeting is detailed in WoW document in section “Methodology”.

## Definition of Done

Definition of Done is the exit-criteria to determine whether a product backlog item (user story) is complete. Since practically it is impossible to create a single Definition of Done that suits every project context, the set of requirements should be negotiated by the Scrum Team sprint-by-sprint. A set of basic requirements for “Done” are provided in the next three subsections.

### Story Definition of Done

* Code completed, refactored and reviewed
* Coding standards are met
* Code is covered by a minimum of 70% Unit Tests and all tests are Green (i.e. passed)
* Continuous integration implemented (build automation, deployment and testing)
* Documentation updated (Release Docs, Deployment Guide, Technical Design, Quick Guide)
* Story is accepted by Product Owner

### Sprint Definition of Done

* All stories completed for the sprint accepted by Product Owner
* Product increment accepted by Product Owner at the iteration demo
* All the automated acceptance tests running for the stories in the sprint
* The entire code gone through code reviewing process
* Database scripts automated and tested
* No Critical or Blocker bug exists in the bug backlog
* Subversion trunks updated

### Release Definition of Done

* All stories for release completed and accepted
* The product tested (release does not have any serious bugs)
* Backup successfully made
* Deployment documents updated

## Release Planning

Release planning is the continuous process of defining, splitting and prioritizing the stories in the release backlog. The purpose of release planning is to define the contents of a release or a specific shippable product increment. This involves identifying and com-mitting to the following:

* Goal for the release
* Prioritized set of user stories that will be developed in the release
* Coarse estimate for each user story
* Date for the release

Release planning assumes that a sufficient product backlog of user stories already exist, at least at the reasonable level. It also assumes that the stories in the product back-log are prioritized.

There are two widely practiced approaches to release planning. If the release definition is date based then only those user stories that the development team is able to complete by the given date should be selected. If the release is functionality based, then it is still necessary to estimate when the stories are likely to be completed.

Below are the main input steps for release planning:

* Estimate the user stories
* Make rough estimates of the relative size of the stories (see Section  [2.5](#page12))
* Establish velocity
* Determine how many story points the team is likely to complete in each sprint (see Section  [2.5](#page12))
* Compute forecast

Date-based release can be estimated to complete (velocity × number of sprints) story points. Functionality-based release can be estimated to complete in (total story points ÷ velocity) sprints.

In case of date-based release, the highest priority stories, whose sum is no more than the number of story points (computed as mentioned above), should be selected.

For a functionality-based release, if the estimated completion date (computed above) is acceptable, then all the stories can be selected for the release. If not, then revert to a date-based release.

## Project Indicators

There are three main groups of key performance indicators available for Scrum Master and Product Owner:

* Product metrics (story points, business value, time remaining, etc.)
* Project monitoring statistics (measuring project burn down using product metrics)
* Quality metrics (team behavior and development performance)

All these categories are detailed in the sections below.

### Product Metrics

As product metrics, story points and business values should to be used.

Story points show the technical value (complexity) of the feature. Scrum Team is responsible for allocating story points. In order to come to agreement with evaluating certain feature, the Planning Poker method could be used. The easiest way for approximating the story complexity is to negotiate with the time required for completing the task.

Backlog items in product backlog should be prioritized as an absolute ordering by business value. The Product Owner is responsible for assigning these metrics for further estimating team performance in the project scope (see Section  [2.5.3](#page14)).

### Project Monitoring Statistics

For visually tracking the progress of a sprint, burn down charts should be used.

The total amount of work (in hours or story points) is estimated at the beginning of the sprint. Every day the team members update their status and their work-remaining estimate.

The remaining work as it is determined every day is charted along the time axis on a chart. Depending on the development progress the amount of remaining work can decrease. If work complexity is underestimated, the amount of remaining work goes up.

The purpose of the burn down chart is to visually indicate if development is proceeding well. If it is clear or very likely that the line of remaining work will not reach zero at the end of the sprint, the team needs to discuss with Product Owner:

* What will we actually be able to finish?
* What things need to drop out at the end of the sprint?
* If the dropped out things are big, do we need to re plan smaller stories into the sprint or can the big ones be broken down?

Burn down charts can be used for release planning. In this case, instead of daily updates, the outcome of sprints is being used for the graph. The comparison data is then the remaining amount of work needed for the next release.

### Quality Metrics

Quality metrics are directly related to performance and are pressed to achieve greater transparency and compliance with project requirements. Quality of the team performance could be evaluated both in a way of the project code value and team behavior.

Such quality metrics as percentage of failed builds, acceptance test levels, code coverage, bug found/fixed ratios show the professional quality of the team. These metrics are partly used as statements for Definition of Done.

One of the most important behavioral metrics is team velocity. It is a measurement of how much the team gets done in an iteration. Velocity is calculated by adding up all the completed story points. Since the point values are merely estimates of the perceived difficulty and time necessary to complete the backlog item, a team’s velocity is not especially useful as such. Instead, it becomes a valuable metric over time as teams complete multiple sprints and have the opportunity to establish a consistent velocity. Once this occurs, the Product Owner can look to the team’s established velocity to determine how much work it can tackle in the next sprint.

## Risk Management

All members of the development team have primary responsibility for identifying and mitigating risks during the project development. Any risks identified need to be reported at Daily Scrum meetings.

# DOCUMENTATION

The following sections contain descriptions of documents to be produced as part of software quality assurance process.



## Provided Documents

Final versions of the documents provided in Section  [1.5](#page7) could be used for quality assurance process as well. For instance, the Software Requirement Specification provides a detailed definition of the system and its architecture.

## User Documentation

User documentation for the project will consist of a User’s Manual, available in printed and electronic form. It will include at least the following information for the end-user of the application:

* The proper procedure and requirements for installing the software
* Minimum system requirements (CPU, RAM, disk space, graphical memory, etc.)
* The license and the redistribution policy for the software
* Instructions on how to use the software
* etc.

Firstly, at least three development team members will participate in reviewing of the User’s Manual to assure its accuracy, readability, and completeness. Then, the documentation needs to be provided to the community for further editing and maintaining.

## Development documentation

OpenM4S Documentation into three different documentation:

* API documentation
* Data Dictionary
* User Interface Library

## Location

All kinds of documentation including provided documents, user and development documentation should be uploaded to the collaborative environment and stored there as files with open access.

# STANDARDS, PRACTICES AND CONVENTIONS



## Conventions

Since architecture of OpenM4S project is based on using third party open source components and they all could have personal coding and naming conventions, it might be hard to rewrite all code using unified standards. But all OpenM4S classes and packages should meet naming, coding, versioning and design conventions.

## Quality Attribute Profiles

It is proposed to use a set of quality attribute profiles that can be considered as the most relevant from software engineering perspective. A profile can be regarded as a set of scenarios. In the table below several example profiles are presented.

Table 2: Quality Attributes Profiles

|  |  |  |
| --- | --- | --- |
| Profile | Example Scenario | Action |
| Performance | Usage scenario from different user | Refactoring to minimize |
|  | perspectives: | overheads and synchronization |
|  | - Administrator | delays |
|  | - Applicant |  |
| Maintainability | Adaptation to the one pilot : | System Refactoring |
|  | - DGMM |  |
| Reliability | The effect of component errors on sys |  |
|  | tem reliability as a whole: |  |
|  | - Application framework |  |
|  | - GEO Server |  |
| Safety | The effect of component failure on the | Data protection and isolation |
|  | integrity and persistence of data: |  |
|  | - Application framework |  |
|  | - Geo Server |  |
|  | - Web Interface |  |
| Security | The security of the whole system with | Enhancing access rights and |
|  | respect to: | patching bugs |
|  | - Integrity (hacker could alter parts of |  |
|  | the system) |  |
|  | - Availability (denial-of-service attacks) |  |

# PRODUCT REVIEWS

Reviews are proposed to be conducted by team developers, interested parties and community members. The main objectives of reviewing software are:

* To reveal all kinds of code errors and bugs in project implementation
* To verify that the software meets its requirements
* To ensure that the software has been represented according to predefined conventions and standards
* To achieve software to be developed in a uniform manner

The following is a list of reviews that could be held throughout the OpenM4S project:

* Peer reviews
* Inspections
* Walkthroughs

Code review is proposed to be lightweight. Usage of automated code reviewing tools is not compulsory.

# VALIDATION, VERIFICATION AND TESTING

The following tests will be conducted during the development life cycle of the OpenM4S project.



## Unit Testing

All code needs to 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.

Test team members are responsible for testing each program produced to demonstrate correct operation of coded modules units.

## Load/Performance Testing

Load/performance testing is conducted to demonstrate the correct operation and performance of the OPENM4S release as well as to identify the operational envelope of the solution. The results of the testing need to be recorded in Test Report prepared by the Test Leader.

## Security Testing

Software testing is vulnerability assessment of software. It verifies the actual response of protective mechanisms of penetration built into the system.

During testing, safety test plays the role of hacker. He is allowed to do everything like:

* Doing attempts to know the password using external resources
* Attack the system using special tools that analyze the defense
* Suppression of the system hoping it will refuse to serve other users
* Deliberate delivering of errors in order to get into the system during recovery
* Tracing of unclassified data hoping to find a key to enter the system

All vulnerabilities are considered to be bugs and should be fixed as soon as possible. In case of OpenM4S project for security testing both developers and the community are responsible.

## Usability testing

Usability testing is a method of assessing ease of product use based on the involvement of users as a tester, tester experts, and the summation of the conclusions derived from them.

By theory, setting up a usability test involves creating a scenario, or realistic situation, wherein the person performs a list of tasks using the product being tested while observers watch and take notes. Several other test instruments such as scripted instructions, paper prototypes, and pre- and post-test questionnaires are also used to gather feed-back on the product being tested.

Main testers in case of OpenM4S project have to be the team of developers, side experts and involved users from the community.

## Compatibility Testing

In context of OpenM4S project, compatibility basically means possibilities of the software to run under different operating systems working on the top of diverse hardware solutions (e.g. CPU platforms, RAM, graphical memory, etc.).

## Backup and Recovery

In order to check possibility of the system to be recovered after failures in case of unstable performance, software product should be artificially shutdown several times before every project release. Time that system is needed to recover should be precisely evaluated. This time value is the main quality measure and the team is responsible for decreasing it as much as possible.

Since the backup process is presumed to run automatically while software runs, the accuracy of copying data needs to be controlled as well.

# PROBLEM REPORTING AND CORRECTIVE ACTION

Problems that result from reviews, walkthroughs, inspections, and tests will be corrected by the team developers. The decision on how to fix this or that error could be reached during Daily Scrum meetings.



## Code errors

Code errors (or bugs) could be contributed as patches to JIRA bug/tracking system by the team developers, interested parties and experts who have access to Subversion repository and rights of contributor in JIRA.

All changes to code need to be documented and the patched system should pass through all relevant SQA procedures. All team members have to be notified of changes made through e-mail distribution or during Daily Scrum.

## Documentation errors

Since User and Developer Guides should be provided as word documentation in open access of collaborative environment, everyone could contribute to its content with mandatory post-moderation.

# TOOLS AND TECHNIQUES



## Testing

In order to run automated tests it is strongly recommended to use Robot Framework.

Robot Framework is a generic test automation framework for acceptance testing. It has easy-to-use tabular test data syntax and utilizes the keyword-driven testing approach. Its testing capabilities can be extended by test libraries implemented either with Python or Java (e.g. see Section  [8.1.3),](#page24) and users can create new keywords from existing ones using the same syntax that is used for creating test cases.

The important issue of using Robot Framework is that it could be easily integrated with Bamboo build automation tool (see Way of Working document for details). Robot Framework is open source and is released under Apache License 2.0. More information is provided on official web page.

In addition it is suggested to exploit concrete tools for unit, system and web testing.

### Unit Testing

### Load/Performance Testing

As a tool for load and performance testing Apache JMeter could be applied.

Apache JMeter may be used to test performance both on static and dynamic resources (files, Servlets, Perl scripts, Java Objects, Data Bases and Queries and FTP Servers.

## Planning

In order to manage and improve Scrum development process, gathering and analyzing statistics it is suggested to use specified planning tools.

JIRA in combination with the Green Hopper can be used to visualize team user stories, defects and tasks divided into separate phases. Green Hopper adds agile project management to JIRA to easily manage user stories, requirements and development tasks in the same place with tracking bugs.

## Verifying

OpenM4S specified coding and naming conventions. In order to ease the check it is better to use automatic tools and utilities for verifying code for standards.