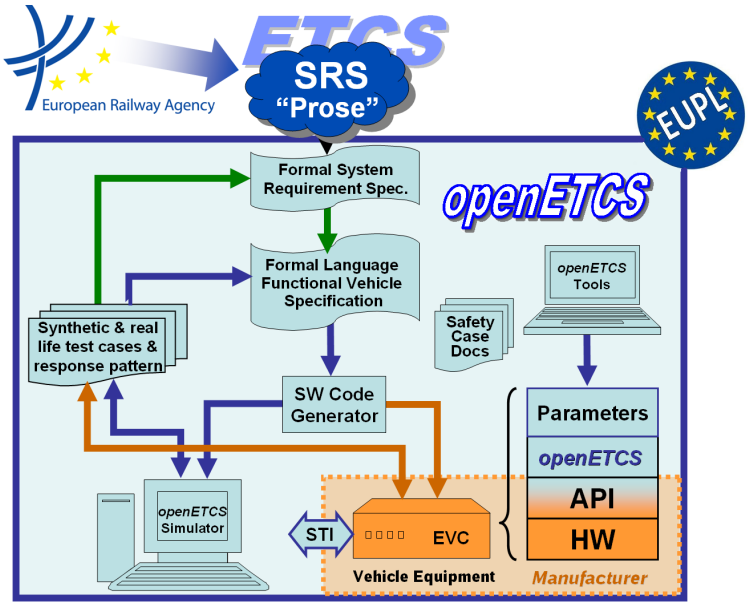
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 1: “Management”

Project Quality Assurance Plan - Revision Process

Ainhoa Gracia

July 16, 2013



Funded by:

Federal Ministry of Education and Research
Région de Bruxelles-Capitale
GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

This page is intentionally left blank

Work-Package 1: “Management” **OETCS/WP1/D1.3.1**
July 16, 2013

Project Quality Assurance Plan - Revision Process

Ainhoa Gracia

Avda. Zugazarte 8,6
48930 Getxo
Vizcaya, España

Description of work

Prepared for openETCS@ITEA2 Project

Contents

Figures and Tables

Figures

Tables

Document History

Version	Date	Chapters modified	Reason	Name
0.1.0	09.04.2013	All	First version	SQS
0.1.1	12.04.2013	Roles	Tables added to the 'Roles' section	SQS
0.1.2	15.04.2013	T.instructions	Technical instructions completed	SQS
0.1.3	16.05.2013	All	Comments of the reviewers analysed and some changes implemented	SQS
0.2.0	23.05.2013	All	New complete version adjusted to the Revision process concept	SQS
0.2.1	06.06.2013	All	Simplified version	SQS

1 Introduction

1.1 Purpose of the document

This document presents the whole process to follow when documents need revision. It aims to provide a set of guidelines as technical instructions for each revision cycle launched that highlights the steps to take.The roles involved in the process are clearly identified as well as their responsibilities and tasks. And finally, the mechanisms needed to achieve the proposed objectives are also included, so the process can be carried out successfully.

1.2 Intended Audience

This document applies to the whole development life-cycle of the project and it addresses all the Author(s), Committers and Contributors involved. This document should be available to all of them in read access mode and it provides guidance about the Revision process to follow anytime revisions of documents are needed.

1.3 Supporting documents

Supporting documents		
Name	Path	Contents
Todonotes package	governance/Review Process	Brief introduction to the todonotes package with detailed descriptions about the available attributes.
Revision Guidelines	Governance/Review Process	It describes major aspects to be assessed during a Revision Cycle.
QA Plan	Governance/QA Plan	It defines the processes, methods and tools that will be used to develop the OpenETCS project.

Table 1. Supporting documents

1.4 Definitions and acronyms

Definitions and acronyms	
Abbreviation	Meaning
Revision Process	It is the process through which committers and selected contributors perform a through analysis of a document, propose contributions and improvements to be included before the document is published in the OpenETCS repository.
Review Process	It is the process through which the OpenETCS community is invited to make comments and suggestions, propose corrections and improvements to a document.
RC	Revision Cycle.
Committer	A person who has editing rights for a specific project.
Contributor	A person without editing rights for a project that contributes with comments or improvements. A Contributor can become a Committer through a voting process.

Table 2. Definitions and acronyms

2 Tools

Tools	
GIT	• GitHub: A web-based hosting service for projects that use Git revision control system. • SmartGit: A graphical front-end for Git distributed version control systems.
Pdf documents	• Adobe Acrobat Reader: Software package that allows to view, navigate and print pdf files. • Diffpdf: Open source application that compares different PDF files for discrepancies. • PDF Creator: Software for creating pdf files. Works like a printer on your PC.
TeX documents	• MiKTeX : Provides the tools necessary to prepare documents using de TeX/LaTeX mark up language. • GhostScript. • GhostView. • TexMaker. • Todonotes package. • Adobe Acrobat Reader: Software package that allows to view, navigate and print pdf files.

Table 3. Tools

3 Revision Process overview

The revision process will be performed:

- After a first version of the document is created by the authors and when it is plausible significant contributions are still needed to have a complete and publishable document.
- After a major rework of a document or after a review process in which major improvements/comments have been detected.

In all cases:

- The revision process will be carried out in a dedicated Revision Cycle Branch.
- In case of conflicts, it is the Product Owner who will be responsible for their resolution

- The Revision Team is composed of Committers and invited Contributors
- Contributions/comments are made directly in the document under revision. However, in the case of “Contributors”, the Issue Tracker Tool will be used.
- The launching, control and clarification processes are managed through the Issue Tracker Tool

3.1 Revision Roles

This section describes the roles of the participants in the revision process of a document:

Roles	
Role	Competencies
Product owner	He/She is the main responsible of a document. He/She is in charge of launching the revision process; preparing the revision plan and controlling and monitoring the process progresses according to the plan; informing and interacting with the revision team members; approving and rejecting changes and resolving conflicts.
Author(s)	They provide support to the Product Owner in the revision of the contributions made by the revision team and therefore in the implementation of the approved ones. When requested by the Product Owner, they will provide support in assessing whether the document is ready for publication or not.
Revision Team	As experts in the theme of the project, they will provide meaningful contributions to guarantee the document meets its scope and purpose in a complete and accurate way.

Table 4. Revision Roles

3.2 Description of the Revision Process

The next figure shows the different stages of the Revision Process. Right after the activities to be performed in each Stage are provided. The technical instructions on how to accomplish such activities are included in the Annex.

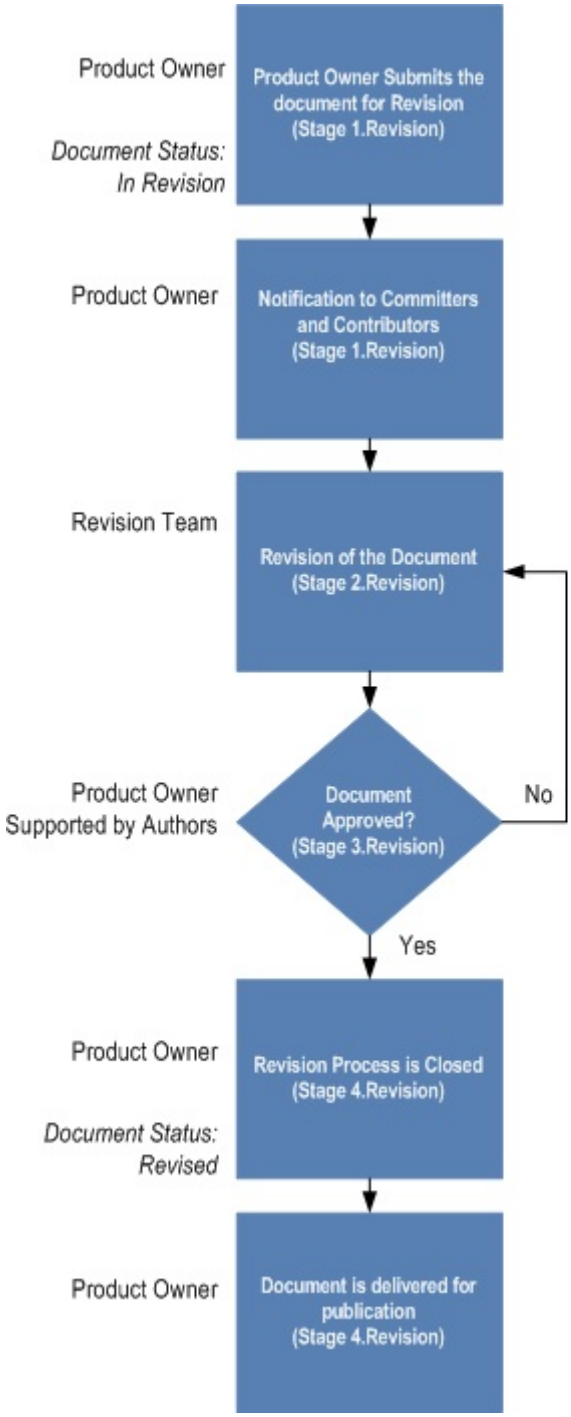


Figure 1. Revision Process flow

3.2.1 Stage One: Launching of the Revision Process

- The Product Owner elaborates a Revision Plan for the document with the envisaged deadlines for the different stages of the process and with the scope of the revision process itself. This information will be notified to the revision team members with the invitation.
- The Product Owner creates the Revision Cycle Branch (RC Branch) and uploads the document to be revised to the RC Branch. Refer to *A1.Create a New RC Branch* in Annex for Technical Instructions.

- The Product Owner updates the status of the document under revision in the wiki page "List of Documents" of the project to "In Revision". Besides, a specific reference to the revision process is to be included in the ReadMe file.
- The Product Owner selects the revision team.
  - At least the Committers of the project where the project belongs to will be selected.
  - Contributors may also be invited to participate when considered appropriate.
- The Product Owner invites the Committers and Contributors to participate. Refer to *A2.Revision Process-Control and Monitoring* in Annex for Technical Instructions.
- The revision process starts officially.

3.2.2 Stage Two: Revision process

- Committers will be allowed to include the comments/contributions directly on the document under revision. Refer to *A3. Revision Activities* in Annex for technical instructions.
- Once a committer finalizes with the revision, a notification will be sent to the Product Owner. Refer to *A3.Revision Activities* in Annex for Technical Instructions.
- Contributors will make their comments/contributions as "Issues". Refer to *A3. Revision Activities* in Annex for technical instructions.
- The Product Owner with the support of the rest of the Author(s) of the document will analyze the comments/contributions as they arrive and when needed, interact with the appropriate revision team members asking for clarification. Refer to *A2.Revision Process-Control and Monitoring* in Annex for Technical Instructions.
- The deadline for receiving comments may be extended by the Product Owner. Both the finalization and the extension will be notified to the revision team members. Refer to *A2.Revision Process-Control and Monitoring* in Annex for Technical Instructions.

3.2.3 Stage Three: Approval Process

- The Product Owner, with the support of the Authors, will approve and/or reject comments for final implementation. Refer to *A4. Approval/Rejection of Changes* for Technical Instructions.
- Resolution of conflicts will be under the responsibility of the Product Owner
- The Product Owner will implement the final changes in the document. Refer to section *A5. Final Implementation* for Technical Instructions.
- The Product Owner will assess the comments received and will decide whether the document is ready for publication or not.
- If it is not ready for publication, the Revision Cycle is re-opened by extending the deadline for receiving comments/contributions. Refer to *A2.Revision Process-Control and Monitoring* in Annex for Technical Instructions.
- If the document is ready for publication, the Revision is ready to be closed. Refer to section *A6.Revision Closing* for technical instructions.

3.2.4 Stage Four: Revision Closing

- The Product Owner merges the RC Branch to the Master Branch. Refer to *A7.RC Branch Merging* in Annex for Technical Instructions.
- The Product Owner updates the status of the document in the wiki to “Revised”.
- The Product Owner opens a “Review Process” for the document.

4 ANNEXES A1.- Technical Instructions

A1. Create a New RC branch

TI	Create a New RC branch
Roles	Product owner
Description	Any changes made to the document to be revised shall be made in a RC branch context. The Product owner shall create the branch and makes it available for the revision team.
Steps	• Steps to create the branch locally are: 1. Open Smartgit 2. Branch menu, add branch and give a new name following this nomenclature: – *RC_<name of the document to be revised>_<version of document>*. 3. Push *add branch & switch*. With these steps the branch is created as a local branch. • Integrate the branch into the GitHub repository 1. Go to the toolbar and press *Push*, accept then the messages. 2. The new branch appears in the Smartgit Branches view, below the origin tag. 3. Confirm that the local branch is linked to the remote GitHub branch: in the local branch the *set tracked branch shall* has been done and it shall address to the RC branch already created in the GitHub repository.

Table 5. TI - Create a New RC branch

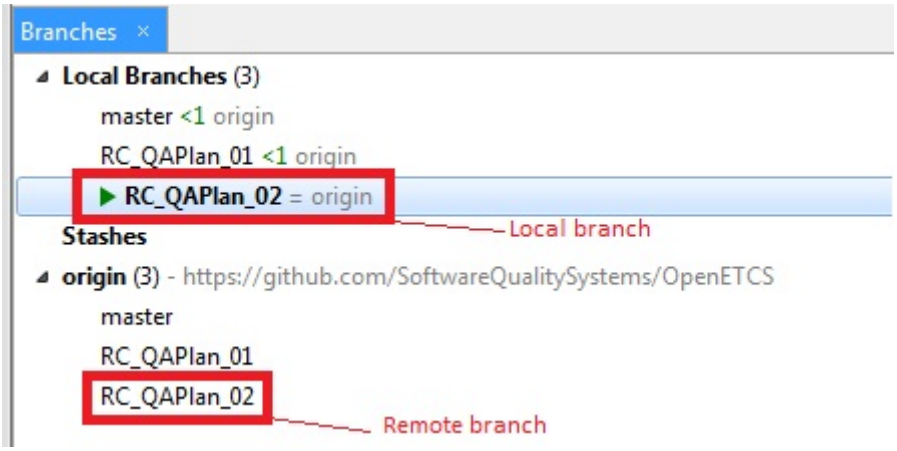


Figure 2. Branches tree in *Smartgit*

A2. Revision Process-Control and Monitoring

TI	Revision Process-Control and Monitoring
Action	Role/Instruction
Invitation to Participate to the Revision Team and launching the revision process	Product Owner Create a new Issue (Main Issue of the RC) in the Repository indicating a new RC is launched: • Open Github and go to the correspondent repository • Select the *Issues section* • Push the *New issue* button • Add a descriptive title indicating the name of the new RC and the document under revision • Add a significant description about the causes that have motivated the new RC and summarizes the objectives of the revision. Provide the list of required Committers/Contributors. • Push *Submit new issue* If considered of interest, an e-mail can be sent to the invited people to the revision process
Interaction with the Revision Team to clarify comments	Product Owner Create an Issue addressing the corresponding Revision Team Member: • Add a Title identifying the RC process • Add Text describing the details of the clarification requested This Issue will be the Main Issue for the interaction with the expert. The rest of the communication will adopt the form of Comments to this Main Issue
Notification of Extension of the Revision Stage	Product Owner Add a Comment to the Main Issue of the RC indicating the deadline for receiving comments has been extended. The new deadline will be indicated.
Notification that the Revision Stage has finalized	Product Owner Add a Comment to the Main Issue of the RC indicating the date for receiving comments has finalized.
Notification the Revision Period has been re-opened	Product Owner Add a Comment to the Main Issue of the RC indicating the Revision period has been re-open and therefore a new deadline for receiving comments is indicated. Besides specific indications for the process will be indicated.

Table 6. TI - Revision Process-Control and Monitoring

A3. Revision Activities

TI	Revision environment setup
Roles	Revision Team
Description	Each Committer shall prepare the working environment locally and ensure it is ready before starting with the Revision tasks. The environment setup shall be optional for Contributors due to the do not have editing rights and they can have access to the documents to be downloaded directly from the GitHub website; in any case, they can configure the environment to have the documents locally, although they cannot edit them and *push* changes into the repository.
Steps	• Setup using *Smartgit* tool 1. Clone the repository with the *Project –> Clone* option. 2. Go to the *Branch* menu, select *Add branch* and give a name. Then, the branch is created as a local branch. 3. Link the local branch to the remote git branch, to do that, select *Set tracked branch* in the context menu of the local branch. The local environment is then ready for working in the Revision Process • Update the environment – It is essential to work in the last version of the document under Revision, so minimal conflicts appear in the future when uploading the changes. To be sure about this, a pull request shall be done to the repository before editing the document. ∗ Select the *Pull* option on the toolbar

Table 7. TI - Revision Activities

TI	Revision work
Roles	Revision Team
Description	The Committers and Contributors shall perform the revision in the expected time and conditions. The work of a Committer/Contributor shall finish when the Product owner confirms the closing of the RC.
Steps	• The Committer shall: 1. Make comments, suggestions or improvement proposals with the *todonotes* (See *A10 Add notes using todonotes*). 2. Add comments in the RC issue thread when it applies (See *A11 Add comments in a RC issue*). 3. Make changes in the document. Each insertion shall be highlighted indicating the initials of the person involved. The Revision process shall be done by different people and what comments have been written by whom shall be known. • The Committer shall integrate the changes into the document that is hosted in the remote repository in GitHub. To do that, click on the *Push* button in the SmartGit tool. When pushing different situations can happen: – There are no conflicts with the original repository . 1. The changes shall be included. – There are one or more conflicts. 1. Click on the *Pull* button so the last version is loaded. 2. Open the *Conflict Solver* Window to compare the committed version in the *Github* and the local version. 3. The conflict shall be solved in the following way. (a) The Committer shall indicate that the version stored in the remote repository is the correct one. The changes in case of conflict are not included. (b) The Committer will *Commit* and *Push* the changes that do not have any conflict and add a notification in the document about that. (c) After finishing the *pushing*, he/she shall perform a *pulling* to obtain the integrated and last committed version; then he/she shall add a note using *todonotes* tool in the section where the conflicts were; he/she shall explain the problem found. 4. Have in mind that after *pushing/pulling* the document, the changes in conflict made by the Committer shall be lost. Anytime a conflict appears, the Committer shall prepare a copy of the suggestions or modifications made by him/her. 5. He/she also adds a comment in the RC issue thread (if it exists) or send an e-mail exposing that problem and requesting a solution. 6. The Author(s) and/or the Product owner shall take part in the discussion and propose a solution. The Committer can put in the document again the suggestions in conflict stored locally if the Author(s) or the Product owner requires that.

Table 8. TI - Revision work

A4. Approval/Rejection of Changes

TI	Approval/rejection of Changes
Roles	Author(s)
Description	The Author(s) shall study each proposal, recommendation or comment that appears in the document under revision and decide how to implement the proposed changes in case he/she estimates it is appropriate to be included in the document. On the other hand, and considering the Committers could include text in the document, the Author(s) shall assess the appropriateness of those sections and decide if this new content is accurate and relevant. When the contributions come from Contributors their issues shall be assessed due to they do not have editing rights.
Steps	• When a Committer notifies that some changes have been pushed to GiThub using the corresponding RC branch, the Author(s) synchronizes their repository to obtain the last commit made for the document with the *SmartGit* tool. • The Author(s) reads carefully any annotation that appears in the document and decides whether the comments shall be implemented or not. • For each annotation the Author(s) find(s) in the document, a written confirmation about what is the decision about the subject shall be provided. 1. Use the *todonotes* to confirm/reject proposals made by the Committers (*A10 Add notes using todonotes*) or to accept/delete new sections, text or paragraphs added by the Committers. 2. In any case, a justification for that decision shall be included. 3. When a suggestion is accepted: (a) Assess whether the recommendation made implies writing new paragraphs, sections, adding new figures, etc. and modify the document accordingly. (b) Highlight the new text or the modified text with *todonotes* using the *inline* option. • Whenever a Contributor has added comments to an opened issue or has created a new one with contributions using the *Issue Tracker* tool, the Author(s) shall collect all the contributions reported, and assess the implementation of changes. The process shall be the same as in case of Committers but the tool to be in contact with the Contributor shall be the *Issue Tracker*. • The Author(s)commits the changes made in the document and push them into the RC branch so the Committers/Contributors can have access to the changes, confirmations and rejections made by the Author(s). • The Author(s) add(s) a message to the RC issue thread (if it exists) or send(s) an e-mail, so everyone can be informed about the commit recently done.

Table 9. TI - Approval/rejection of Changes

A5. Final Implementation

TI	Final Implementation
Roles	Product owner
Description	The Product owner shall provide a final version with the considered changes implemented.
Steps	• Once the *Approval/rejection of comments* have been done, the changes implemented and pushed into the corresponding directory, a complete version of the document with those changes but without the comments shall be prepared. • The pdf and the original document shall be *commited* and *pushed* in the *Review Documents* directory. • A notification shall be included in the issue linked to the ongoing RC (if it exists) or send by e-mail: 1. There, the Product owner shall explain that the changes have been implemented and that there is a complete version available in the corresponding path.

Table 10. TI - Final Implementation

A6. Revision closing

TI	Revision closing
Roles	Product owner
Description	The Product owner shall be the responsible for closing the current Revision Cycle after verifying and confirming all the changes done. The first task to be done is to notify the closing and highlight the results obtained.
Steps	• Add a Comment in the Revision issue thread (if it exists) to indicate that the RC has finished 1. Follow the indications provided in the *A11 Add comments in a RC issue* to add a comment 2. Provide a brief summary of the RC: (a) Indicate the way the objectives have been met (b) Are there pending objectives?. Indicate the reason for closing the RC before all the objectives have been met. (c) Identify the key aspects of the Revision (d) Highlight the results obtained • Close the RC issue thread pushing the *Close* button in *GitHub* when there is a Revision thread. • Or notify the closing of the RC sending an e-mail. The information included in this case is the same as before.

Table 11. TI - Revision closing

A7. RC Branch merging

TI	RC Branch merging
Roles	Product owner
Description	Merging the *RC Branch* to the *Master branch* implies incorporating the changes made in a document to the main branch of the repository. In this way, the new version of the editable document is available in the master branch.
Steps	• Merge the branches 1. Switch to the *master branch*, so that the working tree will be the master branch from this moment on 2. With the RC branch selected the Product owner shall do a click on the right button and select *merge to working tree*. 3. Select the documents not to be included in the master branch and push *Ignore* (Contextual Menu). • Confirm the update 1. Select the *Commit and Push* option. 2. Make *synchronize* so all the changes made are reflected remotely and locally.

Table 12. TI - RC Branch merging

A10. Add notes using ToDoNotes

TI	Add notes using todonotes
Roles	Product owner, Author(s), Revision Team
Description	The package todonotes must be installed locally to make comments in the document under Revision.
Steps	• Verify the installation of the todonotes package locally. In case the package is not installed, follow these steps: 1. Got to the *MixTex* tool directory, select *Maintenance (Admin)* and click on *Package Manager (Admin)*. 2. Look for the *todonotes* package using the filtering option. 3. Select the package after the search is done and install. • Be used to the most common commands available in the package. 1. Go to the *governance* repository in the Github and look for the *Reviews Process* folder. There you can find a *todonotes* document that explains all the commands that can be used during the reviewing. • Identify always any comment done with your initials. 1. The initials of the person who inserts a comment into the document shall be the the first letter of the name plus the surname. • Use different colours in the document for different meanings 1. *Red*: indicate in a comment what paragraph, section or line is proposes to be deleted. 2. *Green*: the suggestion made by has been approved. 3. *Orange*: the comment refers to any conflict found between commits done by different people. It also shall be used when commentign a suggestions that shall be discussed and not implemented for the moment. 4. *Yellow*: any general comment done by the participants in the Revision.

Table 13. TI - Add notes using ToDoNotes

A11. Add comments to a RC Main Issue

TI	Add comments in a RC issue
Roles	Product owner, Author(s), Revision Team
Description	The issue opened by the Product owner at the beginning of the RC shall be used whenever a notification shall be done, conflict reported or questions asked. The collaborative work shall aim to get a dynamic approach when the discussions are fluid, with quick answers and sharing of information.
Steps	• Go to the repository where the document under Revision is located. • Select the issue that has the RC you are working on. • The comments posted shall be descriptive enough so any reader can understand the message. 1. Identified yourself clearly, providing your name and role in the project. 2. When required, include diagrams, figures, partial texts or specific data that help to understand the problematic found. • Do not edit any comment done. It is a better option to rewrite it with the additions you proposed than editing and make the changes directly. In this way, it is assured that everyone reads the new message because in the other case, the change/addition can be missed. • Push the *Comment* button to post the message.

Table 14. TI - Add comments to a RC Main Issue

A12. Versioning

TI	Versioning
Description	Once the changes implemented for the document under revision have been confirmed, the Product Owner prepares a new version of it.
Steps	Versions consist of three numbers, e.g. "1.3.1". All digits start with a "0" for the first release, so the first version is always "0.1.0". The digits of a version number have the following meaning: major.minor.service. An increased number in one of the digits marks the corresponding type of release of an artifact, e.g a version increased from "1.0.1" to "1.0.2" marks a service release, while a version increased from "1.0.1" to "1.1.1" marks a minor release. The three types of releases have the following meaning: • Service: Only contains fixes, does not add any new API (for software) or content (for documents). • Minor: Adds new API (for software) or content (for documents), but remains compatible to the last minor release • Major: Changes existing API (for software) or content (for documents) and is not compatible with the previous major release After the three numbers, a qualifier can be added to specify the state of an artifact: • 0.1.0.draft: The ongoing development, which leads to the release 0.1.0 • 1.2.1.M1: The first milestone on the road to release 1.2.1 • 1.0.0.RC1: The first release candidate for release 1.0.0 Versions should only be increased once in preparation of a release according to the versioning scheme. Versions should not be increased during on-going work. Git will do the versioning during this period. Only graduated projects are allowed to do releases starting with a "1" or higher For further information see *Versioning Artifacts* in ecosystem wiki

Table 15. TI - Versioning

5 ANNEXES A2.- Quick guide for the Revision process

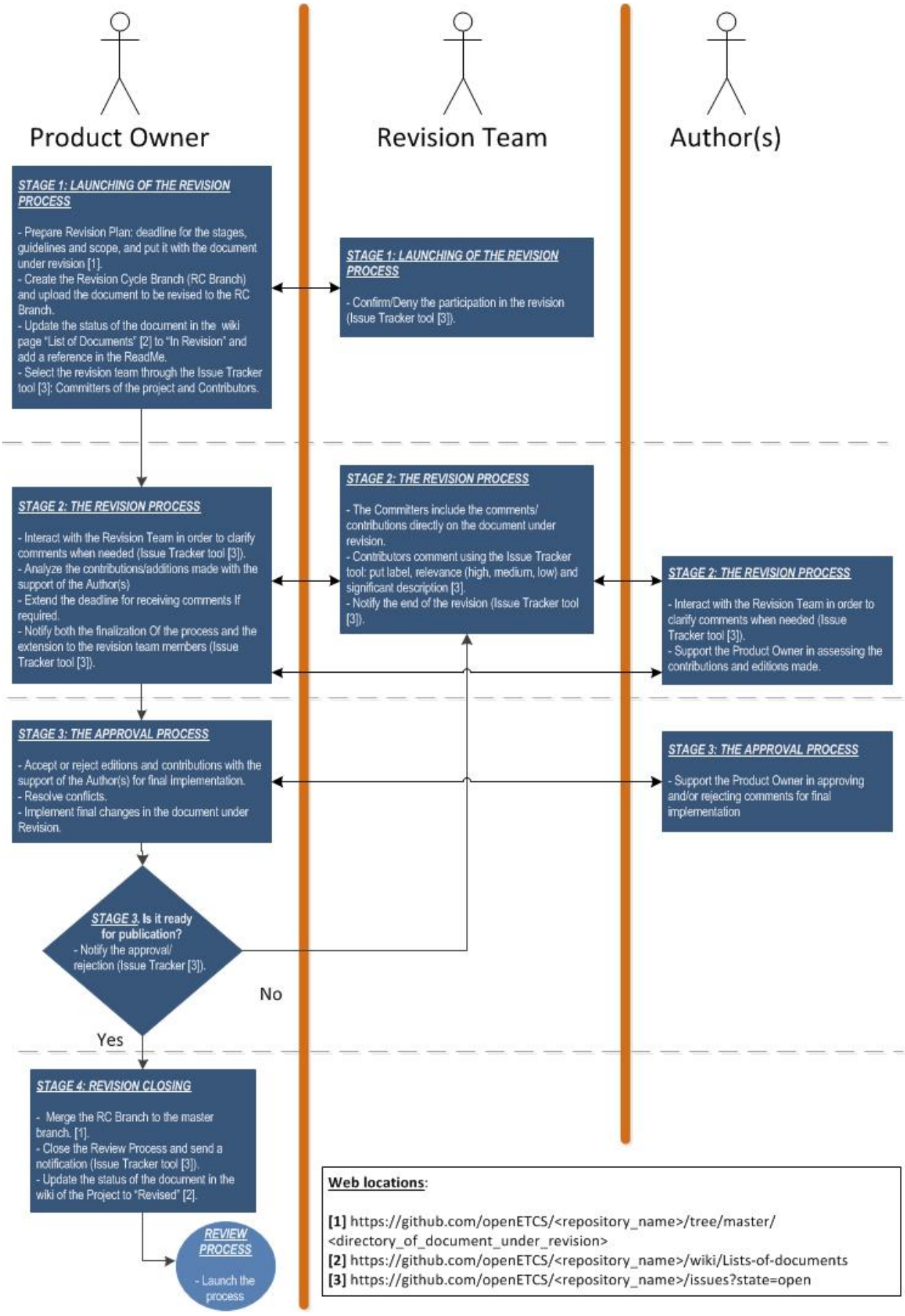


Figure 3. Review Process summary