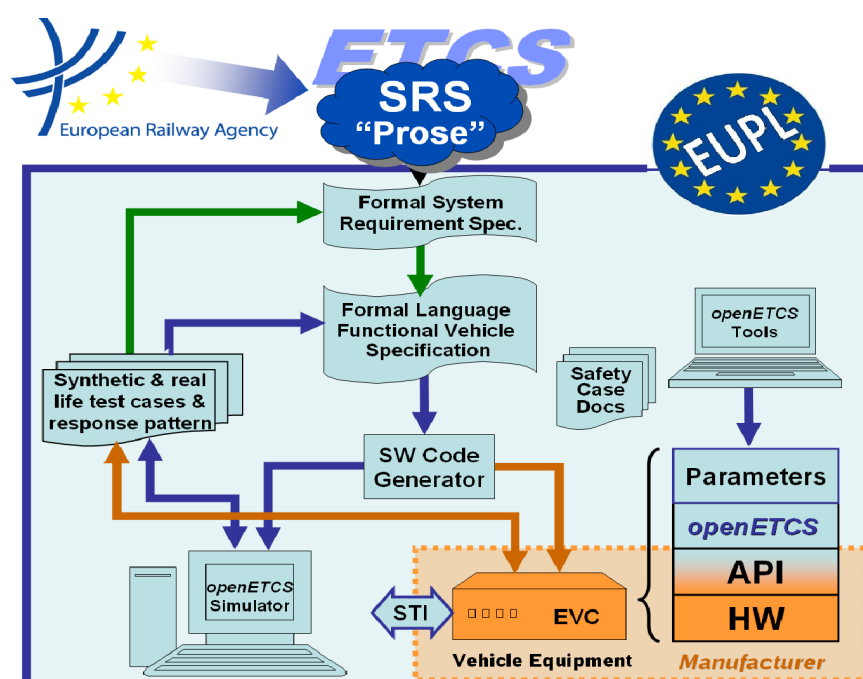


Work-Package 1: “Management”

# Project Quality Assurance Plan - Revision Process

SQS

May 23, 2013



supported by:



This page is intentionally left blank

**Work-Package 1: “Management”**

**OETCS/WP1/D1.3.1  
May 23, 2013**

# Project Quality Assurance Plan - Revision Process

SQS

Avda. Zugazarte 8,6  
48930 Getxo  
Vizcaya, España

## Description of work

This work is licensed under the European Union Public Licence (EUPL v.1.1) and a Creative Commons Attribution-ShareAlike 3.0 Unported License.



Prepared for ITEA2 openETCS consortium  
Europa

**Disclaimer:** This work is licensed under the European Union Public Licence (EUPL v.1.1) and a Creative Commons Attribution-ShareAlike 3.0 – (cc by-sa 3.0)

THE WORK IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

<http://creativecommons.org/licenses/by-sa/3.0/>

# Contents

## Document History

Version	Date	Chapters modified	Reason	Name
0.0.1	09.04.2013	All	First version	SQS
0.0.2	12.04.2013	Roles	Tables added to the 'Roles' section	SQS
0.0.3	15.04.2013	T.instructions	Technical instructions completed	SQS
0.0.4	16.05.2013	All	Comments of the reviewers analysed and some changes implemented	SQS
0.0.5	23.05.2013	All	New complete version adjusted to the Revision process concept	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) and committers 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

Name	Path	Contents
Todonotes package	governance/Review Process	Brief introduction to the todonotes package with detailed descriptions about the available attributes.

## 1.4 Definitions and acronyms

Abbreviation	Meaning
Revision Process	The OpenETCS committers are invited to perform a revision of the contents of a document. The committers can edit the document, writing complete sections or making changes directly in it (LaTeX, todonotes and Smartgit tools). The author(s) shall support the whole process answering questions and resolving conflicts. Once the author(s) and committers are satisfied with the results, the Product owner shall confirm and approve or reject and request more changes to the revised document. Finally, the approved document is pushed into the corresponding GitHub repository.
Review Process	Any person involved in the GitHub community and/or the OpenETCS project is invited to participate in a Review Process. They shall not edit the document but send comments and suggestions creating the corresponding issues in the <i>Issue Tracker</i> tool. Once the deadline has arrived, the author(s) shall collect the comments and make the changes they consider appropriate. Any doubt shall be answered by the reviewers using the issue threads opened and in case of conflicts the collaboration of the Product owner may be required. The process may be launched after a Revision Process ends, there are relevant issues reported for a release or whenever the Product owner considers appropriate.

RC	Revision Cycle - A complete cycle with an established deadline (one or two weeks maximum) where the author(s) and the committers can read, analyse and make suggestions or changes directly to the document under revision. Once the document is approved the cycle finishes and a new version shall be available in the corresponding GitHub repository.
Product owner	The role played by the main author of a document or the project leader. The responsibilities of the role include launching a Revision process, inviting committers to the process, closing discussions, taking decisions when there are conflicts and doing the final approval of the document among others.
Committer	The role played by a member of the OpenETCS project who have been granted with this role by other committers and the project leader. There are some tasks that only can be done by committers such as: editing and pushing changes or new documents into a GitHub repository, closing of issues in the <i>Issue Tracker</i> or being involved in Revision processes.

## 2 Tools

Tools	
GIT	<ul style="list-style-type: none"> <li>• GitHub: A web-based hosting service for projects that use Git revision control system.</li> <li>• SmartGit: A graphical front-end for Git distributed version control systems.</li> </ul>
Pdf documents	<ul style="list-style-type: none"> <li>• Adobe Acrobat Reader: Software package that allows to view, navigate and print pdf files.</li> <li>• Diffpdf: Open source application that compares different PDF files for discrepancies.</li> <li>• PDF Creator: Software for creating pdf files. Works like a printer on your PC.</li> </ul>
TeX documents	<ul style="list-style-type: none"> <li>• MiKTeX : Provides the tools necessary to prepare documents using de TeX/LaTeX mark up language.</li> <li>• GhostScript.</li> <li>• GhostView.</li> <li>• TexMaker.</li> <li>• Tdonotes package.</li> <li>• Adobe Acrobat Reader: Software package that allows to view, navigate and print pdf files.</li> </ul>

## 3 Revision Process overview

### 3.1 Structure of the repository



The Revision process involves the creation of the following directory inside the structure of each repository once a *new branch* for the revision has been created.

Structure of the repository		
Name		Content
Revision ments	Docu-	<ul style="list-style-type: none"> <li>• The document under revision: In Tex and PDF formats with all its history of changes, comments and sections added.</li> </ul>

### 3.2 Revision Roles

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

Roles	
Role	Competencies
Product owner (main author or project leader)	<ul style="list-style-type: none"> <li>• Launch the Revision Process.</li> <li>• Open a Revision cycle for a document creating a git new branch (<i>TI.1.1. New RC branch</i>)</li> <li>• Notify the Committers via a personal e-mail or opening a new issue. (<i>TI.1.2. Invite Committers</i>)</li> </ul> <p>Control Revision Process (<i>TI.1.3. Supervision of the Revision Process</i>)</p> <ul style="list-style-type: none"> <li>• Approve the revised version and confirmation of changes made (<i>TI.1.4. Final approval and confirmation of the revised version</i>)</li> <li>• Send the notification of closing. (<i>TI.1.5. Revision closing</i>)</li> <li>• Merge the associated RC branch into the repository master branch. (<i>TI.1.6. RC Branch merging</i>)</li> <li>• Generate an established version of the document and pushed it in the corresponding GitHub repository. (<i>LaTex and PDF versions</i>)</li> </ul>

Roles	
Role	Competencies
Author(s)	<ul style="list-style-type: none"> <li>• Supervision of the work being done. <ul style="list-style-type: none"> <li>– Use todonotes tool to add, verify and reject comments in the document being reviewed. (<i>TI.2.1. Approval/rejection of comments</i>)</li> <li>– Make changes to the document when required.</li> <li>– Make a request to the product owner for a final confirmation of the changes made.</li> </ul> </li> </ul>

Roles	
Role	competencies
Committers	<ul style="list-style-type: none"> <li>• Confirm the participation in the Revision. Justify the no involvement in any other case.</li> <li>• Prepare the working environment: the committer clones the repository in local, creates a branch and associates it to the repository on GitHub (<i>TI.3.1. Revision environment setup</i>).</li> <li>• Be aware of how to perform the revision work (<i>TI.3.2 Revision work</i>) <ul style="list-style-type: none"> <li>– Make comments, suggestions or improvement proposals with the todonotes package installed for the LaTeX editing tool. (<i>TI.4.2 Add notes using todonotes</i>)</li> <li>– Work directly in the document, adding texts and sections.</li> <li>– Send comments in case of conflicts or doubts to the author(s) through the <i>Issue Tracker</i> or e-mail.</li> </ul> </li> </ul>

### 3.3 Description of the Revision Process

This subject is concerned with the validation and verification of the documentation generated within the OpenETCS project. The process shall aim to confirm that what has been produced is correct and meets the expectations of the committers as well as improving the quality of the document and its reliability; it also ensures the technical approach is appropriate before releasing the related documentation.

A Revision Cycle will help the project to better meet its strategic goals and objectives, so the correctness of the deliverables is ensured and they cover their intended purpose. The following aspects shall be assessed during the Revision Process:

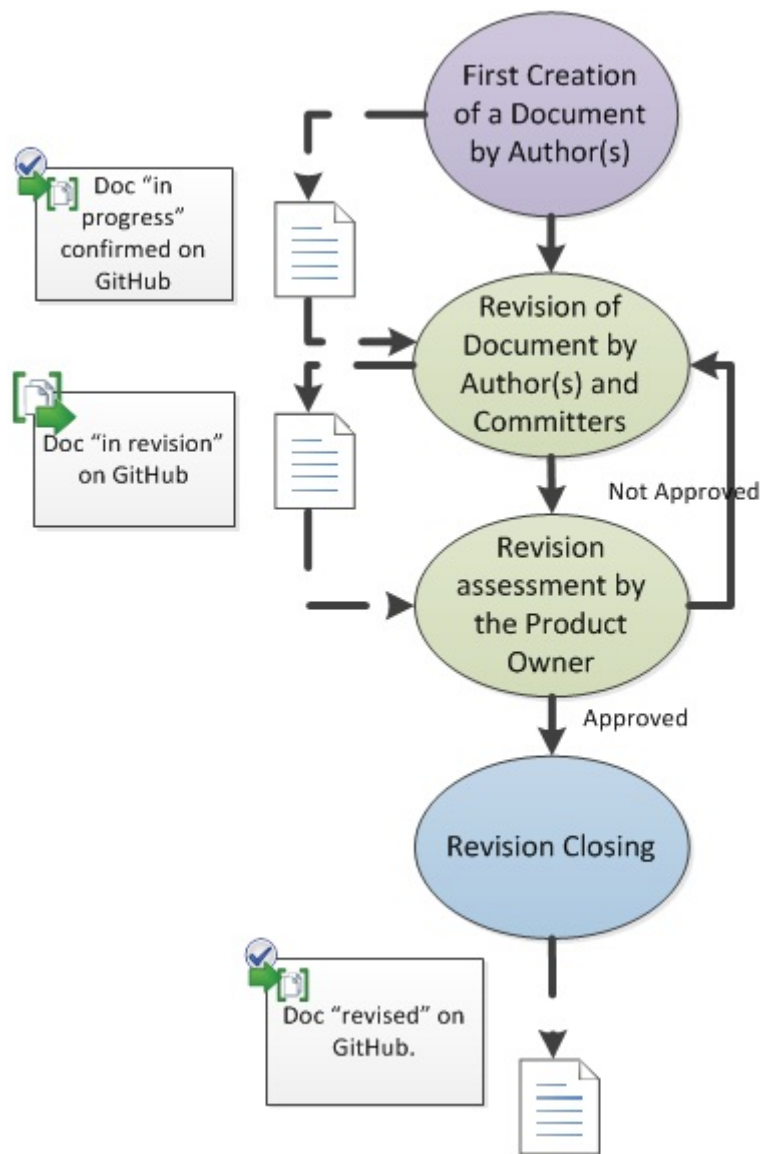
- The material is technically accurate

- The goals of the document have been achieved
- Quality, style and grammar (do not forget the spell check)
- Document uses consistent tense
- Document addresses the audience appropriately
- Sentences use simple structures
- Unnecessary information is eliminated
- Does the reader leave wanting more?
- Internal cross-reference within the document is accurate
- References to other existing documents are valid

The Revision Process refers to documents that are prone to be under revision consideration because:

- The authors have delivered its first version
- After a Review Process has finished and there are major changes and improvements to be done in the document by the Author(s) and Committers.
- The Product owner justified a new revision.

The following diagram provides a general overview of the process.



**Figure 1. Revision-Review Processes flow**

This whole process is independent from the habitual work done in the repositories; so, whereas the OpenETCS members work as usually do in the master branch, the new Revision Cycle shall be attached to a specific repository branch (previously created by the Product owner). Then any change, comment, suggestion or improvement to do to the identified document under revision, shall be done inside this RC branch and in the correspondent *Revision Documents* directory inside the OpenETCS structure.

In this way, the day to day work with regard to the master branch shall not be affected by the work being done in the Revision context; the whole revision cycle for a specific document shall be linked to a specific branch of the repository. This is the key of the process because with this mechanism there shall be no dependencies between the ordinary work done in the project and the different Revision cycles launched for different documents at different times.

During their lifetime, these documents shall be involved at least in one complete Revision Cycle, but it there will be as many Revision Cycles as considered necessary. The different Revision Cycles linked to each document shall be numbered and associated to their branches.

The process is divided in the following stages

### 3.3.1 Document Creation Stage

After the author(s) has completed the initial draft of a document, it must be made available to the committers involved in the Revision Process.

The document shall be pushed into the corresponding GitHub repository in the appropriate formats (PDF, LaTeX, etc).

Having the document in a format that is viewable online in GitHub provides the highest degree of access from both local and remote locations.

The most common online formats for documents in OpenETCS project are LaTeX and PDF. Each has its own advantages and, to an extent, disadvantages.

- LaTeX documents
  - They are supported by multiple opensource tools, text processors and other editing packages.
  - It is generally easy to publish and provide access to remote users via GitHub, as well as being the most familiar format to online users.
  - LaTeX provides the standard formatting and font size thanks to the use of OpenETCS Latex templates.
- PDF documents
  - The other common format is PDF. This requires the Adobe Acrobat™ software be installed on the author's and committers PC, but provides the advantage of preserving the look and feel of the original document.
  - PDFs are also familiar to online users and are viewable in a Web browser, although it is not possible in the GitHub context.

For this first version the Product owner shall be clearly identified:

- The main author
- Or the project leader

And he/she shall be the responsible for launching the Revision process and taking decisions related to the approval of the revised version of the document.

### 3.3.2 Revision Stage

The launching of the Revision process shall consist on the creation of a new *Revision Cycle* by the Product Owner:

- When starting the cycle the Revision plan and the deadline shall be clear.
- The Product owner shall contact the committers who have specific knowledge and appropriate experience and invite them to the process. At least, the list shall involve the committers of the project where the document is localised.

- Identify the main contributors, at least one name shall be obtained.
- Prepare a private e-mail for each of the identified contributors highlighting the purpose of the Revision and what is expected from his/her collaboration.
- Specify a deadline for confirmation so the Revision Process is not delayed.
- The Product owner shall be the responsible for addressing and conducting the main aspects of the Revision during the expected time of the cycle.
- The Product owner shall be aware of the progress of the Revision, so it is under control.

The concept of the Revision stage is simple, but it can be time consuming and error prone.

Only confirmed people shall be involved in a Revision Process editing and modifying a document. They shall be registered Committers with appropriate knowledge and skills to be capable of assessing and revising a document.

Whenever a person is required to be involved in the process and he/she is not a commiter, then a request shall be prepared by the Product owner to launch a voting process in the project. Once the person has been confirmed as a Committer, he/she can contribute in the Revision Process. Any other type of reviewers shall be involved only in the Review Process.

The process is limited to determined number of *author(s)* and *committers* that can edit a document and push the changes to a GitHub repository.

During the Revision process, the author(s) and committers generally benefit from collaborating with each other, improving their productivity; and in most cases, they will convey the errors, corrections, and suggestions they want in the document. The participants in the revision shall be guided by the Product owner during the whole process so the optimum level of integration is reached.

The conflicts that can appear due to simultaneous work in the same document shall be resolved before merging the changes into the document under Revision. So any conflict detected, question or problem found at any moment could be communicated using the *Issue Tracker* tool or sending an email. The Author(s) and/or the Product Manager shall be responsible for providing the appropriate support.

The revision shall be done locally by the committers/autho(s) and once they are satisfied with the changes push them to GitHub. They can use the Smartgit tool and ensure they have switched their work context to the required RC branch. After this, then should synchronize the contents (so they have the last committed version of the document) and push their revision. The author(s) and committers can use the todonotes package to include comments and suggestions into the document or highlight new sections or paragraphs.

When the document under revision is an intensive or complex one, the Product owner or the author(s) can propose online meetings to solve pending issues.

When a revision is pushed to GitHub by a committer, the update shall be notified by sending a message in to the Revision team.

The stage implies the following key points:

- The committers must communicate to the author(s) they have made changes to a document.

- Addressing the process with the use of the Smartgit tool, or any equivalent tool, to push the changes to a GitHub repository shall provide an adequate history of the changes. At any moment there shall be available only one document and not different versions of it made by different people involved.
- With the todonotes tool the changes made in the document should be highlighted, so any reader can identify them easily and a consistent final confirmation of changes shall be made by the Product owner. This is essential when a considerable number of committers are involved, because otherwise, the confusion becomes even greater.

At the end of the Revision Process and once the committers and author(s) are satisfied with the results, the Process owner shall be required to finally confirm and approve the version, so it can be pushed into the GitHub repository with the corresponding numbering modified.

To summarize a successful Revision cycle lies in:

- The committers' ability to communicate changes as well as their ability to collaborate together.
- Committers can benefit from seeing what others have done.
- By knowing who has requested a change, and why, committers can leverage the work of one another. This prevents multiple members of the same revision team from notifying the author of the same error, or some other correction.
- Once this process is complete, the author(s) then notify the Product owner responsible for document the approval, the next stage in the Revision process.

### 3.3.3 Approval Stage

Now that the Revision stage is completed, the approval stage begins.

During this stage, the Product owner can resolve duplicate or conflicting information and requests.

If required, more information can be requested from a committer about a particular aspect of the revision.

The changes made are then either confirmed or rejected with a justification.

When the changes and corrections have been accepted, then any comment made using todonotes or text highlighted shall be deleted so a clear version of the document is obtained.

### 3.3.4 Closing

The Product owner prepares the version (in LaTeX and PDF) and pushed both documents to GitHub with the appropriate version numbering.

He/she shall be also the responsible for closing the corresponding issue in the repository (if it exists) or confirming the closing via e-mail to all the author(s) and committers involved.

He shall merge the associated RC branch into the repository *master branch* so the changes are officially available in the main branch of the repository.

## 4 ANNEXES - Technical Instructions

In this section the whole Revision process is explained step by step using technical instructions, so all the participants (author(s) and committers) involved in the process are aware of the mechanisms they shall implement to work and achieve the expected objectives.

### 4.1 Technical Instructions for the Product Owner

This section includes the Technical Instructions the Product owner shall be aware of.

#### 4.1.1 New RC branch

TI	TI.1.1. 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	<ul style="list-style-type: none"> <li>Steps to create the branch locally are: <ol style="list-style-type: none"> <li>Open Smartgit</li> <li>Branch menu, add branch and give a new name following this nomenclature: <ul style="list-style-type: none"> <li><i>RC_&lt;name of the document to be revised&gt;_&lt;number of Revision&gt;</i>.</li> </ul> </li> <li>Push <i>add branch &amp; switch</i>. With these steps the branch is created as a local branch.</li> </ol> </li> <li>Integrate the branch into the GitHub repository <ol style="list-style-type: none"> <li>Go to the toolbar and press <i>Push</i>, accept then the messages.</li> <li>The new branch appears in the Smartgit Branches view, below the origin tag.</li> <li>Confirm that the local branch is linked to the remote GitHub branch: in the local branch the <i>set tracked branch shall</i> has been done and it shall address to the RC branch already created in the GitHub repository.</li> </ol> </li> </ul>



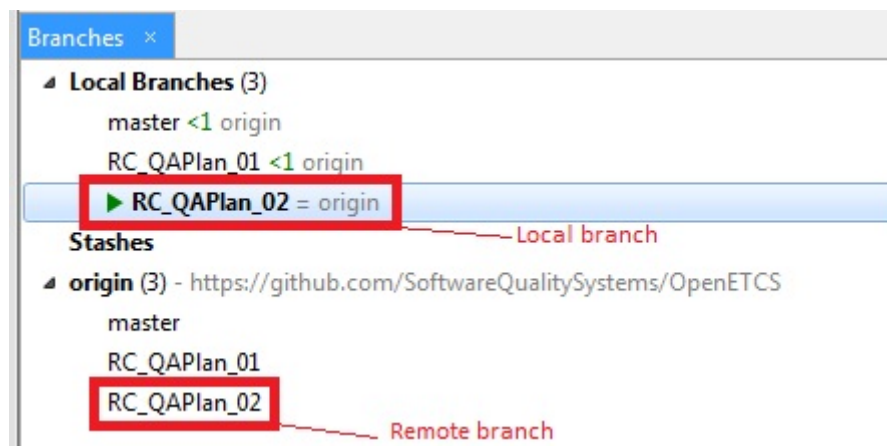
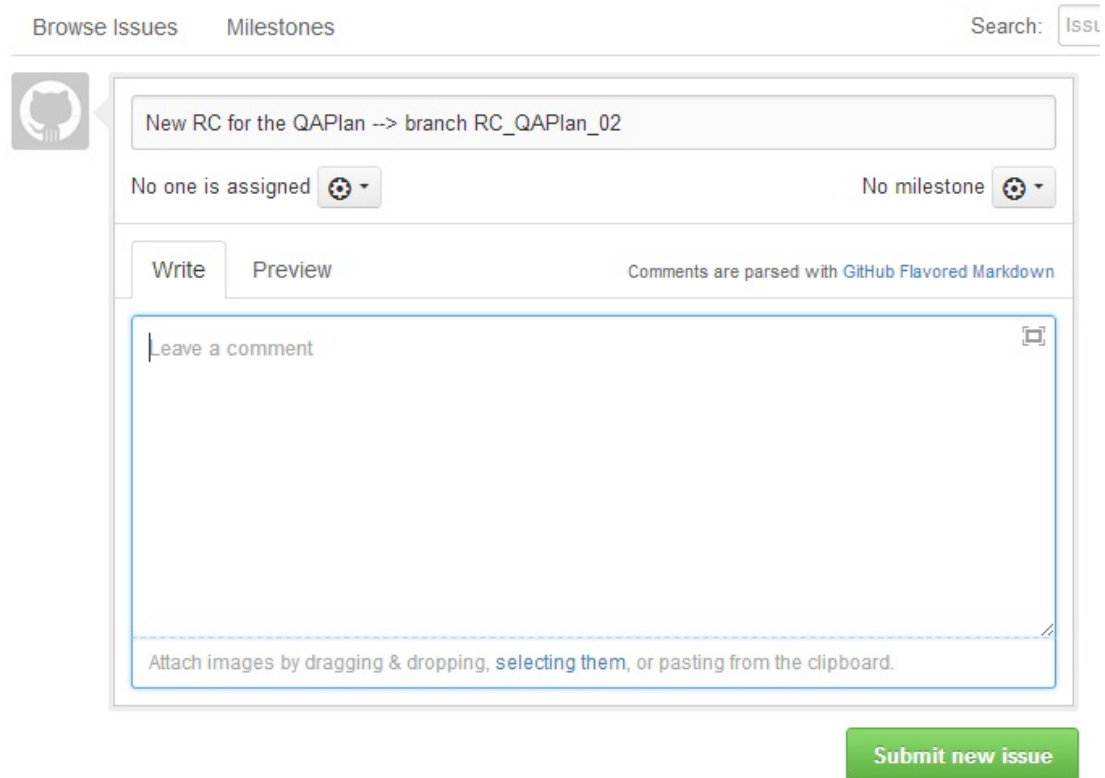



Figure 2. Branches tree in Smartgit



#### 4.1.2 Invite Committers

TI	TI.1.2. Invite Committers
Roles	Product owner
Description	A new issue could be opened to contain the discussions, comments and notifications related to a specific RC for a document. Otherwise, the communications for the revision can be made privately using the e-mail and the fixed list of author(s) and committers.
Steps	<ul style="list-style-type: none"> <li>• Create a new issue in the repository that contains the document under revision. <ol style="list-style-type: none"> <li>1. Open Github and go to the correspondent repository</li> <li>2. Select the <i>Issues</i> section</li> <li>3. Push the <i>New issue</i> button</li> <li>4. Add a descriptive title indicating the name of the new RC and the document under revision</li> <li>5. 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.</li> <li>6. Push <i>Submit new issue</i></li> <li>7. Notify the opening of the issue to the whole revision team sending an e-mail and confirming the use of the <i>Issue tracker</i> to monitor the progress of the process as well as to have the discussions.</li> </ol> </li> <li>• Or send a private e-mail <ol style="list-style-type: none"> <li>1. Add a descriptive subject indicating the name of the new RC and the document under revision</li> <li>2. Add a significant description about the causes that have motivated the new RC and summarizes the objectives of the revision.</li> </ol> </li> </ul>




Browse Issues Milestones Search:

 New RC for the QAPlan --> branch RC\_QAPlan\_02

No one is assigned  No milestone 

Write Preview Comments are parsed with GitHub Flavored Markdown

Leave a comment 

Attach images by dragging & dropping, selecting them, or pasting from the clipboard.

Submit new issue

Figure 3. New issue in *Github*

#### 4.1.3 Supervision of the Revision Process

TI	TI.1.3. Supervision of the Revision Process
Roles	Product owner
Description	The Product owner shall be responsible for assessing the progress done in the RC
Steps	<ul style="list-style-type: none"> <li>The author(s) of the document shall inform the Product owner when they and the committers have reached an agreement about the state and changes made to the document. The Product owner can read the last version available and start analyse the contents to decide whether the document shall be approved or not.</li> <li>The Product owner shall be the responsible for updating the state of the document under revision and its backlog in the Wiki of the repository.</li> </ul>

#### 4.1.4 Final confirmation of changes

TI	TI.1.4. Final confirmation of changes
Roles	Product owner
Description	The Product owner shall provide a final version with the considered changes implemented.
Steps	<ul style="list-style-type: none"> <li>• Once the <i>Approval/rejection of comments</i> 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.</li> <li>• The pdf and the original document shall be <i>committed</i> and <i>pushed</i> in the <i>Review Documents</i> directory.</li> <li>• A notification shall be included in the issue linked to the ongoing RC (if it exists) or send by e-mail: <ol style="list-style-type: none"> <li>1. There, the Product owner shall explain that the changes have been implemented and that there is a complete version available in the indicated path</li> </ol> </li> </ul>

#### 4.1.5 Revision closing

TI	TI.1.5. 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	<ul style="list-style-type: none"> <li>• Add a Comment in the Revision issue thread (if it exists) to indicate that the RC has finished <ol style="list-style-type: none"> <li>1. Follow the indications provided in the <i>TI.4.2 Add comments in a RC issue</i> to add a comment</li> <li>2. Provide a brief summary of the RC: <ol style="list-style-type: none"> <li>(a) Indicate the way the objectives have been met</li> <li>(b) Are there pending objectives?. Indicate the reason for closing the RC before all the objectives have been met.</li> <li>(c) Identify the key aspects of the Revision</li> <li>(d) Highlight the results obtained</li> </ol> </li> </ol> </li> <li>• Close the RC issue thread pushing the <i>Close</i> button in <i>GitHub</i> when there is a Revision thread.</li> <li>• Or notify the closing of the RC sending an e-mail. The information included in this case is the same as before.</li> </ul>

#### 4.1.6 RC Branch merging

TI	TI.1.6. RC Branch merging
Roles	Product owner
Description	Merging the <i>RC Branch</i> to the <i>Master branch</i> 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	<ul style="list-style-type: none"> <li>• Merge the branches               <ol style="list-style-type: none"> <li>1. Switch to the <i>master branch</i>, so that the working tree will be the master branch from this moment on</li> <li>2. With the RC branch selected the Product owner shall do a click on the right button and select <i>merge to working tree</i>.</li> </ol> </li> <li>• Confirm the update               <ol style="list-style-type: none"> <li>1. Make <i>push</i> to the repository and the merge shall be done.</li> <li>2. Make <i>synchronize</i> so all the changes made are reflected remotely and locally.</li> </ol> </li> </ul>

## 4.2 Technical Instructions for the Author(s)

This section includes the Technical Instructions the Author(s) shall be aware of.

### 4.2.1 Approval/rejection of comments

TI	TI.2.1. Approval/rejection of comments
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.
Steps	<ul style="list-style-type: none"> <li>• 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 <i>SmartGit</i> tool.</li> <li>• The Author(s) reads carefully any annotation that appears in the document and decides whether the comments shall be implemented or not.</li> <li>• 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. <ol style="list-style-type: none"> <li>1. Use the <i>todonotes</i> to confirm/reject proposals made by the Committers (TI.4.2 Add notes using <i>todonotes</i>) or to accept/delete new sections, text or paragraphs added by the committers.</li> <li>2. In any case, a justification for that decision shall be included.</li> <li>3. When a suggestion is accepted: <ol style="list-style-type: none"> <li>(a) Assess whether the recommendation made implies writing new paragraphs, sections, adding new figures, etc. and modify the document accordingly.</li> <li>(b) Highlight the new text or the modified text with <i>todonotes</i> using the <i>inline</i> option.</li> </ol> </li> </ol> </li> <li>• The Author(s) commits the changes made in the document and push them into the RC branch so the committers can have access to the changes, confirmations and rejections made by the Author(s).</li> <li>• 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.</li> </ul>

## 1 Introduction

Refer to FPP in order to give a hint/overview how to get familiar with whole openETCS !!!!

This software quality Assurance Document will cover the standards, processes, and procedures for the openETCS project in order to achieve a correct implementation.

### 1.1 openETCS Project Goals

The OpenETCS main objective is the development of an “open proofs” platform that integrates technologies from various stakeholders and enables the use of formal verification techniques in order to dramatically improve the software quality for embedded control systems in terms of reliability, maintainability, safety, and security.

AGracia

The Introduction is pending to be made

RKaseroni

Ok!!. The introduction shall be prepared once all the sections have been completed

Figure 4. Using *todonotes* package Accept/Reject changes *Github*

### 4.3 Technical Instructions for the Committers

This section includes the Technical Instructions the Committers shall be aware of.

#### 4.3.1 Revision Environment setup

TI	TI.3.1. Revision environment setup
Roles	Committers
Description	Each Committer shall prepare the working environment locally and ensure it is ready before starting with the Revision tasks.
Steps	<ul style="list-style-type: none"> <li>• Setup using <i>Smartgit</i> tool               <ol style="list-style-type: none"> <li>1. Clone the repository with the <i>Project</i> → <i>Clone</i> option.</li> <li>2. Go to the <i>Branch</i> menu, select <i>Add branch</i> and give a name. Then, the branch is created as a local branch.</li> <li>3. Link the local branch to the remote git branch, to do that, select <i>Set tracked branch</i> in the context menu of the local branch. The local environment is then ready for working in the Revision Process</li> </ol> </li> <li>• Update the environment               <ul style="list-style-type: none"> <li>– 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.                   <ul style="list-style-type: none"> <li>* Select the <i>Pull</i> option on the toolbar</li> </ul> </li> </ul> </li> </ul>

#### 4.3.2 Revision work

TI	TI.3.2 Revision work
Roles	Committers
Description	The Committers shall perform the revision in the expected time and conditions. The work of a committer shall finish when the Product owner confirms the closing of the RC.
Steps	<ul style="list-style-type: none"> <li>• The Committer shall: <ol style="list-style-type: none"> <li>1. Make comments, suggestions or improvement proposals with the <i>todonotes</i> (See <i>TI.4.1 Add notes using todonotes</i>).</li> <li>2. Add comments in the RC issue thread when it applies (See <i>TI.4.2 Add comments in a RC issue</i>).</li> <li>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.</li> </ol> </li> <li>• The Committer shall integrate the changes into the document that is hosted in the remote repository in GitHub.</li> <li>• To do that, click on the <i>Push</i> button in the SmartGit tool. When pushing different situations can happen: <ul style="list-style-type: none"> <li>– There are no conflicts with the original repository . <ol style="list-style-type: none"> <li>1. The changes shall be included.</li> </ol> </li> <li>– There are one or more conflicts. <ol style="list-style-type: none"> <li>1. Click on the <i>Pull</i> button so the last version is loaded.</li> <li>2. Open the <i>Conflict Solver</i> Window to compare the committed version in the <i>Github</i> and the local version.</li> <li>3. The conflict shall be solved in the following way. <ol style="list-style-type: none"> <li>(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.</li> <li>(b) The Committer will <i>Commit</i> and <i>Push</i> the changes that do not have any conflict and add a notification in the document about that.</li> <li>(c) After finishing the <i>pushing</i>, he/she shall perform a <i>pulling</i> to obtain the integrated and last committed version; then he/she shall add a note using <i>todonotes</i> tool in the section where the conflicts were; he/she shall explain the problem found.</li> </ol> </li> </ol> </li> <li>4. Have in mind that after <i>pushing/pulling</i> 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.</li> <li>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.</li> <li>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.</li> </ul> </li> </ul>

Steps	<ul style="list-style-type: none"> <li>In any case, the Committer shall inform about the progress of the work posting a message in the RC issue thread or sending an e-mail. (See <i>TI.4.2 Add comments in a RC issue</i>).</li> </ul>
-------	---

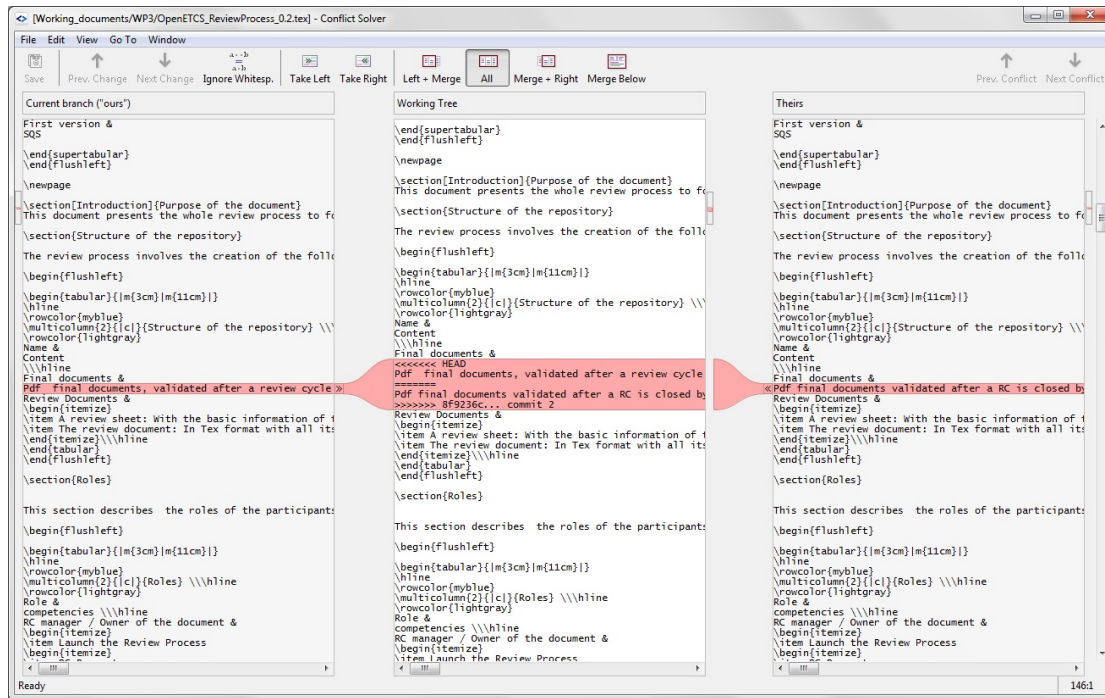


Figure 5. Conflict Solver Window

## 4.4 Technical Instructions for any Role

This section includes any Technical Instruction that shall be followed by the Product owner, the Author(s) or the Committers at different phases of the Revision Process

### 4.4.1 Add notes using todonotes



TI	TI.4.1 Add notes using todonotes
Roles	Product Owner, Author(s), Committers
Description	The package todonotes must be installed locally to make comments in the document under Revision.
Steps	<ul style="list-style-type: none"> <li>• Verify the installation of the todonotes package locally. In case the package is not installed, follow these steps: <ol style="list-style-type: none"> <li>1. Got to the <i>MixTex</i> tool directory, select <i>Maintenance (Admin)</i> and click on <i>Package Manager (Admin)</i>.</li> <li>2. Look for the <i>todonotes</i> package using the filtering option.</li> <li>3. Select the package after the search is done and install.</li> </ol> </li> <li>• Be used to the most common commands available in the package. <ol style="list-style-type: none"> <li>1. Go to the <i>governance</i> repository in the Github and look for the <i>Reviews Process</i> folder. There you can find a <i>todonotes</i> document that explains all the commands that can be used during the reviewing.</li> </ol> </li> <li>• Identify always any comment done with your initials. <ol style="list-style-type: none"> <li>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.</li> </ol> </li> <li>• Use different colours in the document for different meanings <ol style="list-style-type: none"> <li>1. <i>Red</i>: indicate in a comment what paragraph, section or line is proposes to be deleted.</li> <li>2. <i>Green</i>: the suggestion made by has been approved.</li> <li>3. <i>Orange</i>: 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.</li> <li>4. <i>Yellow</i>: any general comment done by the participants in the Revision.</li> </ol> </li> </ul>

#### 4.4.2 Add comments in a RC issue

TI	TI.4.2 Add comments in a RC issue
Roles	Product owner, Author(s), Committers
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	<ul style="list-style-type: none"> <li>• Go to the repository where the document under Revision is located.</li> <li>• Select the issue that has the RC you are working on.</li> <li>• The comments posted shall be descriptive enough so any reader can understand the message. <ol style="list-style-type: none"> <li>1. Identified yourself clearly, providing your name and role in the project.</li> <li>2. When required, include diagrams, figures, partial texts or specific data that help to understand the problematic found.</li> </ol> </li> <li>• 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.</li> <li>• Push the <i>Comment</i> button to post the message.</li> </ul>