EOSC Technical Specification

Software Quality Assurance

Technical Area:	Software Quality Assurance (SQA)
Version:	version 1.0
Status:	Final
Document Link:	

DELIVERY SLIP

Date	*Name*	*Partner/Activity*
From:	Joao Pina	LIP
Moderated by:	Pablo Orviz	CSIC
Reviewed by:	Cristina Duma, Baptiste Grenier	INFN, EGI
Approved by:	Joao Pina	LIP

DOCUMENT LOG

Issue	*Date*	*Comment*	*Author*
	10/01/2020	Initial version	Joao Pina
	15/03/2020	Final version	Joao Pina

TERMINOLOGY

https://wiki.eosc-hub.eu/display/EOSC/EOSC-hub+Glossary

Terminology/Acronym	*Definition*
SQA	Software Quality Assurance
EOSC	European Open Science Cloud
QC	Quality Criteria
CI	continuous Integration
CD	Continuous delivery

Table of Contents

Introduction

The software quality assurance (SQA) is the process responsible for the overall supervision of the software development lifecycle ensuring that the required quality level is achieved. The SQA encompasses all software development processes starting from the definition of requirements, coding, release, testing and integration.

This technical area covers ways to deliver quality software for EOSC consumption and favours the adoption of automated solutions over the traditional manual-based validation mechanisms. The automation allows not only to speed up the development tasks but as well improves the reliability of the developments “ensuring the fast execution of defined tests at each change in the codebase” and keeping them aligned with the initial user requirements and design “Fast feedback received at any development stage - the faster release of quality software”.

Adopted standards

The present document follows the well-known practices and standards adopted by the open-source community.

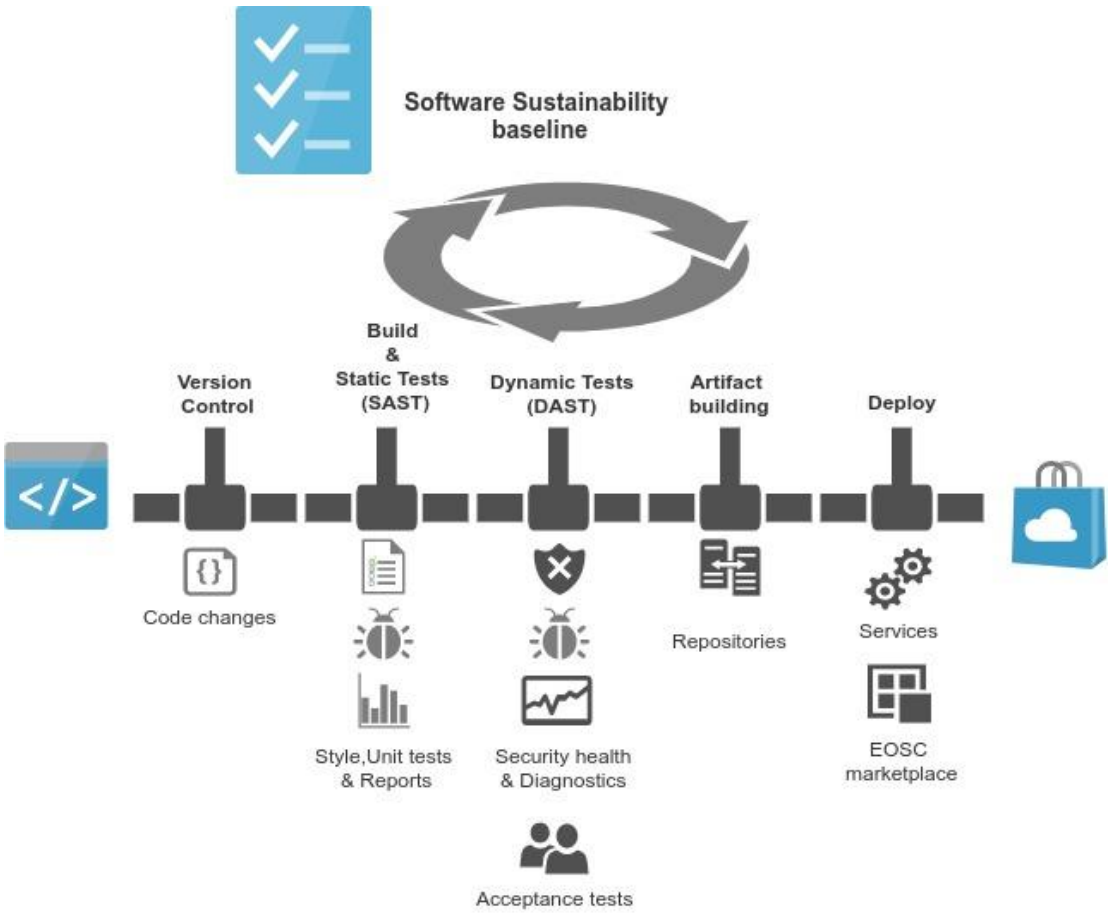
Standard	Short description	References
IEEE 730-2014	This standard establishes the requirements for initiating, planning, controlling and executing the Software Quality Assurance processes of software development.	IEEE 730-2014
ISO/IEC/IEEE 12207:2017	International Standard - Systems and software engineering -- Software life cycle processes - establishing a common framework for software life cycle processes	IEEE 12207:2017

Guidelines	Short description	References
Common SQA Baseline Criteria for Research Projects	A set of Common Software Quality Assurance Baseline Criteria for Research Projects	White paper for SQA Baseline for Research projects v3.0
EGI QC 7	7th release of the EGI Quality Criteria, that is used for the validation of software products within EGI’s Software Provisioning Process	http://egi-qc.github.io/
Semantic Versioning	Best practices for handling software versions	https://semver.org/

High-level Service Architecture

The SQA delivers assistance to service providers throughout the software development lifecycle in order to attain desired quality and timely delivery in the software produced, and consequently, promotes the maturity of the EOSC services.
Even if currently, there are no specific EOSC-hub services that offer support for the assessment of the software quality requirements range from code review to static and dynamic testing, which includes both security and interoperability tests. The assessment of compliance with the quality requirements is implemented using continuous integration (CI) and delivery (CD) pipelines, where successfully produced artefacts can be made readily available through the EOSC repositories.

A possible architecture for the Integration of sustainable software for quality services into EOSC is depicted in the following figure:



Interoperability guidelines

The following items make a short description of the topics relevant to EOSC:

- **Code Accessibility:**
  - Following the open-source model, the source code shall be open and publicly available through social coding platforms in order to increase its visibility and foster collaboration.
- **Licensing:**
  - Licenses must be physically present (e.g. as a LICENSE file) in the root of all the source code repositories related to the software component.
- **Versioning:**
  - Semantic versioning is recommended for tagging the new software releases, avoiding any dependency conflict.
- **Documentation:**
  - Meaningful and differentiated documentation must be available and maintained for each specific audience of the given software: user, admin, and developer. It shall be available online (using a documentation repository) and preferably produced using a markup language (such as Markdown or reStructuredText). Thus, the documentation is treated as code using a VCS.
- **Code Workflow:**
  - Software is best managed by means of a version control system (VCS) solution, which facilitates the adoption of a branching model to conduct the development. Thus, the production version of the software remains in a working state, while the new features or bugs are added. Moreover, several versions of the software can be maintained simultaneously, such as long-term support (LTS) versions.
- **Code metadata:**
  - Software shall be uniquely identified via a persistent identifier so that it can be easily discovered, reused, citable and preserved. Adding metadata to describe the software (in the code repository) is the first step towards its identification.
- **Security:**
  - Security assessment shall be continuously performed on every change in the source code. Tools for security static (SAST) and dynamic analysis testing (DAST) already cover the most common security flaws in the code and in the running services.
- **Code review:**
  - Human oversight of the changes done in the code shall be the last step in the assessment of each new feature or bug implemented, once the test phase has been completed. The suitability of the changes implemented, the statements and/or libraries used, and the security review tasks are commonly associated with code reviews.

Examples of solutions implementing this specification

Some software products delivered under EOSC are already compliant with the quality conventions described in the *Interoperability Guidelines* section. Examples of such software products and their respective continuous integration and delivery pipelines are:

- Infrastructure Manager
  - https://github.com/indigo-dc/im/blob/master/Jenkinsfile
- udocker
  - https://github.com/indigo-dc/udocker/blob/master/Jenkinsfile
- PaaS orchestrator
  - https://github.com/indigo-dc/orchestrator/blob/master/Jenkinsfile
- cloud-info-provider
  - https://github.com/EGI-Foundation/cloud-info-provider/blob/master/Jenkinsfile
- WaTTs
  - https://github.com/indigo-dc/wattson/blob/master/Jenkinsfile
- oidc-agent
  - https://github.com/indigo-dc/oidc-agent/blob/master/Jenkinsfile

Procedure to integrate a service with the EOSC Hub Quality Assurance service

Currently, there are no specific EOSC-hub services that offer support for the assessment of the software quality nevertheless the SQA already provides some ready-to-use continuous integration and delivery pipelines based on automation services such as Jenkins. To this end, a library with the most common functionalities needed during the testing and delivery phases is available.
So a service provider must fulfil the following conditions:

1. Source code must reside on a hosting service repository with version control (e.g. GitHub).
2. Licensing, Documentation and versioning must be publicly accessible.

In addition service providers should:

3. Deploy an automation service (e.g Jenkins) and adapt pipelines[1] in order to make automatic tests. A library for Jenkins with some common functionality needed to implement the testing and delivery phases is already provided[2].

[1] https://jenkins.io/doc/book/pipeline/
[2] https://github.com/indigo-dc/jenkins-pipeline-library