A set of Common Service Quality Assurance Baseline Criteria for Research Projects

A DOI-citable version of this manuscript is available at http://hdl.handle.net/.

This manuscript was automatically generated on .

Authors

Abstract

The purpose of this document is to define a set of quality standards, procedures and best practices to conform a Service Quality Assurance plan to serve as a reference within the European research ecosystem elated projects for the adequate development, deployment, operation and integration of services into production research infrastructures.

Copyright Notice

Acknowledgements

The EOSC-Synergy project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 857647.

Document Log

Issue	Date	Comment
v0.1	27/04/2020	First draft version
v0.2	28/02/2020	Second draft version

Introduction

The Open Science realization in Europe is already taking its first steps by means of the implementation of the European Open Science Cloud (EOSC). The EOSC aims at providing researchers with a unique, federated and inclusive view of fit-for-purpose services, developed and operated by the diverse European research infrastructures, including the underlying e-Infrastructures. Consequently, the ultimate success of the EOSC heavily relies on the quality aspects of those services, such as their stability or functional suitability.

The meaning of **service** can be regarded from different perspectives. From an IT Service Management (ITSM) standpoint, such as the EOSC SMS process model, a service is devised as a means to "provide value to the customer". The same goal is shared by the DevOps paradigm, but in this case there is a more pragmatic vision the customer satisfaction is achieved through the continuous delivery of quality-assured services, with a shorter life cycle, as the final outcome of a comprehensive software development process.

The ITSM model has a broader focus. A service is an "intangible asset" that also includes additional activities such as customer engagement and support. Consequently, it is a much heavier process that might not be appropriate to be applicable for all types of services. The DevOps model, on the other

hand, narrows down the scope to meet the user expectations by acting exclusively on the quality features of the service, which is seen as an aggregate of software components in operation.

Purpose

This document provides an initial approach to service quality assessment, meant to be applied in the integration process of the services existing under the EOSC-Synergy project, which eventually will be accessible as part of the EOSC offerings.

The criteria compiled in this document favours a pragmatic and systematic approach that puts emphasis on the programmatic assessment of the quality conventions. To this end, the criteria herein compiled builds on the DevOps culture already established in the preceding Software Quality Assurance baseline document [5] to outline the set of good practices that seek the usability and reliability of services, and meet the user expectations in terms of functional requirements.

Contextualization of a Service

As a result, a **Service**, as conceived in this document, represent the following:

- Web service [WS] [7]:
  - A web service is an application or data source that is accessible via a standard web protocol (HTTP or HTTPS).
  - Web services are designed to communicate with other programs, rather than directly with users.
  - Most web services provide an API, or a set of functions and commands, that can be used to access the data.
- Web application [WApp] [8]:
  - A web application or “web app” is a software program that runs on a web server.
  - Web apps must be accessed through a web browser.
- Platform [Plat]: an integrated set of Web services, Web applications and software components.

Examples are: Web portals, Scientific portals and gateways, data repositories.

Goals

The herein proposed baseline harnesses the capabilities of the quality factors in the underlying software to lay out the principles for attaining quality in the enabled services within the EOSC context. According to this view, software quality is the foundation to shape user-centric, reliable and fit-for-purpose services.

The Service Quality baseline aims at fulfilling the following goals:

- Complement with a DevOps approach the existing approaches to assess and assure the quality and maturity of services within the EOSC, i.e. Technology Readiness Levels (TRLs) and EOSC Service Management System (SMS).

- Build trust on the users by strengthening the reliability and stability of the services, with a focus on the underlying software, thus ensuring a proper realization of the verification and validation processes.

- Ensure the functional suitability of the service by promoting testing techniques that check the compliance of the user requirements.
- Improve the usability by identifying the set of criteria that fosters the service adoption.
- Promote the automated validation of the service quality criteria.

Notational Conventions

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 1.

Quality Criteria

The following sections describe the quality conventions and best practices that apply to the development, operation and integration phases of a **Service** with a production infrastructure for research, such as the EOSC ecosystem. These guidelines rule the **Service** development and operation process within the framework of the EOSC-Synergy project.

Some of the criteria in this document is similar or based on the document "Software Quality Assurance baseline2 [5], for such cases the following tag is added to the criteria: [Ref.5-QC.XyNN] where QC.XyNN is the codename of the criteria in that document.

Integration Testing [SvcQC.Int]

Integration testing refers to the evaluation of the interactions among coupled **Service** or parts of a system that cooperate to achieve a given functionality.

- **[SvcQC.Int01]** Integration testing outcome MUST guarantee the overall operation of the **Service** whenever new functionality is involved. [Ref.5-QC.Int01].
- **[SvcQC.Int02]** Integration testing SHOULD be automated.
- **[SvcQC.Int03]** Ad-hoc pilot **Service** infrastructures and/or local testbeds MAY be used to cope with the integration testing requirements. [Ref.5-QC.Int04].

Scalability tests [SvcQC.Sca]

Scalability Testing is a non-functional test methodology in which an application’s performance is measured in terms of its ability to scale up or scale down the number of user requests or other such performance measure attributes [9].

- **[SvcQC.Sca01]**

Elasticity tests [SvcQC.Ela]

- **[SvcQC.Ela01]**

Acceptance and System tests [SvcQC.Acc]

Test of functionality of the **Service**, tests done before the **Service** enters into production. Tests for the public API.

- **[SvcQC.Acc01]**

Documentation [SvcQC.Doc]

- **[SvcQC.Doc01]** Documentation MUST be available online, easily findable and accessible. [Ref.5-QC.Doc03].

- **[SvcQC.Doc02]** Documentation SHOULD have a Persistent Identifier (PID).

- **[SvcQC.Doc03]** Documentation MUST be version controlled. [Ref.5-QC.Doc01.1].

- **[SvcQC.Doc04]** Documentation MUST be updated on new **Service** versions involving any change in the installation, configuration or behaviour of the **Service**. [Ref.5-QC.Doc04].

- **[SvcQC.Doc05]** Documentation MUST be updated whenever reported as inaccurate or unclear. [Ref.5-QC.Doc05].

- **[SvcQC.Doc06]** Documentation MUST have a non-software license.

- **[SvcQC.Doc07]** Documentation MUST be produced according to the target audience, varying according to the **Service** specification. The identified types of documentation and their RECOMMENDED content are:

  - **[SvcQC.Doc07.2]** Deployment and Administration. [Ref.5-QC.Doc06.3]:
    - Installation and configuration guides.
    - Service Reference Card, with the following RECOMMENDED content:
      - Brief functional description.
      - List of processes or daemons.
      - Init scripts and options.
      - List of configuration files, location and example or template.
      - Log files location and other useful audit information.
      - List of ports.
      - Service state information.
      - List of cron jobs.
      - Security information.
      - FAQs and troubleshooting.
  - **[SvcQC.Doc07.3]** User. [Ref.5-QC.Doc06.4]:
    - Detailed User Guide for the **Service**.
    - Public API documentation (if applicable).
    - Command-line (CLI) reference (if applicable).

Policies [SvcQC.Pol]

- **[SvcQC.Pol01]** The **Service** MUST include the following policy documents:
  - **[SvcQC.Pol01.1]** Acceptable Usage Policy (AUP).
  - **[SvcQC.Pol01.2]** Access policy.
  - **[SvcQC.Pol01.3]** Privacy policy.

Security [SvcQC.Sec]

- **[SvcQC.Sec01]** The **Service** public endpoints and APIs MUST be secured with encryption.

- **[SvcQC.Sec02]** Dynamic application security testing (DAST) [6] SHALL be performed from the outside, to the **Service** in an operation state, to look for security vulnerabilities (e.g. SQL injection, cross-site scripting, DDOS). [Ref.5-QC.Sec03].

- **[SvcQC.Sec03]** Manual penetration testing MAY be part of the application security verification effort. [Ref.5-QC.Sec04].

- **[SvcQC.Sec04]** The **Service** SHOULD have an Authentication mechanism.

- **[SvcQC.Sec05]** The **Service** MUST use strong ciphers for encryption.

- **[SvcQC.Sec06]** Credentials used in the **Service** MUST be signed by a recognized certification authority.

- **[SvcQC.Sec07]** The **Service** MUST validate the credentials and signatures.

Automated Deployment [SvcQC.Aud]

- **[SvcQC.Aud01]** Production-ready **Service** SHALL be deployed as a workable system with the minimal user or system administrator interaction leveraging Infrastructure as Code (IaC) tools.

Support [SvcQC.Sup]

- **[SvcQC.Sup01]** The **Service** MUST have a tracker or helpdesk for operational and users issues.

- **[SvcQC.Sup02]** The **Service** SHOULD have a tracker for the underlying software issues. [Ref.5-QC.Man01].

- **[SvcQC.Sup03]** The **Service** SHOULD include an Operational Level Agreement (OLA) with the infrastructure where it is integrated.

- **[SvcQC.Sup04]** The **Service** MAY include Service Level Agreement (SLA) with user communities.

Monitoring, Alerts, Metrics [SvcQC.MAM]

- **[SvcQC.MAM01]** The **Service** in an operational production state SHOULD be monitored.

    - **[SvcQC.MAM01.1]** The **Service** public endpoints SHOULD be monitored.
    - **[SvcQC.MAM01.2]** The **Service** public APIs SHOULD be monitored.
    - **[SvcQC.MAM01.3]** The **Service** SHOULD for security related criteria (DAST).

- **[SvcQC.MAM02]** The **Service** SHOULD have alerts.

- **[SvcQC.MAM03]** The **Service** SHOULD have metrics.

Glossary

API
: Application Programming Interface

CLI
: Command Line Interface

DAST

Dynamic Application Security Testing

EOSC

European Open Science Cloud

VCS

Version Control System

References

1. **Key words for use in RFCs to Indicate Requirement Levels**
   Scott Bradner
   (1997) https://www.ietf.org/rfc/rfc2119.txt

References

1. **Key words for use in RFCs to Indicate Requirement Levels**
   Scott Bradner
   (1997) https://www.ietf.org/rfc/rfc2119.txt