Software Maturity Levels

CESSDA Technical Framework Workplan phase 4, Deliverable 1 (a component of the Technical Architecture)

Status:	Public
Author:	John Shepherdson, CESSDA Platform Delivery Director
Contributors:	Members of CESSDA Technical Working Group
Date:	29 March 2019
Document:	Software Maturity Levels
Version:	Issue 03.00
DOI:	10.5281/zenodo.2614050

Version History

Version	Release Date	Comment
03.00	March 2019	Registered DOI via Zenodo
02.01	N/A	Added minimum and expected levels for CA8
02.00	December 2018	Following review by CESSDA CTO.
01.02	N/A	Moved to new document template. Extensive changes to sections CA5 and CA6. Minor modifications to CA11. Added section CA12. Now recognises importance of TRLs wrt to EOSC Hub.
01.01	N/A	Reviewed and updated all sections, including links. Global change from CRL to SML.
01.00	October 2017	Extracted from Technical Architecture v1.0.

Management Summary

Audience

Potential suppliers of software artefacts for the CESSDA technical research infrastructure. In the first instance, the CESSDA Service Providers, but potentially any software development organisation.

Purpose

This document is based on an extract from CESSDA Technical Architecture v1.0, May 2016 (the latest version of the latter can be found at [5]). It was created so that readers interested in CESSDA's Software Maturity Levels (SMLs) can find out more, without having to browse through the entire Technical Architecture document.

Usability is not only a political imperative of European Research Infrastructure Consortiums' need to maximise their return on investment, but is also essential for growth with limited funds and ongoing interoperability.

Mandating and checking the sustainability/usability of the software components of CESSDA's technical Research Infrastructure is essential if it is to strengthen and grow, however there are always risks attached, for example: how much effort is required to integrate it into the current technical framework, how will it be maintained, does it conform to the standards required? Therefore the need to measure the maturity of software designed for use by CESSDA is essential to ensure the quality of the technical Research Infrastructure is maintained.

Table of Contents

Glossary

Acronym	Expansion	Description
API	Application Programming Interface	*"In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software and applications."* Source: Wikipedia: Application programming interface
CI	Continuous Integration	*"The practice, in software engineering, of merging all developer working copies to a shared mainline several times a day."* Source: Wikipedia: Continuous Integration
RI	[CESSDA] Research Infrastructure	"*A seamless social science data archive service for the whole of the European Research Area (ERA), which is capable of supporting the research needs of the next generation of social scientists wherever in Europe they may be, or beyond."* Source: CESSDA SaW project overview
-	Software artefacts	Software products, applications, services, components.

Scope

CESSDA Software Maturity Levels.

Summary

This document lays out an approach for assessing the maturity of the components of the technical Research Infrastructure (RI), so that over time CESSDA can mandate minimum levels that Service Providers (SPs) and others have to meet as a prerequisite to supplying software artefacts for the RI.

Software Maturity Levels

Objective

Mandating the sustainability/usability of the software components of the technical Research Infrastructure is essential if CESSDA is to strengthen and grow. There are risks attached to adopting software systems and components, for example: how much effort is required to integrate it into the current technical framework?, how will it be maintained?, does it conform to the standards required? Therefore the need to measure the maturity of software used within CESSDA is essential to ensure the quality of the technical Research Infrastructure is maintained. Reuse Readiness Levels (RRLs) [1], as developed by NASA Earth Science Data Systems, form the basis upon which the CESSDA software maturity assessments are made. Usability is not only a political imperative of research infrastructures as they need to demonstrate a return on investment, but is also essential for growth with limited funds and ongoing interoperability.

Background

The measurement of maturity can be achieved in various ways. Services (and service management) use Capability Maturity Modelling, for example FitSM [2]. A method commonly used for technology is the 9 point Technology Readiness Levels (TRLs) scale [3], however this does not address usability, which is essential for the development of CESSDA's technical Research Infrastructure. RRLs address this gap in the assessment of the maturity of software artefacts. Note that the EU adopted TRLs as part of the H2020 programme [4] and both FitSM and TRLs have been subsequently adopted by the EOSC-hub, which mandates that TRL Level 8 is the minimum acceptable for a system to be considered production-ready by them. Interestingly, both RRLs and TRLs were devised by and are widely used by NASA.

RRLs define 9 levels of maturity ranging from 1 ("software is not recommended for reuse") to the most mature 9 ("software is being reused by many classes of

users…"), however if we are to align a software maturity assessment model with the CESSDA Capability Development Model (CESSDA-CDM) for Service Providers (CESSDA-SaW WP3, deliverable D3.1) then it would be prudent to reduce the number of levels to 5. The following tables provides a mapping from the NASA RRL via CMM to CESSDA Software Maturity Levels (SMLs). Note that in the CESSDA context, 'reuse' becomes 'use' as typically components are commissioned, rather than re-purposed.

Table 1 shows the correspondence between the various levels in the RRL, CML and SML scales. Given that one of CESSDA's goals is to have its tools and services listed in the EOSC Market Place, the requirements imposed by CESSDA will be continuously adopted to ensure compliance.

Reuse Readiness Levels	*Capability Maturity Levels*	*CESSDA Software maturity levels*
RRL1 - Limited reusability; not recommended for reuse RRL2 - Initial reusability; reuse not practical	CMM1 Initial (chaotic, ad hoc or reactive)	SML1 - Initial usability; software use is not recommended.
RRL3 - Basic reusability; might be reusable by skilled users at substantial effort, cost, and risk. RRL4 - Reuse is possible; might be reused by most users with some effort, cost, and risk.	CMM2 Repeatable (active)	SML2 - Use is feasible; the software can be used by skilled personnel but with considerable effort, cost and risk. Assessment of effort, cost and risk shall be made before use is attempted.
RRL5 - Reuse is possible; might be reused by most users with some effort, cost, and risk.	CMM3 Defined (proactive)	SML3 - Use is possible by most users; with some effort, cost, and risk. A risk assessment should be made before use.
RRL6 - Software is reusable; the software can be reused by most users although there may be some cost and risk. RRL7 - Software is highly reusable; the software can be reused by most users with minimum cost and risk.	CMM4 Managed	SML4 - Software is usable; with little effort, cost, and risk.
RRL8 - Demonstrated local reusability; the software has been reused by multiple users. RRL9 - Demonstrated extensive reusability; the software is being reused by many classes of users over a wide range of systems.	CMM5 Optimised	SML5 - Demonstrable usability; there is clear evidence that the software is widely used by many users.

Table 1: Correspondence of levels in RRL, CML and SML scales

CESSDA Software maturity levels

The software maturity levels provide guidance on what minimum, expected and excellent standards look like, and will be used to evaluate the products produced by SPs.

CA1: Documentation

The NASA RRL scale looks at the overall document suite, but it would be both pragmatic and useful to break that down into three: end-user, operational and development documents. That allows phased delivery (and scoring) of documentation. A mean score can be calculated, if required - e.g. in cases where only operational and development documentation are required for acceptance.

- Minimum standard - SML2
- Expected standard - SML3
- Excellent standard - SML5

CA1.1: End-user Documentation

SML1 - Initial usability: Partial or no external documentation available; Documentation is insufficient to gain an understanding of the functionality of the software even for an experienced user.

SML2 - Use is feasible: There is external documentation that is accessible and sufficient for an expert user to configure and use the software for the user's individual needs. Terminology and methodology is not explained.

SML3 - Use is possible by most users: There is a user manual that can guide a reasonably skilled user through use and customisation of the software to the user's individual requirements. Documentation is consistent with current version of the software.

SML4 - Software is usable: There are examples of walk-through tutorials, how-to guides, demonstrations of various use case customisations if needed for the user's individual needs. Documentation is consistent with current version of the software.

SML5 - Demonstrable usability: User materials and tutorials can be used as training resources. There is detailed in-software contextual user support documentation. Documentation is consistent with current version of the software. User created documentation and comments form part of the documentation available.

CA1.2: Operational Documentation

SML1 - Initial usability: Partial or no external documentation available; Documentation is insufficient to gain an understanding for the deployment and configuration of the software without additional technical support or significant investment of time.

SML2 - Use is feasible: There is external documentation that is accessible and sufficient for an expert to deploy and configure the software for all users. Terminology and methodology is not fully explained. Exception and failure messages are not fully explained.

SML3 - Use is possible by most users: There is a deployment and configuration manual that can guide an experienced operational user through deployment, management and configuration of the software. Exception and failure messages are explained, but descriptions of solutions are not available. Documentation is consistent with current version of the software.

SML4 - Software is usable: There are examples of walk-through tutorials, demonstrations of various configurations if needed. Exception and failure messages are fully explained, and solutions are documented. Upgrade workflows are fully documented, if needed. Documentation is consistent with current version of the software.

SML5 - Demonstrable usability: Documentation is appropriate for different categories of deployment and management of the software. Deployment and configuration demonstrations, materials and tutorials can be used to teach other users. Documentation is consistent with current version of the software. User created documentation and comments form part of the documentation available.

CA1.3: Development Documentation

SML1 - Initial usability: Partial or no external documentation available of the application program interface (API); no, partial or inconsistently commented source code. Documentation is insufficient to gain an understanding of the software/service functionality without significant investment of time.

SML2 - Use is feasible: There is external documentation that describes public API functionality and is sufficient to be used by an experienced developer. If available, source code is consistently and clearly commented. Source code naming conventions are adhered to and consistent.

SML3 - Use is possible by most users: There is external documentation that describes all API functionality, human computer interface (HCI) and code

modules, which is sufficient to be used by any developer. An extension guide provides information on how to customise and add to the software, add plug-ins. Internal and external documentation are sufficient to allow an experienced developer to understand program flow and logic with moderate effort.

SML4 - Software is usable: There is a guide to the documentation and how to use it. There are examples of how to use the API & HCI for different use cases, and materials & tutorials can be used to train other developers. Documentation is consistent with current version of the software.

SML5 - Demonstrable usability: All stages of the software development lifecycle are fully documented, including design, testing and future improvement planning. Documentation is appropriate for different categories of users. Documentation describes the current version of the software. User created documentation and comments form part of the documentation available.

CA2: Intellectual Property

- Minimum standard - SML2
- Expected standard - SML3
- Excellent standard - SML5

SML1 - Initial usability: Software developers have been identified and their responsibilities have been determined. Relevant policies of developer organisation(s) (or developers) have been reviewed for applicability to intellectual property rights. There may be evidence of a draft intellectual property rights agreement that would result from cooperative activities with other developer organisations (or developers). Rights are not specified internally in the source code or externally in documentation. Usage rights or limitations have not been specified.

SML2 - Use is feasible:
Developer organisation(s) (or developers) have an agreement that addresses any potential conflicts in the proposed intellectual property rights and responsibilities for development. A limited rights statement has been drafted, and applied inconsistently in documentation and source code. Developer organisation(s) (or developers) may be contacted to negotiate rights for use.

SML3 - Use is possible by most users:
Agreements on development responsibilities, the list of developers, a recommended citation, and intellectual property rights statements, offering limited rights for use, are available, perhaps upon request, for review. Developer organisations(s) (or developers) may be contacted through a single point to obtain formal statements on restricted rights or to negotiate

additional rights.

SML4 - Software is usable:
There is evidence that all developer organisation(s) (or developers) have confirmed that the list of developers, recommended citation, and intellectual property rights statements, including limited rights for use, in the software source code, documentation, and in the expression of the software upon execution, conform to their institutional policies and agreements. These include any legal language that has been approved by all parties or their representatives, machine-readable code expressing intellectual property, and concise statements in language that can be understood by laypersons, such as a pre-written, recognisable license. Brief statements are available describing limited rights, restrictions, and conditions for use. Developer organisations(s) (or developers) may be contacted through a single point to obtain formal statements on restricted rights or to negotiate additional rights.

SML5 - Demonstrable usability: There are multiple statements embedded into the software product describing unrestricted rights and any conditions for use, including commercial and non-commercial use, and the recommended citation. The list of developers is embedded in the source code of the product, in the documentation, and in the expression of the software upon execution. The intellectual property rights statements are expressed in legal language, machine-readable code, and in concise statements in language that can be understood by laypersons, such as a pre-written, recognisable license.

CA3: Extensibility

- Minimum standard - SML3
- Expected standard - SML4
- Excellent standard - SML5

SML1 - Initial usability: The software was not designed with extensibility in mind, so there is either no ability to extend or modify program behavior, or it is very difficult to do, even for usages similar to those of the software core design; execution parameters cannot be changed. There is no, or limited, availability of the source code; the logical flow of code may be hard to follow, with little to no cohesion.

SML2 - Use is feasible: There is some consideration to extensibility, but that may only exist for a limited number of use cases, through use of methods such as object-oriented design or other tools which provide logical cohesion. Some extensibility is possible through configuration changes; isolation of configuration parameters and constants in clearly identified sections of source code; and/or limited opportunity for software modification.
Where source code is available, there is evidence that there is effective use of

programming practices designed to enable use, such as object oriented design.

SML3 - Use is possible by most users: Future extensibility is designed into the system for a moderate range of use cases. The procedures for extending the software are defined, whether by source code modification or through the provision of some type of extension functionality (e.g., callback hooks or scripting capabilities). Where source code modification is part of the extension plan, the software is well-structured, has a moderate to high level of cohesion, and has configuration elements clearly separated from logic and display elements.

SML4 - Software is usable: The extensibility capability for the software is well defined, broad range of use cases, providing many points of extensibility. A detailed extensibility plan is publically available and is sufficient to allow an experienced developer to become familiar with the project to extend the software in a reasonable amount of time. Documentation should include clear information about the range of use cases to which the software can be extended as well as potential limitations on expansion.

There is evidence that the software has been extended and applied to a context to the original. This extension may have been done by another group or project, using extension documentation, but may have involved ad hoc and substantial assistance from the original development team.

SML5 - Demonstrable usability: There is evidence that the software has been extended externally by users outside of the original development group using existing documentation only. There is a clear approach for modifying and extending features across a in multiple scenarios, with specific documentation and features to allow the building of extensions which are used across a range of domains by multiple user groups. There may be a library available of user-generated content for extensions and user generated documentation on extension is also available.

CA4: Modularity

- Minimum standard - SML3
- Expected standard - SML4
- Excellent standard - SML5

SML1 - Initial usability: There is evidence that the source code was written with no designs or consideration for organising the code in terms of functionality for modularity or use. It may have been a demonstrator or pilot project.

SML2 - Use is feasible: There is no distinction between generic and solution-specific functionality. The source code is organised into a primary

system that provides general functionality and one or two subsystems that each provide multiple, unrelated, functions; code within each module contains many independent logical paths. The architecture is closed with only a few internal functions accessible by external programs through the primary system.

SML3 - Use is possible by most users: There is evidence that the architecture is open, with full structuring into individual components that provide functions or services to outside entities (i.e., open architecture); internal functions or services documented, but not consistently; modules have been created for generic functions, but modules have not been created for all of the specified functions; code within each module contains many independent logical paths.

SML4 - Software is usable: There is clear organisation of all components into libraries or service registries with consistent documentation of all libraries as APIs or standard web service interfaces. Modules have been created for all specified functions and organised into libraries with consistent features within interfaces, but source code within each module may contain many independent logical paths.

SML5 - Demonstrable usability: It is evident that all functions and data are encapsulated into objects or accessible through web service interfaces. There is consistent error handling with meaningful messages and advice, and use of generic extensions to program languages for stronger type checking and compilation-time error checking. Services are available externally and code within each module contains few independent logical paths.

CA5: Packaging

- Minimum standard - SML3
- Expected standard - SML4
- Excellent standard - SML5

SML1 - Initial usability: Only source code or executable available; i.e. no packaging to create a package or container. There is no, or incomplete, configuration documentation and no deployment facility is available. Even an experienced user may have difficulties deploying the software.

SML2 - Use is feasible: Software includes package or container creation. Detailed configuration instructions are available and deployment is possible for an experienced user.

SML3 - Use is possible by most users: The packaged/containerised software is easily configurable for different contexts as locations of resources (files,

directories, URLs) are configurable. All configuration-specific information is centralised.

SML4 - Software is usable: Configurable scripts are available to deploy the packaged/containerised software from the command line. Versions of deployed software can be upgraded/rolled back from the command line. Data and/or index files can be restored from the command line.

SML5 - Demonstrable usability: A Continuous Integration server job (or equivalent) is available to deploy the packaged/containerised software. Administrators are notified if deployment fails. Versions of deployed software can be upgraded/rolled back from a Continuous Integration server job (or equivalent). Data and/or index files can be restored from a Continuous Integration server job (or equivalent).

CA6: Portability

- Minimum standard - SML3
- Expected standard - SML4
- Excellent standard - SML5

SML1 - Initial usability: Porting to the target platform as a whole is not feasible (e.g. due to licensing or dependencies not available for the target platform) or prohibitively expensive.

SML2 - Use is feasible: The complete source code is available, without external dependencies that are not portable, but the software cannot be ported to the target platform without significant changes to the software or the target context. Porting to the target platform will require significant effort.

SML3 - Use is possible by most users: The software is moderately portable to the target platform. The software can be ported with only relatively small changes necessary to the context or the software itself. Porting to the target platform would be relatively easy to implement.

SML4 - Software is usable: The software is highly portable to the target platform. No changes to the software are necessary and the effort to port the software to the target platform is minimal.

SML5 - Demonstrable usability: The software is completely portable to the target platform. In theory at least, the software will run on the target platform provided it is packaged/containerised.

CA7: Standards Compliance

- Minimum standard - SML1

- Expected standard - SML3
- Excellent standard - SML5

SML1 - Initial usability: The software and software development process comply, at least in part, with locally defined standards and best practices. The standards may be internally or externally described, but are implemented with modifications to meet local conditions. There may be little or no documented evidence of standards used, but it maybe possible to infer this from the software consistency of functionality and interfaces.

SML2 - Use is feasible: The software and software development process endeavour to comply with recognised standards or widely used best practices, but without verification or testing and may not be complete. There maybe some documented evidence that standards are used, but it may not be complete.

SML3 - Use is possible by most users: The software and software development process comply with open, recognised or proprietary standards, but there is incomplete verification of compliance. Compliance to recognised standards has be tested but this may not be for all components. There is documented evidence of standards being used, but not of the verification of components.

SML4 - Software is usable: The software and software development process comply with recognised and proprietary standards. Compliance with these standards has been verified through testing for all components. Documented evidence for selected standards and the verification through testing is available.

SML5 - Demonstrable usability: Compliance with open or internationally recognised standards for the software and software development process, is evident and documented, and verified through testing of all components. Ideally independent verification is documented through regular testing and certification from an independent group.

CA8: Support

Assessment may not be required at this stage. Minimum and expected standard levels to be defined in a future version of this document.

- Minimum standard - SML1
- Expected standard - SML1
- Excellent standard - SML5

SML1 - Initial usability: There is known contact information available for the developer organisation(s) and there is a willingness to provide minimal,

occasional support without guarantees. It may not be possible for an end-user to use the software without some support.

SML2 - Use is feasible:
The developer organisation(s) respond to reported issues with updates/patches that are usually made available in a reasonably timely fashion. Some support is available, but may be intermittent and without guarantees of continuation. There is evidence of an informal user community that provides answers, for example, via a mailing list or bulletin board. Documentation and source code availability may be sufficient for an experienced user (developer, operations or end-user) to not require extensive support.

SML3 - Use is possible by most users:
Support is centralised in a website containing relevant resources, answers to FAQ, other useful information and a community support question & answer area (e.g. bulletin or message/discussion board). There is evidence that the developer organisation(s) occasionally engage with users in the community support area. There are regular timetabled releases of updates/patches that are made available and urgent (perhaps due to security issues) update/patches in a timely fashion.There is no opportunity to obtain a support Service Level Agreement (SLA) with the developer(s) or a third party.

SML4 - Software is usable:
There is organised and clearly defined support by the developer with an email helpdesk and additional documentation such as case studies and other detailed information for a range of user communities (developers, operations staff, and end-users). There is explicit evidence that no continuity of support is implied. It maybe possible to negotiate an SLA for support, but this is not a standard offering of the developer organisation(s).

SML5 - Demonstrable usability: The support by the organisation(s) is clearly defined with frequent and timely updates, releases, etc., responding to the needs of the user communities, as well as consolidation of changes by the community. There is a staffed telephone/email helpdesk available as well as a maintained website. Discussion groups are active and include regular input from the developer(s) and developer organisation(s). There is evidence that continuity of support is implied. Support may be free or fee-based via a support Service Level Agreement (SLA) with the developer(s) or a third party.

CA9: Verification and Testing

- Minimum standard - SML2
- Expected standard - SML3
- Excellent standard - SML5

*SML1 - Initial usability: S*oftware application formulated and unit testing performed.

SML2 - Use is feasible: Software application demonstrated and tested in a laboratory context. Testing includes testing for error conditions and proof of handling of unknown input.

SML3 - Use is possible by most users: Software application demonstrated, tested and validated in a relevant context.

SML4 - Software is usable: Actual software application "qualified" through test and demonstration (meets requirements) and successfully delivered.

SML5 - Demonstrable usability: Actual software application tested and validated through successful use of application output.

CA10: Security

- Minimum standard - SML1
- Expected standard - SML3
- Excellent standard - SML5

SML1 - Initial usability: Security was addressed in the development phases up to and including design.

SML2 - Use is feasible: Security was addressed in the development phases up to and including implementation. Developers have undertaken appropriate Security training.

SML3 - Use is possible by most users: Security was addressed in the development phases up to and including implementation.

SML4 - Software is usable: Security was addressed in the development phases up to and including verification and testing.

SML5 - Demonstrable usability: Security was addressed in the development phases up to and including product release.

CA11: Internationalisation and Localisation

- Minimum standard - SML1
- Expected standard - SML3
- Excellent standard - SML5

SML1 - Initial usability: Internationalization and Localization not addressed.

SML2 - Use is feasible: Software is locale aware.

SML3 - Use is possible by most users: Content to localize is stored in resource/property files and/or and external data store.

SML4 - Software is usable: Content to localize (text, layout, graphics and multimedia, keyboard shortcuts, fonts, locale data and character sets, build process) has been internationalized.

SML5 - Demonstrable usability: Software has been tested with multiple pseudo or genuine translations.

CA12: Authentication and Authorisation

- Minimum standard - SML2
- Expected standard - SML3
- Excellent standard - SML5

SML1 - Initial usability: Single user mode, no fine-grained authorization.

SML2 - Use is feasible: Multiple users, no fine-grained authorization (i.e. all users have same role).

SML3 - Use is possible by most users: Multiple users, fine-grained authorization (i.e. users can have different roles).

SML4 - Software is usable: Authentication is externalised (e.g. via social login, federated access management etc) and authorisation is fine-grained (supporting groups and roles) and can be externalised (e.g. via ldap).

SML5 - Demonstrable usability: Full rights management by users, sharing/delegation of permissions/access to individual data from within the system.

References

[1] Nasa Reuse Readiness Levels (RRLs)
https://earthdata.nasa.gov/files/RRLs_v1.0.pdf
(https://earthdata.nasa.gov/library)

[2] The FitSM Standard http://fitsm.itemo.org/fitsm

[3] Wikipedia: Technology readiness level
https://en.wikipedia.org/wiki/Technology_readiness_level#In_the_European_Union

[4] HORIZON 2020 WORK PROGRAMME 2016– 2017, 20. General Annexes, G. EC Technology Readiness Levels
http://ec.europa.eu/research/participants/data/ref/h2020/other/wp/2016-2017/annexes/h2020-wp1617-annex-ga_en.pdf

[5] CESSDA Technical Architecture v4, bit.ly/tech_arch4_0