

CESSDA's Software Maturity Model

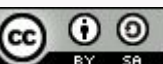
March 2019

John Shepherdson
CESSDA Platform Delivery Director

 cessda.eu

 [@CESSDA_Data](https://twitter.com/CESSDA_Data)

DOI: [10.5281/zenodo.2588385](https://doi.org/10.5281/zenodo.2588385)



Software Maturity Model

Outline

- Technical Infrastructure
- Protection of Assets
- CESSDA's common interoperability characteristics
- Quality control
- Software acceptance criteria
- Software maturity levels
- Question

Technical Infrastructure

Was bottom up

- where is expertise, source code, documentation? ...

Opportunity: Transition to CESSDA ERIC

- Greenfield site for technical development/deployment
- Central hosting, monitoring and management
- Guidelines, policies and procedures

Using Bitbucket cloud to host source code
and Google Cloud Platform to build, test and deploy

Protection of Assets

CESSDA Bitbucket Code repositories

Ensure CESSDA has access to and IPR for

- source code
- configuration files
- technical documentation

that underpin its products and services

CESSDA Code Repositories

- Form for write access
 - Contributor license agreement
- Working towards Open Source
 - Apache 2 license
 - Separate application code from deployment scripts

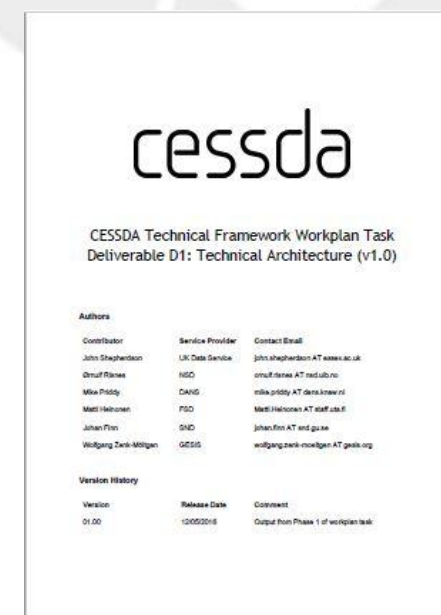
Technical Framework

Guide development of various (software) tools and services for CESSDA Research Infrastructure

- Promote good practice for software development
- Meet common interoperability characteristics

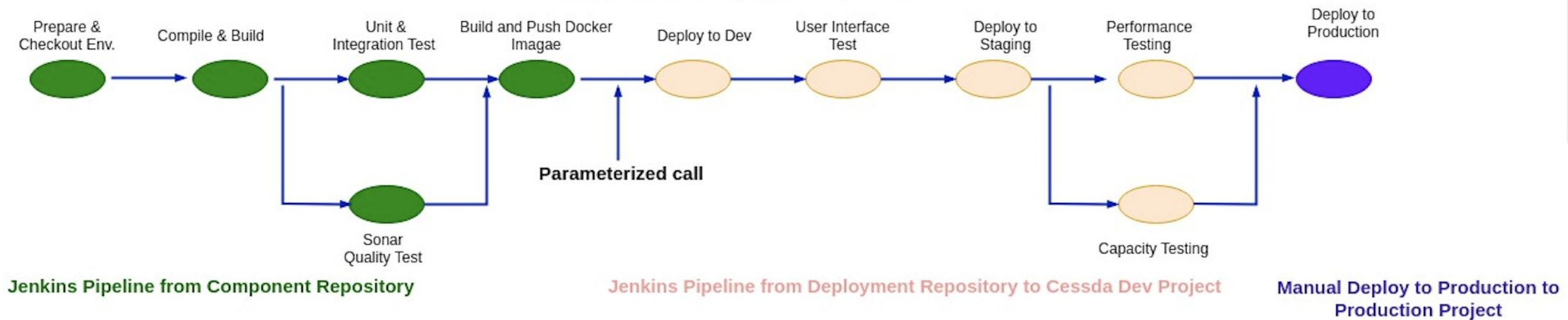
Infrastructure for Development, Staging and Production

- Harmonise development tool chain for SPs
- Apply consistent set of tests (SonarQube, Selenium, JMeter ...)



Jenkins

Deployment Pipeline



Automated acceptance testing (up to a point):

- Jenkins pipelines for CI/CD
-
-
-

Common Interoperability Characteristics

1. **Loosely coupled but coordinated** - enable Service Providers to retain independence, yet fully interact in an integrated service
2. **Sustainable** - enable medium and long term investment and business change decisions to be made
3. **Extensible** - enable additional services to be built on or around it, including adapting to changing functional requirements over time

Common Interoperability Characteristics

4. **Maintainable** - enable components to be updated when IT specifications change
5. **Standards based** - enable the coordinated and planned change to all the coupled, but coordinated, services

Quality Control

How to achieve?

Set standards and guidelines

- REST APIs c/w API design standards
- Technical Architecture
- Software acceptance criteria (Software Maturity Levels)
- Common development environment

Quality Control

Software Maturity Levels

- ensure technical quality of the research infrastructure
- guidance on standards
 - minimum, expected and excellent
- originally based on NASA's 9 RRLs

NASA Earth Science Data Systems Software Reuse Working Group (2010). Reuse Readiness Levels (RRLs), Version 1.0. April 30, 2010. Available: <http://www.esdswg.org/softwarereuse/Resources/rrls/>

- revised in light of 'Capability Development Model' from CESSDA SaW project

Quality Control

Standards and guidelines

- Adoption of 12 Factor App principles
- [User Experience guide](#)
- [Software Adoption Policy](#)
- [Software Adoption Procedure](#)
- Contributor's License Agreement

Loosely coupled but coordinated

Adopt (micro) services architecture based on RESTful web service APIs

- provides a mechanism for reusing and combining software artefacts

See also 12 factor app, number 7 ([Port binding - Export services via port binding](#))

Sustainable

The provision of common standards

- Technical Architecture document

Common development and test environment

- via the technical infrastructure

Deployment environment

- via extensions to the technical infrastructure

Central source-code repository

See also 12 factor app, number 1 ([Codebase - One codebase tracked in revision control, many deploys](#))

Extensible

Service API is key

- Integration point for new services
- Combination point for building new features

Version and support two versions simultaneously

- Allows services to evolve, without breaking contract provided to consumers

See also 12 factor app, number 8 (Concurrency - Scale out via the process model)

See also 12 factor app, number 9 (Disposability - Maximize robustness with fast startup and graceful shutdown)

Maintainable

Again, service API is key

- implementation of a service can be changed as required, to take advantage of developments in software technology
- location of services can be changed as required, to take advantage of developments in hardware technology

See also 12 factor app, number 2 ([Dependencies - Explicitly declare and isolate dependencies](#))

Standards Based

- Provision of common architectural standards (via Technical Architecture)
- A consistent (in both the calling and return structures and formats) and versioned API

See also 12 factor app, number 4 ([Backing services - Treat backing services as attached resources](#))

Acceptance Criteria

1. Documentation (end user, operational, developer)
2. Intellectual property issues
3. Extensibility
4. Modularity
5. Packaging
6. Portability
7. Standards Compliance
8. Support
9. Verification and testing
10. Security (by design)
11. Internationalisation and Localization
12. Authentication and Authorisation

Acceptance Criteria

5 levels for each

- descriptive text to aid decision making
- plus 'not applicable'

Minimum, expected and excellent level indicated

- minimum and expected will change over time

Software Maturity Levels

Colour	Meaning
Orange	Minimum standard
Yellow	Expected standard
Green	Excellent standard

See also [Software Maturity Levels](#) document

Software Maturity Levels

L1

Intellectual Property (IPAT)	Extensibility (EXT)	Modularity (MOD)	Reliability (REL)	Portability (POR)	Standards Compliance (SCAT)	Support (SUP)	Verification and Testing (VAT)	Security (SEC)	Internationalization and Localization (ICAT)
Software developers have been identified and their responsibilities have been determined. Relevant policies of developer organizations (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 organizations (or developers). Rights are not specified internally in the source code or externally in documentation; use rights or limitations have not been specified.	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 users similar to those of the software. Some design parameter changes 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.	There is evidence that the source code was written with no design or consideration for organizing the code in terms of functionality for modularity or use. It may have been a demonstration or pilot project.	Only source code or executable available, i.e., no packaging. There is no or incomplete installation documentation and no auto-build/installation facility is available. There are experienced users who have difficulties installing the software.	The software as a whole is not portable, however, if there is source code provided with sufficient detail, code not and custom-built documentation is available. There are some portable elements may be portable. Portable elements are provided for a specific platform, but there is no useful information on porting. Porting as a whole is not feasible (e.g., due to technology or portability dependent).	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 developed, but are consistent with best practices. There may be some or no documented evidence of standards used, but it is feasible to infer this from the software consistency of functionality and structure.	There is known contact information available for the developer or publisher and there is a willingness to provide minimal, occasional support without parameters. It may not be possible for an end user to use the software without some support.	Software application form tested and unit testing performed.	Security was addressed in the development phase up to and including design.	Internationalization and Localization not addressed.
Developer organizations (or developers) have an agreement that addresses the potential conflicts of the associated intellectual property rights and responsibilities for development. A limited rights statement has been drafted, and applied consistently in documentation and source code. Developer organizations (or developers) may be contacted to negotiate rights for use.	There is some consideration to extensibility, but that may only need 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 is clearly identified sections of source code; and/or limited opportunity for software modification.	There is no distinction between generic and solution-specific functionality. The source code is organized into a primary system that provides general functionality and one or two sub-systems that each provide multiple, unrelated, functions; code within each module contains many independent logical parts. The architecture is shared with only a few internal functions available by external programs through the primary system.	Software includes auto-build features, but is only portable for one operating system. Detailed installation instructions are available and building for other operating systems is available for an experienced user.	The complete source code is available, without external dependencies that are not portable, but the software cannot be ported without significant changes to the software or the target platform. Documentation on porting is insufficient, and should be taken into consideration when assessing the effort required. Cost-benefit and risk analysis, compared to rewriting for the new platform and size class, should be undertaken before porting is considered, and only initiated if the cost/benefit of using the software slightly outweighs the cost of developing new software. Porting will nonetheless require significant effort.	The software and software development process compliance is complete with recognized standards or widely used best practices, but without verification or testing and may not be complete. There may be some documented evidence that standards are used, but it may not be complete.	The developer organizations (or developers) respond to repeated requests 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 confidentiality. 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 to develop operations or end-user to not require extensive support.	Software application development tested and tested in a laboratory context. Testing includes testing for error conditions and growth of handling of software input.	Security was addressed in the development phase up to and including implementation. Developers have undertaken appropriate security training.	Software is locale aware.
Agreements on developer responsibilities, the list of developers, a recommended status, and intellectual property rights statements, offering limited rights for use, are available. Software users request, for review. Developer organizations (or developers) may be contacted through a single point to obtain formal statements on intellectual rights or to negotiate additional rights.	Some 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., add-on tools or scripting capabilities). Where source code modification is part of the extension process, the software is well-encapsulated, has a moderate to high level of cohesion, and configuration elements clearly separated from logic and data elements.	There is evidence that the architecture is open, with full documentation of the architecture that provides functions or services to outside entities (i.e., users/architectural functions or services documented, but not necessarily available for reuse). The architecture has been designed for generic functions, but modules have not been created for all the specified functions; code within each module contains many independent logical parts.	The software is easily configurable for different contexts such as, features or modules (i.e., functions, files) are configurable. A configuration-specific information is available.	The software is reasonably portable. The software can be ported with only minor modifications (e.g., changes to the system or the software itself). Documentation on porting is sufficient, and it is feasible to infer this from the software consistency of functionality and structure. Some documentation on porting is available, but it may not be complete. There is documented evidence of standards being used, but not of the verification of compliance.	The software and software development process comply with recognized standards or widely used best practices, but there is incomplete verification of compliance. Standards have been used for all components. There is documented evidence of standards being used, but not of the verification of compliance.	Support is centralized in a website containing relevant resources, answers to FAQs, other useful information and a community support question & answer area (e.g., bulletin board/message/forum board). There is evidence that the developer organizations (or developers) regularly engage with users in the community support area. There are regular releases of updates/patches that are made available and urgent responses due to security issues. There is no evidence to obtain a support Service Level Agreement (SLA) with the developer (or a third party).	Software application development tested and validated in a relevant context.	Security was addressed in the development phase up to and including implementation.	Content is locale aware.
There is evidence that all developer organizations (or developers) have confirmed that the list of developers, recommended status, and intellectual property rights statements, including limited rights for use, in the software source code, documentation, and in the operation of the software upon installation, conform to their institutional policies and agreements. There is evidence of a legal language that has been approved by attorneys or their representatives, machine-readable code expressing intellectual property, and source statements in language that can be understood by lawyers, such as a patent, copyright, license, or other statements are available describing limited rights, restrictions, and conditions for use. Developer organizations (or developers) may be contacted through a single point to obtain formal statements on intellectual rights or to negotiate additional rights.	The extensibility capability for the software is well defined, based on a range of use cases, providing some points of extensibility. A detailed extensibility plan is publicly 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 to extension. There is evidence that the software has been extended and applied to a context to the original. The extension may have been done by another group or group, using extension documentation, but may have involved ad hoc and substantial evidence from the original development team.	There is a clear organization of all components into libraries or service modules with consistent documentation of all the use cases as APIs or standard web service interfaces. Modules have been created for all specified functions and organized into libraries with consistent features within interfaces, but source code within each module may contain many independent logical parts.	On installation there is a self-detect and auto-build for more than one supported platform with OS detection configuration. The available (along with) the configuration file for platform-specific testing. Validation is reasonably straightforward with the possibility to recover all created files and configurations.	The software is highly portable. The software can be ported to all but the most obscure or obsolete systems without modification. The documentation is complete and thorough. No changes to the software are necessary and the effort to port the software is minimal.	The software and software development process comply with recognized and proprietary standards. Compliance with these standards has been verified through testing for all components. Documented evidence for selected standards with the verification through testing is available.	There is organized and clearly defined support by the developer with an email helpline and additional documentation such as user guides 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 may be possible to negotiate an SLA for support, but this is not a standard offering of the developer organizations.	Actual software application "qualified" through test and demonstration (i.e., requirements) and successfully delivered.	Security was addressed in the development phase up to and including verification and testing.	Content is locale aware, broad, graphical and multimedia, keyboard shortcuts, fonts, locale data and character sets, build process has been internationalized.
There are multiple statements indicating that the software project describing source code, rights and any conditions for use, including commercial and non-commercial use, and the recommended status, the list of developers is internationalized. The source code, the product, the documentation, and in the operation of the software upon installation, the intellectual property rights statements are expressed in legal language, machine-readable code, and in source statements in language that can be understood by lawyers, such as a patent, copyright, license.	There is evidence that the software has been extended/modified by users outside of the original development group using existing documentation only. There is a clear approach for modifying and extending features, source code multiple times, with a specific documentation and business to allow the building of extensions which are well tested a range of developers by multiple user groups. There is a clear evidence of user-generated content for extensions and user-generated documentation as evidence to user practices.	It is evident that all functions and data are encapsulated into objects or modules through well-defined interfaces. There is a consistent use of naming with meaningful messages and data and use of generic interfaces to program languages for a single type checking and compilation that is so on checking. Functions are available across well-defined and well-defined module boundaries independent logical parts.	A user interface enables the user to manage the software to build, configure, and install the software. An interface is also available.	The software is consistently portable. The software can be ported to all but the most obscure or obsolete systems with only minor modifications. The documentation is complete and thorough. No changes to the software are necessary and the effort to port the software is minimal.	Compliance with open or internationally recognized standards for the software and software development process, is verified and documented, and verified through testing of all components. There is documented evidence of standards being used, but not of the verification of compliance.	The support by the organizations is clearly defined with formal and formal policies, procedures, etc., regarding the use of the software community, as well as the availability of changes by the community. There is a defined responsibility for the software as well as a maintenance website. User-generated content and user-generated content from the developer and developer organizations. There is evidence that continuity of support is implied. Support may be free or negotiated via a support Service Level Agreement (SLA) with the developer (or a third party).	Actual software application tested and validated through manual or automated testing and successfully delivered.	Security was addressed in the development phase up to and including product release.	Software has been tested with internationalization.

L5

SML - Security

0 - Topic area is not relevant

1 - Security was addressed in the development phases up to and including design (MINIMUM)

2 - Security was addressed in the development phases up to and including implementation.

3 - Security was addressed in the development phases up to and including implementation. Developers have undertaken appropriate Security training. (EXPECTED)

4 - Security was addressed in the development phases up to and including verification and testing

5 - Security was addressed in the development phases up to and including product release (EXCELLENT)

Software Maturity Levels

1 - Initial usability; software use is not recommended.

2 - Use is feasible; the software can be used by skilled personnel but with considerable effort, cost and risk.

3 - Use is possible by most users; with some effort, cost, and risk. A risk assessment should be made before use.

4 - Software is usable; with little effort, cost, and risk.

5 - Demonstrable usability; there is clear evidence that the software is widely used by many users.

SML - Example

Product Name	Documentation - End User (CA1.1)	Documentation - Operational (CA1.2)	Documentation - Development (CA1.3)	Intellectual Property (CA2)	Extensibility (CA3)	Modularity (CA4)	Packaging (CA5)
Component 1	2	3	4	3	4	4	3
Portability (CA6)	Standards Compliance (CA7)	Support (CA8)	Verification and Testing (CA9)	Security (CA10)	Internationalisation and Localization (CA11)	Overall Usability	
4	4	2	2	2	1	3	

3 - Use is possible by most users; with some effort, cost, and risk. A risk assessment should be made before use. (EXPECTED)

Capability Development Model

Structured collection of elements that identify and describe the characteristics of effective preservation processes and activities

- Organisational Infrastructure
- Digital Object Management
- Technical Infrastructure

See [CESSDA Capability Development Model](#) for details



Thanks for listening

Any Questions?

(please contact john.shepherdson@cessda.eu)