CESSDA’s Software Maturity Model

March 2019

John Shepherdson
CESSDA Platform Delivery Director

DOI: 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
- Questions
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
Protection of Assets

CESSDA Bitbucket Code repositories

Ensure CESSDA has access to and IPR for

• source code
• configuration files
• technical documentation

that underpin its tools 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

Separate Development, Staging and Production environments
Technical Framework

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

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

Deploy Docker containers to Kubernetes clusters
- Managed ingress
- Auto scale up/down
- Rolling upgrades of containers
Deployment Pipeline

Automated acceptance testing (up to a point):
- Jenkins pipelines for CI/CD
- Plus other one-off tools

excellent is always level 5
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 coordinated and planned changes to 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

Standards and guidelines

• Adoption of 12 Factor App principles
• User Experience guide
• Software Adoption Policy
• Software Adoption Procedure
• Contributors’ License Agreement
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


• revised in light of ‘Capability Development Model’ from CESSDA SaW project
Common Interoperability Characteristics

How do standards and guidelines help us achieve them?
Loosely coupled but coordinated

Adopt (micro) services architecture based on RESTful web service APIs
• provides a mechanism for reusing and combining software artefacts

Use Docker containers

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 repositories: source-code, containers

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

**API design standards**

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)*
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

Managed Ingress, Reverse Proxy, DNS

See also 12 factor app, number 2 (Dependencies - Explicitly declare and isolate dependencies)
Standards Based

- Provision of common architectural standards (via Technical Architecture)
- Consistent calling and return structures and formats
- Versioned API (API design standards)

Software Maturity Model

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

<table>
<thead>
<tr>
<th>Colour</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minimum standard</td>
</tr>
<tr>
<td></td>
<td>Expected standard</td>
</tr>
<tr>
<td></td>
<td>Excellent standard</td>
</tr>
</tbody>
</table>

See also [Software Maturity Levels](#) document
# Software Maturity Levels

## L1

- Intellectual Property (IP)
- Architecture (ARCH)
- Modularity (MOD)
- Portability (PORT)
- Metadata Compliance (MET)
- Support (SUP)
- Verification and Testing (VER)
- Security (SEC)
- Interoperability and Interchangeability (I & I)

### L1 Description

- **Software design and implementation**: The software design and implementation are not suitable for the intended use. The software is not modular and lacks modularity, making it difficult to understand and maintain.
- **Sustainability**: The software is highly dependent on specific hardware, which limits its portability. The software is not designed to be portable to other platforms.
- **Operational Environment**: The software is tested in a controlled environment and may not function correctly in different operational environments.
- **Security Measures**: The software does not meet security standards and lacks essential security measures.
- **Documentation**: The documentation is incomplete and does not provide adequate information for users to understand and use the software.

## L5

- Intellectual Property (IP)
- Architecture (ARCH)
- Modularity (MOD)
- Portability (PORT)
- Metadata Compliance (MET)
- Support (SUP)
- Verification and Testing (VER)
- Security (SEC)
- Interoperability and Interchangeability (I & I)

### L5 Description

- **Software design and implementation**: The software design and implementation are mature and meet high standards. The software is highly modular and scalable, allowing for easy modification and extension.
- **Sustainability**: The software is designed to be easily ported to different platforms and environments.
- **Operational Environment**: The software is tested in various operational environments and has demonstrated high performance and reliability.
- **Security Measures**: The software meets stringent security standards and incorporates advanced security measures.
- **Documentation**: The documentation is comprehensive and provides detailed information for users to understand and use the software.

---

**Cessda**

[Logo]
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.
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)