Is this metadata management tool any use?

Extending CESSDA's software maturity matrix to the DDI domain

John Shepherdson
Head of CESSDA
Technical Work
Group

License: CC BY 4.0 (for exceptions, see last slide)

Structure

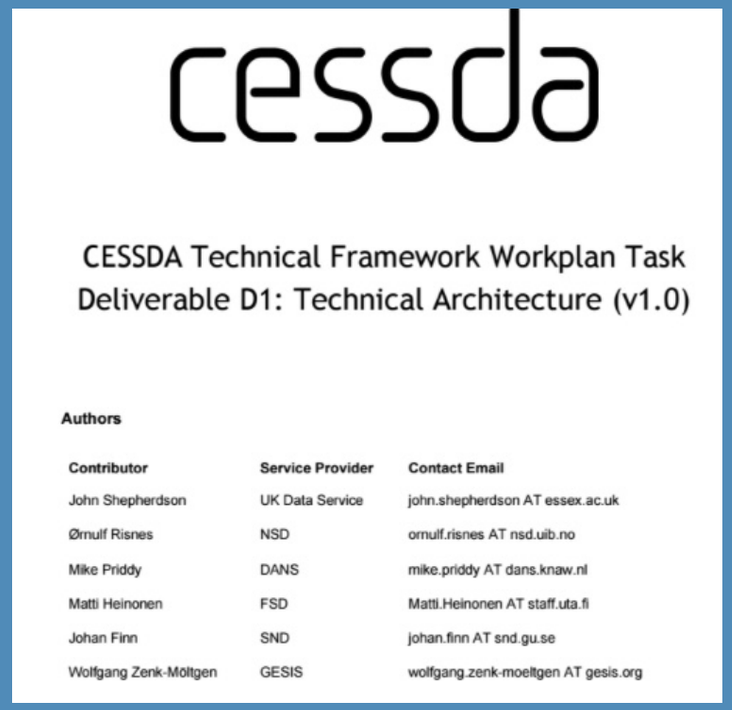
CESSDA

- Overview, vision, objectives
- Common interoperability characteristics

Ensuring software quality

- Technical framework
- Quality control and acceptance criteria
- Software Maturity Levels (SML) scoring

Extending SML to DDI Tools



cessda

CESSDA Technical Framework Workplan Task
Deliverable D1: Technical Architecture (v1.0)

Authors

Contributor	Service Provider	Contact Email
John Shepherdson	UK Data Service	john.shepherdson AT essex.ac.uk
Ørnulf Risnes	NSD	ornulf.risnes AT nsd.uib.no
Mike Priddy	DANS	mike.priddy AT dans.knaw.nl
Matti Heinonen	FSD	Matti.Heinonen AT staff.uta.fi
Johan Finn	SND	johan.finn AT snd.gu.se
Wolfgang Zenk-Möltgen	GESIS	wolfgang.zenk-moeltgen AT gesis.org

CESSDA

- Permanent legal entity owned and financed by the individual member states' ministry of research or a delegated institution
- Each member is represented by a national institution, a Service Provider
- Norway is hosting CESSDA, main office in Bergen
- Recognised as an ESFRI Landmark in the ESFRI 2016 Roadmap in the field of social and cultural innovation

CESSDA's vision

The vision of CESSDA, as stated in its statutes, is to provide a **full scale sustainable research infrastructure** that enables the research community to conduct high-quality research which in turn leads to effective solutions to the major challenges facing society today

CESSDA's objectives

- To support national and international research and cooperation in areas expected to be of great importance in the future
- To facilitate access to social science (and related areas) data resources for researchers regardless of the location of researcher or data
- To provide large scale, integrated and sustainable data services to the social sciences and facilitate and support research, teaching and learning

CESSDA's 5 common interoperability characteristics

1. **Loosely coupled but coordinated** - enable Service Providers to retain independence, yet fully interact in an integrated service

1. **Sustainable** - enable medium and long term investment and business change decisions to be made

CESSDA's 5 common interoperability characteristics

3. **Extensible** - enable additional services to be built on or around it, including adapting to changing functional requirements over time

3. **Maintainable** - enable components to be updated when IT specifications change

3. **Standards based** - enable the coordinated and planned change to all the coupled, but coordinated, services

CESSDA's Technical Framework

A guide for the development of the various (software) products and services that form part of the CESSDA Research Infrastructure

- promote good practice for software development
- protect software assets
- meet common interoperability characteristics

Technical infrastructure for Development, Staging and Production

- harmonise software development tool chain for SPs

Protection of Software Assets

Ensure CESSDA has access to

- source code
- configuration files
- technical documentation

For Research Infrastructure components

Quality Control

Software Maturity Levels

- ensure quality of the research infrastructure is maintained
- guidance on minimum, expected and excellent standards
- originally based on NASA's RRLs
- revised in light of 'Capability Development Model' from CESSDA SaW project

Maturity Modelling - More Info

See EDDI16 presentation:

A Capability Development Model for Assessing and Improving Distributed Infrastructures and their Services

Mike Priddy, Trond Kvamme, Marion Wittenberg

11 Product Acceptance Criteria

- Documentation
  - Development, Operational, End User
- Intellectual property issues
- Extensibility
- Modularity
- Packaging

11 Product Acceptance Criteria

- Portability
- Standards compliance
- Support
- Verification and testing
- Security
- Internationalisation and Localisation

Software Maturity Levels - SML

0 - Not applicable
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

Intellectual Property

1. Software developers have been identified and their responsibilities have been determined.

2. Developer organisation(s) (or developers) have an agreement with CESSDA that addresses any potential conflicts in the proposed intellectual property rights and responsibilities for development.

3. 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.

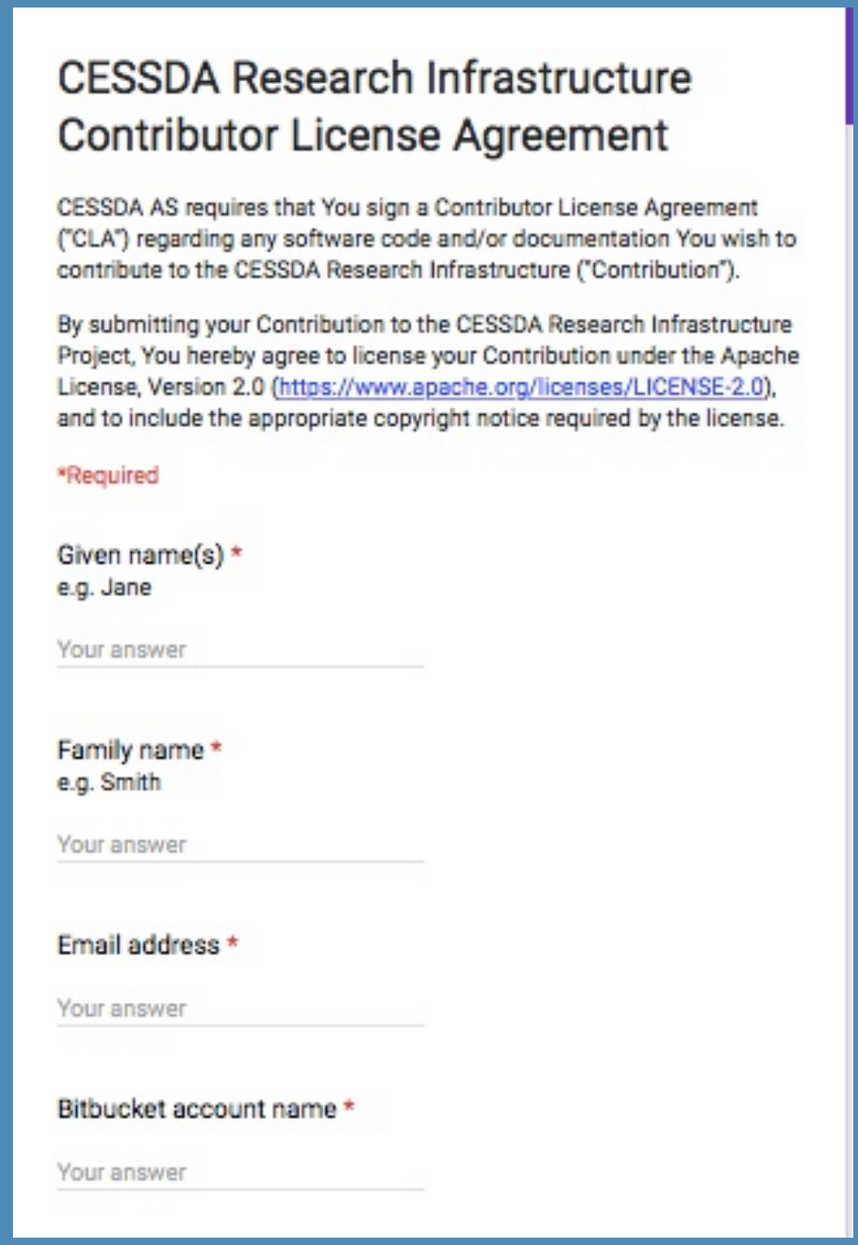
Intellectual Property

4. 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 CESSDA's policies and agreements.
5. 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 CESSDA's policies and agreements.

Intellectual Property

Developer sign up – get write access to CESSDA's code repos

Complete online contributor's agreement - re code ownership and IP



CESSDA Research Infrastructure Contributor License Agreement

CESSDA AS requires that You sign a Contributor License Agreement ("CLA") regarding any software code and/or documentation You wish to contribute to the CESSDA Research Infrastructure ("Contribution").

By submitting your Contribution to the CESSDA Research Infrastructure Project, You hereby agree to license your Contribution under the Apache License, Version 2.0 (https://www.apache.org/licenses/LICENSE-2.0), and to include the appropriate copyright notice required by the license.

*Required

Given name(s) *
e.g. Jane

Your answer

Family name *
e.g. Smith

Your answer

Email address *

Your answer

Bitbucket account name *

Your answer

cessda

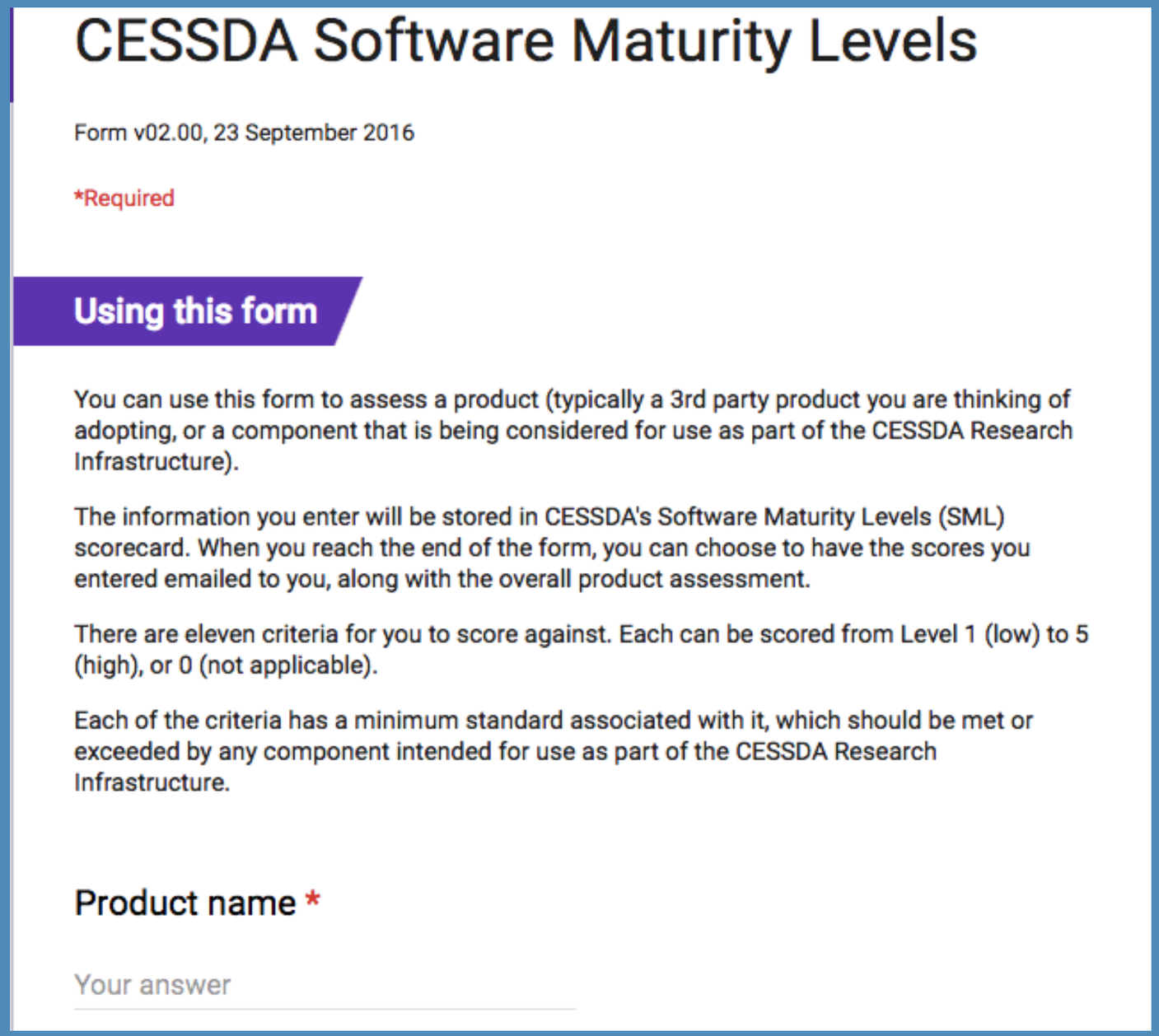
Software Maturity Levels Matrix

Intellectual Property (CA1)	Extensibility (CA2)	Modularity (CA3)	Packaging (CA3)	Portability (CA4)	Standards Compliance (CA5)	Support (CA6)	Verification and Testing (CA7)	Security (CA10)	Internationalisation and Localisation (CA11)
Software developer(s) 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: use rights or limitations have not been specified.	The software was not designed with extensibility in mind, or 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.	There is evidence that the source code was written with no design or consideration for organising the code in terms of functionality for modularity or use. It may have been a demonstrator 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. Even an experienced user may have difficulties installing the software.	The software as a whole is not portable, however, if there is some source code provided with sufficient internal, external and system/hardware documentation is available (here elements may be portable. Executable binaries 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 licensing) or prohibitively expensive.	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 sensibleness of functionality and interfaces.	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.	Software application functionality and unit testing performed.	Security was addressed in the development phases up to and including design.	Internationalization and Localisation not addressed
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 re regulator rights for use.	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.	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 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.	Software includes auto-build feature, but is only available for one operating system. Detailed installation instructions are available and building for other operating systems is possible 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 system. Documentation on porting is insufficient, and should be taken into consideration when assessing the effort required. Cost-benefit and risk analyses, compared to rewriting for the new context and use cases, should be undertaken before porting is considered, and only initiated if the cost benefits of using the software slightly outweigh the cost of developing new software. Porting will nonetheless require significant effort.	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.	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.	Software application demonstrated and tested in a laboratory context. Testing includes testing for error conditions and proof of handling of unforseen input.	Security was addressed in the development phases up to and including implementation. Developers have undertaken appropriate security training.	Software is locale aware.
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 organisation(s) (or developers) may be contacted through a single point to obtain formal statements on restricted rights or to negotiate additional rights.	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.	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 all specified functions; code within each module contains many independent logical paths.	The software is easily configurable for different contexts such as, locations of resources (files, directories, URIs) are configurable. All configuration specific information is centralised.	The software is moderately portable. The software can be ported with only relatively small changes necessary to the context or the software itself. Documentation on porting exists and is complete, but requires considerable effort and expertise. Some rudimentary understanding of the underlying software or the target system may be necessary.	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 named but this may not be for all components. There is documented evidence of standards being used, but not of the verification of components.	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 timestamped releases of updates/patches that are made available and urgent (perhaps due to security issues) updates/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.	Software application demonstrated, tested and validated in a relevant context.	Security was addressed in the development phases up to and including implementation.	Content to localize (text, layout, graphics and multimedia, keyboard shortcuts, fonts, locale data and character sets, build process) has been internationalized.
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 some statements in language that can be understood by laymen, such as a pre-written, recognisable license.	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 its multiple source files, with specific documentation and features to allow the building of extensions which are used across a range of scenarios by multiple user groups. There may be a library available of user-generated content for extensions and user-generated documentation or extension is also available.	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 in programming languages for stronger type checking and compile-time error checking. Services are available externally and code within each module contains few independent logical paths.	A user interface guides the installer through all steps needed to build, configure, and install the software. An uninstaller is also available.	The software is completely portable. The software can be ported to all systems since it runs on an application layer rather than on the underlying operating system layer. Such software is written in languages Java, C#, etc. in theory at least, the software will run on any system in which the appropriate application layer has been installed.	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.	The support by the organisation(s) is clearly defined with frequent and timely updates, releases, etc., responding to the needs of the user community, as well as accessibility of changes by the community. There is a staffed telephone/e-mail 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 insured. Support may be from a low-hazard risk a support Service Level Agreement (SLA) with the developer(s) or a third party.	Actual software application tested and validated through successful use of application and...	Security was addressed in the development phases up to and including product release.	Software has been tested with pseudo-translations.

Colour	Meaning
	Minimum standard
	Expected standard
	Excellent standard

cessda

Online Form



CESSDA Software Maturity Levels

Form v02.00, 23 September 2016

*Required

Using this form

You can use this form to assess a product (typically a 3rd party product you are thinking of adopting, or a component that is being considered for use as part of the CESSDA Research Infrastructure).

The information you enter will be stored in CESSDA's Software Maturity Levels (SML) scorecard. When you reach the end of the form, you can choose to have the scores you entered emailed to you, along with the overall product assessment.

There are eleven criteria for you to score against. Each can be scored from Level 1 (low) to 5 (high), or 0 (not applicable).

Each of the criteria has a minimum standard associated with it, which should be met or exceeded by any component intended for use as part of the CESSDA Research Infrastructure.

Product name *

Your answer

https://goo.gl/forms/uwuye0nTUkti7AiH2

Confirmation Email

Thanks for using the CESSDA Software Maturity Levels (SML) form.

The overall score for Open Source Metadata Harvester (OSMH) is Level 3

Which means "Software is usable; the software can be used by most users although there may be some cost and risk" (i.e. expected standard for CESSDA RI use).

You entered the following values:

CA1.1: End user Documentation	0
CA1.2: Operational documentation	2
CA1.3: Development Documentation	2
CA2: Intellectual Property	3
CA3: Extensibility	4
CA4: Modularity	3
CA5: Packaging	2
CA6: Portability	4
CA7: Standards Compliance	2
CA8: Support	2
CA9: Verification and Testing	3
CA10: Security	3
CA11: Internationalisation and Localisation	3

Not applicable

and provided the following feedback:

"It would sometimes be good to be able to comment on a score, because there are nuances and trade-offs here. Also, an N/A-category could be useful. Used 0 for the CA1.1 since the OSMH don't have "end users" as such."

Regards,

The CESSDA Technical Work Group

Extending SML for DDI Tools

Some suggested criteria:

- Imports/exports multiple versions of DDI
- Multiple representation formats supported (formal syntax)
- Has declared semantics (uses CVs, thesauri, ontologies ...)
- Maintains Provenance
- Supports Curation
- Data cleansing, consistency checking
- etc.

Extending SML for DDI Tools

Straw poll

Is this approach any use?

Please raise an arm if you think so

Extending SML for DDI Tools

Why useful?

Could reduce barriers to acceptance and reuse

- DDI tools
- DDI metadata

More discussion at panel session ‘Re-Use of Software and Administered Metadata’ on Wednesday at 15:30

Conclusion

Software reuse is by design, not by accident.

Adding ‘reusability’ at the end is time consuming and expensive

Thanks for listening

Any question?

cessda

—in your circle of trust.

website: www.cessda.net / *twitter:* @CESSDA_Data

License

This presentation is offered under license CC-BY 4.0

The license does not apply to the following copyrighted material used in this presentation:

- Photograph of the CESSDA Main Office
- CESSDA logotype
- CESSDA colour bar (present at bottom of this and every slide)

cessda