CLARIAH Shared Development Roadmap 2022-2023

Contents

Chapter 1

Introduction

To streamline the development of the CLARIAH infrastructure across work packages, a CLARIAH Shared Development Roadmap (SDR) is created. This SDR specifies what high-level epics CLARIAH will implement and deliver in its the infrastructure. These can subsequently serve as a showcase for the success of CLARIAH.

1.1 Prior work

In phase 1 we identified and described all components and services that are existing or have been worked on in CLARIAH-CORE/PLUS. We called those "CLARIAH+ services" but there was some confusion about the term, we are moving to the term "CLARIAH+ epic" now.

1.2 Goals

In this phase 2 we will put the emphasis on formulating high-level epics and the underlying services, sofware and data components that implement these epics.

The goals for phase 2 are:

- Establish which are the *core shared* CLARIAH+ epics. Emphasis for the shared development roadmap is on these core shared stories, i.e. we are trying to steer the focus for the remainder of CLARIAH+ to development of core infrastructural components.
- Define for each:
  1. What it entails; what are the user stories that make up the epic? Are we all on board with the plan?
  2. Which software/service/data components are needed and what interoperability standards will be adhered to?
  3. How do the components fit together to make a whole? How do the core components relate to other components? By definition, the core shared components should be heavily inter-related. The use of diagrams is strongly encouraged here.
  4. Where can service instance(s) be hosted?
  5. What people need/want to be involved?
  6. Come up with a resource planning that can serve as input for the revised 2022-2023 workplan. For each epic we have a dedicated GitHub projects kanban board and list that should be used for planning and during development (this offers transparency and facilitates cross-institutional collaboration).
  7. Come up with arguments for additional investments from CLARIAH in the services if needed
- A similar procedure applies to the domain-specific services, though the details of these may often be confined to a particular WP. We do need broad agreement which domain-specific services we consider worth the effort and which to drop.

The overall objectives of the Shared Development Roadmap are:

- Identify and foster cross-WP and cross-institutional collaboration opportunities (CLARIAH services spanning components from multiple WPs)
- Fill 'gaps' needed to provide an actual common shared infrastructure.
- Get an accurate overview of how CLARIAH epics relate to everything that has been developed in CLARIAH.

- We seek to harmonize various solutions developed within CLARIAH; determine which are mature and have potential, which can be discarded, which to go forward with in a potential successor project.
- Foster interoperability between software/data components

1.3 Definitions

1.3.1 CLARIAH Epic

A **CLARIAH Epic** is describes the needs of a scholar in broad and generic themes. The epic is made up of multiple user stories, which are mostly described from a scholarly perspective. The epic effectively formulates a 'CLARIAH service' in the largest sense of the word; a product/functionality offered by CLARIAH to scholars.

The implementation(s) of an epic facilitate a certain *scholarly workflow*; a scholarly workflow is defined as a sequence of steps (e.g. analysis, data transformation, presentation) that serves specific needs of a scholar.

We distinguish three types of CLARIAH epics:

1. **Shared core epics** - The epics describe stories that provide core shared functionality for the infrastructure as a whole. These services that implement these serve scholarly use cases and are generic or pervasive (i.e. the needs or requirements that need to be fulfilled are often not directly formulated and recognized as such by scholars). These provide an important pillar for the shared infrastructure. Their development is typically a cross-WP effort.
2. **Domain-specific epics** -- These epics describe stories that cover important functionality for scholars, but are not considered central to the shared infrastructure. The resulting services may be of a more domain-specific nature.
3. **Provisioning epics** -- These are the exception to the rule in the sense that they are not scholarly stories, but are important facilitating stories for the core infrastructure. Their implementations provide low-level pervasive functionality required by most other services (such as authentication, deployment).

A CLARIAH epic is formulated from the perspective of where we want to go with CLARIAH and its common infrastructure, rather than from the perspective of what is the current status-quo.

A CLARIAH epic is realized by one or more components that implement it. These components can be service instances, software components and/or data components and/or interoperability standards that enables a certain workflow.

We aim for "minimal viable" epics and services, this means that we also aim for a minimal viable set of components that makes up a service

Note: Our use of epic corresponds largely with a (highest-level) epic in 'agile' terminology (although some might call this a theme). However, underlying user stories may themselves be considered epics. At this stage we are not concerned with what can be accomplished in single 'sprints'.

1.3.2 Software

Software is any computer program and refers to the codebase as such. We make an explicit distinction between the software and software offered as a service, a **service instance**. Software can be described on any level, it may itself consists of *software components* (i.e. dependencies), *data components* (data required to run the software), and adhere to certain *interoperability standards*. Software has a *provider*, which is the institute/person that maintains and/or develops it.

Software is described in terms of the function(s) that it implements.

1.3.3 Service Instance

This is an instance of **software** when running as a *service* in the CLARIAH infrastructure, offered over the web using a particular *interface*. We use this term to refer to a specific service instance, hosted by a specific *provider*. The provider of an instance need not be the same as the provider of the underlying software. Services consist of underlying software/data components. A software component is mandatory.

1.3.4 Data

Data is any kind of data collection, usually in a specific form and fit for a particular purpose. Data should best be considered immutable, any processing done on it forms a new (derived) dataset. Data can therefore itself consist of *data components* to describe that it is builds upon other data in some way.

Data can also be linked to interoperability standards, such as data formats.

1.3.5 Interoperability Standard

An **Interoperability Standard** is any agreement/protocol/data model/data format designed to facilitate interoperability between multiple software components.

1.4 Evaluation

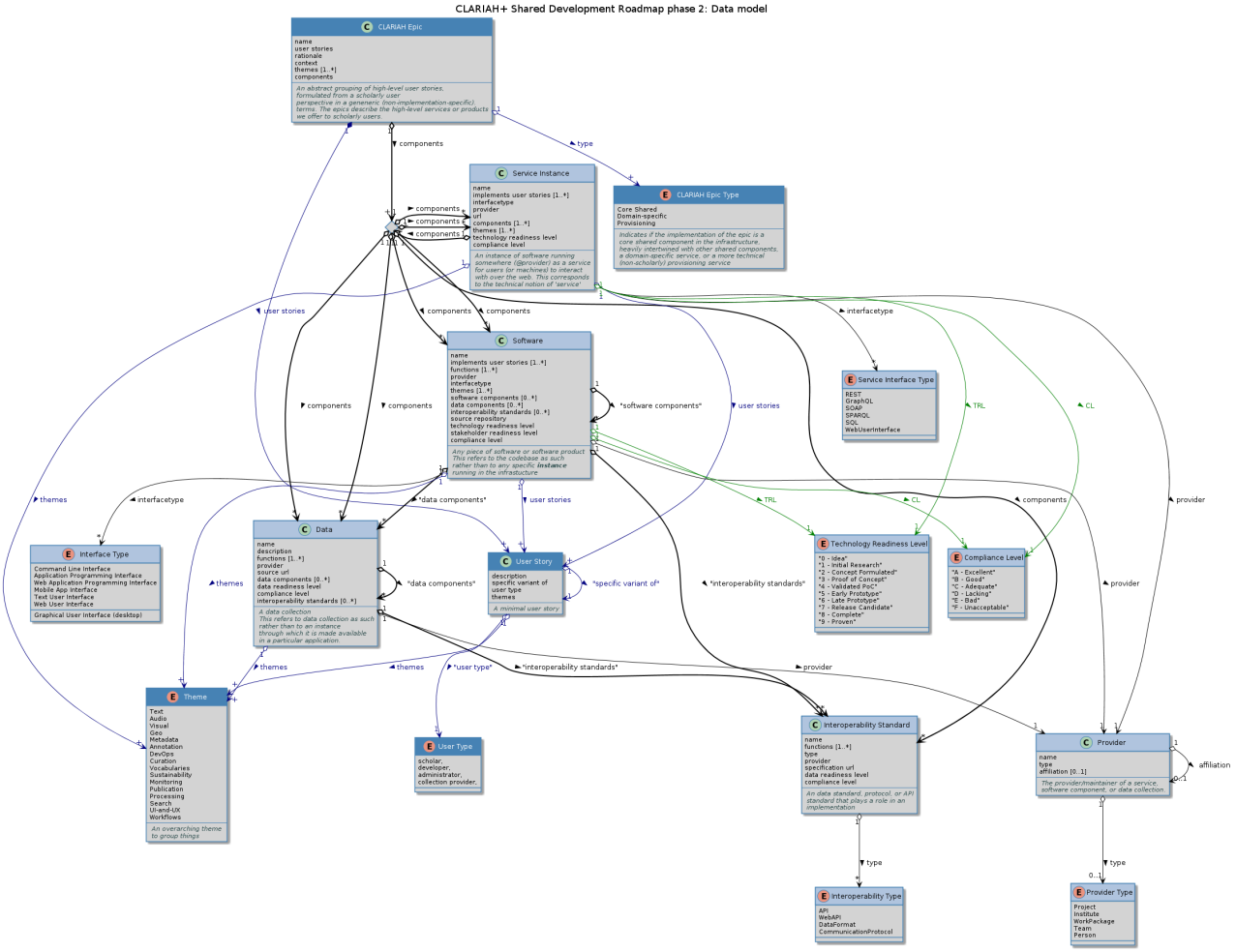
The status of all service/software/data components and the CLARIAH services as a whole are assessed on three axes: a user axis, a technology axis, and compliance axis:

For the user axis we use the Stakeholder Readiness Level (SRL) a measure that defines the user readiness of a new service to be used by scholars. The technology axis we define using the Technology Readiness Level (TRL), a measure that defines the development status of a service.

Compliance with the CLARIAH Software requirements, the CLARIAH Infrastructure requirements, the Documentation Standard, and possible other future requirement specifications. The compliance levels themselves are defined here. It is suggested that SRL level should be estimated by a user panel. Ideally the other levels should also be independently estimated, but in this phase a self-assessment is more realistic.

1.5 Data model

The following schematic (UML) illustrates the data model we use for describing CLARIAH services. Classes with dark blue headers are described primarily from a user perspective, these are connected with blue arrow, classes with light blue headers denote more technical terms. Green arrows are used for evaluation relations.


CLARIAH+ Shared Development Roadmap phase 2: Data model
CLARIAH Epic
name
user stories
rationale
context
themes [1..*]
components
An abstract grouping of high-level user stories, formulated from a scholarly user perspective in a generic (non-implementation-specific) terms. The epics describe the high-level services or products we offer to scholarly users.
CLARIAH Epic Type
Core Shared
Domain-specific
Provisioning
Indicates if the implementation of the epic is a core shared component in the infrastructure, heavily intertwined with other shared components, a domain-specific service, or a more technical (non-scholarly) provisioning service.
Service Instance
name
implements user stories [1..*]
interfacetype
provider
url
components [1..*]
themes [1..*]
technology readiness level
compliance level
An instance of software running somewhere (@provider) as a service for users (or machines) to interact with over the web. This corresponds to the technical notion of 'service'
Software
name
implements user stories [1..*]
functions [1..*]
provider
interfacetype
themes [1..*]
software components [0..*]
data components [0..*]
interoperability standards [0..*]
source repository
technology readiness level
stakeholder readiness level
compliance level
Any piece of software or software product. This refers to the codebase as such rather than to any specific instance running in the infrastucture
Service Interface Type
REST
GraphQL
SOAP
SPARQL
SQL
WebUserInterface
Interface Type
Command Line Interface
Application Programming Interface
Web Application Programming Interface
Mobile App Interface
Text User Interface
Web User Interface
Graphical User Interface (desktop)
Data
name
description
functions [1..*]
provider
source url
data components [0..*]
data readiness level
compliance level
interoperability standards [0..*]
A data collection. This refers to data collection as such rather than to an instance through which it is made available in a particular application.
User Story
description
specific variant of
user type
themes
A minimal user story
Technology Readiness Level
"0 - Idea"
"1 - Initial Research"
"2 - Concept Formulated"
"3 - Proof of Concept"
"4 - Validated PoC"
"5 - Early Prototype"
"6 - Late Prototype"
"7 - Release Candidate"
"8 - Complete"
"9 - Proven"
Compliance Level
"A - Excellent"
"B - Good"
"C - Adequate"
"D - Lacking"
"E - Bad"
"F - Unacceptable"
Theme
Text
Audio
Visual
Geo
Metadata
Annotation
DevOps
Curation
Vocabularies
Sustainability
Monitoring
Publication
Processing
Search
UI-and-UX
Workflows
An overarching theme to group things
User Type
scholar,
developer,
administrator,
collection provider,
Interoperability Standard
name
functions [1..*]
type
provider
specification url
data readiness level
compliance level
An data standard, protocol, or API standard that plays a role in an implementation
Provider
name
type
affiliation [0..1]
The provider/maintainer of a service, software component, or data collection
Interoperability Type
API
WebAPI
DataFormat
CommunicationProtocol
Provider Type
Project
Institute
WorkPackage
Team
Person
components
user stories
interfacetype
type
software components
data components
specific variant of
user type
interoperability standards
provider
themes
TRL
CL
affiliation

Chapter 2

Shared Epics

for the infrastructure as a whole. The services that implement these serve scholarly use cases and are generic or pervasive (i.e. the needs or requirements that need to be fulfilled are often not directly formulated and recognized as such by scholars). These provide an important pillar for the shared infrastructure. Their development is typically a cross-WP effort.

2.1 FAIR Tool Discovery

key	value
id	fair-tool-discovery
coordinator	Maarten van Gompel
wp	1, 2, 3, 4, 5, 6
github-projects-link	https://github.com/orgs/CLARIAH/projects/1
participants	Maarten van Gompel (KNAW HuC)
themes	Metadata, DevOps, Curation, Vocabularies, Sustainability, Processing, Search
evaluation	**trl**: 0, **cl**: 0, **srl**: 0

2.1.1 Rationale

One key goal of the CLARIAH infrastructure is to provide scholars with information where they can find tools that they need for their work. CLARIAH and its predecessor projects have developed a lot of useful tools already. Some of these can be found in repositories such as the CLARIN switchboard. Others are distributed and disseminated on an individual or work package level. However, it would be in the benefit of both scholars and tool providers to have a central place (INEO) where scholars can go to **find** and/or discover tools. At the same time, the tools that they find via the CLARIAH infrastructure should also be **accessible**, so that these tools can indeed be used. From a CLARIAH perspective, we would to some extent also like to guarantee accessibility/usability of tools, and also, that tools are **interoperable** with other tools or CLARIAH infrastructure components. Finally, ideally tools should also be **re-usable**, even if tools change during time (related to sustainability of tools). In practice, it will be hard to warrant full FAIRness of tools provided/disseminated by CLARIAH. We could however at least aim for making tools findable and accessible. For interoperability and re-usability (sustainability) we could aim for a system that informs scholars of the status of tools that are disseminated, e.g., by labeling tools (giving "stars") for it compatibility level, documentation level, and adherence to CLARIAH software requirements. One of the key requirements of a tools discovery service that we propose therefore, is a sound system for aggregating and updating information on tools that reside in various places, the tool metadata.

It is not the aim of the tool discovery service itself to provide means for execution. We assume that (if applicable) the service provides links to individual services (e.g., LaMachine) for executing code using data. However, as being able to execute code is key to "accessibility", making (CLARIAH) tools executable on e.g., web services or local services is part of the development roadmap for this service.

2.1.2 User Stories

1. **As a scholar, I** am looking for tools (please see the definition in the next subsection) and want to browse through and search in a registry of available tools **in order to** select the tools I need to further my research. The registry should offer sufficient information for me to make an informed decision on suitable tools to explore.
2. **As a scholar, I** want to upload my data and automatically be presented with tools that can operate on such data **in order to** more effectively find tools suited for my data. I want to be automatically redirected to the tool I choose, with my data
3. **As a scholar, I** am looking for tools offering a particular interface **in order to** be able to find tools I can communicate with in the fashion I need. For instance, I want tools I can access through the web using a UI; web services with a web API so I can programmatically interact with it from my own scripts; tools I can use locally from the command line; tools that are software libraries which I can use in my own scripts; or even tools that are apps I can run on a smartphone or GUI tools on a desktop.
4. **As an infrastructure provider, I** want all tool metadata to be automatically harvested from the source **in order to** ensure the data is always up to date and facilitate maintenance.
5. **As an infrastructure provider, I** want to be interoperable with the wider CLARIN infrastructure **in order to** have tools available in other CLARIN portals.

2.1.3 Needs & Dependencies

- Compliance to the software/infrastructure requirements as described in the next section
- Cross-WP agreement on some additional vocabulary
- WP4 involvement

2.1.4 Requirements

- Software MUST define CodeMeta software metadata along with the source code
- Services MUST expose a public endpoint providing their specification
- Services SHOULD expose a public endpoint providing high-level CodeMeta metadata
- Services MAY participate in the CLARIN switchboard

2.1.5 Service Description

This core service provides infrastructure for finding tools. The term “tool” here is deliberately ambiguous and can refer to a piece of software in the broadest sense, it may be a web application, web service, programming library, or any composition thereof. Tools may live in a wide variety of places. We seek to standardize the way by which their metadata is described using Codemeta and OpenAPI, which will be posited as software & infrastructure requirements. Codemeta provides basic software metadata, whilst OpenAPI provides metadata covering web service specification. We automatically collect this metadata from as close to the source as possible using a CLARIAH Tool Harvester, the source being either a source code repository or a webservice endpoint. We aggregate all metadata into a central backend solution called the CLARIAH Tool Store. Portals like Ineo, the CLARIN Switchboard or others can either directly query the tool store over an API, or we offer export facilities over an API.

2.1.6 Components

- **Service Component: Ineo**
  - **Function** Web-frontend for scholarly end-users for CLARIAH-wide tool discovery (amongst other things)
  - **URL:** ?
  - **Provider**: ?
  - **Implements:** 1
  - **Interface**: WebUI
  - **TRL**: ?
  - **Software Component: Ineo**
    - **Provider**: KNAW HuC with external party
    - **URL:** ?

- **Service Component: CLARIN Switchboard**
  - **Function** Web-application for end-user for tool discovery focussed on tools that take direct data input from a user upload throughout CLARIN.
  - **URL:** https://switchboard.clarin.eu/
  - **Provider**: CLARIN-ERIC (partner)
  - **Implements:** 1, 2, 5
  - **Interface**: WebUI
  - **TRL**: 8
  - **Software Component: CLARIN Switchboard**
    - **URL:** https://github.com/clarin-eric/switchboard
    - **Provider**: CLARIN-ERIC (partner)
  - **Data Component**: **CLARIN Switchboard Tool Registry**
    - **URL:** https://github.com/clarin-eric/switchboard-tool-registry/
    - **Provider**: CLARIN-ERIC (partner)
    - **Function**: Registry holding all tool metadata descriptions for the switchboard
- **Service Component: CLARIAH Tool Store**
  - **Function** Stores all harvested/aggregated tool metadata
  - **Function:** API for updating (invoked by harvester)
  - **Function:** API for querying (invoked by end-user, portals)
  - **Interface**: REST, possibly SPARQL
  - **TLR:** 0
  - **Software Component: CLARIAH Tool Harvester**
    - **URL**: (does not exist yet)
    - **Function:** Harvester for software & service metadata. Periodically queries all endpoints listed in the CLARIAH Tool Source Registry and updates the tool store.
    - **Provider**: ?
    - **Implements:** 4
    - **Interface**: CLI
    - **TRL**: 0
    - **Comment:** There may be a role for dataverse here to serve as the implementation, but it kind of feels like overkill to me for this purpose.
    - **Comment:** Another option is to use the baserow database we aim to use for components and instances, but here we don't have actual Linked Open Data
  - **Data Component**: **CLARIAH Tool Source Registry**
    - **URL**: (does not exist yet)
    - **Provider**: ?
    - **Function:** Simple registry of software source repositories and service endpoints. Serves as input for the harvester.
    - **Comment:** Could be Implemented as a simple plain text list of URLs in a git repository on github, new registrations can be added using pull requests. Or implemented using the planned baserow database that holds all software components.
    - **TRL**: 0
  - **Software Component: Ineo Export**
    - **URL**: (does not exist yet)
    - **Provider**: ?
    - **Function:** Client using the tool store API (or a direct extension thereof) converting output to a format understood by Ineo, for interoperability with it. Needed if (and only if) we can't make Ineo correct directly to our tool store backend
    - **Implements:** 4
    - **Interface**: CLI or WebAPI
    - **TRL**: 0
  - **Software Component: Switchboard Export**
    - **URL**: (does not exist yet)
    - **Provider**: ?
    - **Function:** Client using the tool store API (or a direct extension thereof) converting output to the format required by the CLARIN Switchboard, for interoperability with it
    - **Implements:** 4, 5
    - **Interface**: CLI or WebAPI

    - **TRL**: 0
  - **Software Component: CMDI Export**
    - **URL**: (does not exist yet)
    - **Provider**: ?
    - **Function:** Client using the tool store API (or a direct extension thereof) converting output to an established CMDI software metadata profile for interoperability with CLARIN
    - **Implements:** 4, 5
    - **Interface**: CLI or WebAPI
    - **TRL**: 0
  - **Interoperability Standard: CodeMeta**
    - **URL**: https://codemeta.github.io
    - **Provider**: The CodeMeta Project (3rd party)
    - **Function**: Software metadata schema (Linked open data, aligning with schema.org)
    - **Function**: Crosswalks for mappings with other existing metadata schemes
    - **TRL**: 8
  - **Interoperability Standard: OpenAPI**
    - **URL**: https://openapi.org
    - **Provider**: OpenAPI Initiative (3rd party)
    - **Function**: Service API specification schema
    - **TRL**: 8
  - **Interoperability Standard: CMDI**
    - **URL**: https://openapi.org
    - **Provider**: CLARIN? (partner)
    - **Function**: Metadata Schema
    - **TRL**: 8
  - **Interoperability Standard: CLARIAH Tool Metadata**
    - **URL**: (does not exist yet)
    - **Provider**: ?
    - **Function**: Extra domain-specific vocabulary need for CLARIAH
    - **Comment**: CodeMeta's vocabulary is too limited to cover all our needs, we need to define extra vocabulary
    - **Comment**: Earlier work in the WP3 'metadata for tools project' can serve as valuable input here
    - **TRL**: 0

2.1.7 Workflow Schema

2.1.8 Evaluation

2.1.9 Context

- Ineo is supposed to become the entry point for CLARIAH tools, however, it can be considered a thin layer and back-end functionality and automatic harvesting needs to be resolved separately.
- A system called CLAPOP was developed in CLARIN and uses manually crafted software metadata descriptions in CMDI (no harvesting) with rich information for scholars. The information may be outdated however.
- A LaMachine Portal (labirinto) was developed as a solution to provide a portal page for any LaMachine installation/deployment, automatically harvesting the tools available within. It uses CodeMeta which is more generic but less specific for scholars.
- The CLARIN Switchboard is developed by CLARIN-ERIC and gives users the option to select tools from a wider CLARIN ecosystem, based on the data they upload. It is largely limited to singular data (single files).

2.1.10 Use cases

2.1.11 Planning

2.1.12 Resources

Chapter 3

Domain-specific Epics

Domain-specific epics describe stories that cover important functionality for scholars, but are not considered central to the shared infrastructure. The resulting services may be of a more domain-specific nature.

Chapter 4

Appendix: Evaluation Metrics

4.1 Technology Readiness Level

The technology axis we define using the "Technology Readiness Level" (TRL), a measure that defines the development status of a service. See the various levels in the table below. The stages that group the levels correspond to the vocabulary already used in the CLARIAH-PLUS workplan. We generally aim for at least TRL7.

- **Planning** stage *(pre-alpha)*:
  - **0 - Idea** - Unproven, untested and largely unformulated concept
  - **1 - Initial Research** - Basic (scholarly) needs observed and reported
  - **2 - Concept Formulated** - Initial technology/application has been concept formulated
- **Proof of Concept** stage *(alpha)*:
  - **3 - Proof of Concept** - Initial Proof-of-concept of key functionality . Concept presented for initial feedback from scholarly users. Not yet validated and not suitable for end-users yet.
  - **4 - Validated Proof of Concept** - Validated Proof-of-concept of key functionality. Technology validated in its own experimental setting (e.g. the lab). Not mature enough for end-users yet.
- **Experimental** stage *(beta)*:
  - **5 - Early Prototype** - Technology validated in target setting (e.g. with potential end-users)
  - **6 - Late Prototype** - Technology demonstrated in target setting, end-users adopt it for testing purposes.
  - **7 - Release Candidate** - Technology ready enough and in initial use by end-users in intended scholarly environments. Further validation in progress.
- **Production** stage (stable):
  - **8 - Complete** - Technology complete and qualified, released for all end-users in scholarly environments.
  - **9 - Proven** - Technology complete and proven in practice by real users.

4.2 Compliance Level

- **A - Excellent** - Technology adheres to as-good-as all posited infrastructure and software requirements.
- **B - Good** - Technology adheres well to the requirements, there only some minor lapses
- **C - Adequate** - Technology adheres to a sufficient amount of requirements, but some major ones are lacking.
- **D - Lacking** - There are too many major requirements that are not met
- **E - Bad** - Many requirements are not met.
- **F - Unacceptable** - Technology violates or is completely dismissive of most requirements. It can not possibly be accepted without drastic changes.

4.3 Stakeholder Readiness Level

We use the "Stakeholder Readiness Level" (SRL), a measure that defines the user readiness of a new service to be used by scholars. This measure can be used for example to prioritize development using criteria such as:

- **Value:** the added value of the service for scholars (1-10) EST=10
- **Support/Commitment:** the enthusiasm in the community to adopt the service (1-10)
- **Cost:** costs for development but also cost involved for using (1-10)
- **Adaptability:** the level of adaptability in existing work processes (1-10)
- **Risks:** an assessment of the risks and their manageability that are involved in using the service (1-10)