CLARIAH
Shared Development Roadmap 2021-2023

To streamline the development of the CLARIAH infrastructure across work packages, a CLARIAH Shared Development Roadmap (SDR) is created. This SDR specifies the “CLARIAH services” within the infrastructure that CLARIAH PLUS will deliver and will serve as a showcase for the success of CLARIAH.

Timetable

- TC meeting of 9 September 2021: discuss products/components/workflows methodology for Tech Day
- 30 September 2021: focussed Tech Day (small group) on CLaaS services
- 28 Oktober 2021: SDR 1.0 with key CLARIAH services
- 28 Oktober 2021: full Tech Day on services and relation to CLaaS infrastructure
- 25 November 2021: full Tech Day on services and relation to CLaaS infrastructure
- 3 December 2021: SDR explained on CLARIAH board heidag
- 14 December 2021: decision on final version SDR + additional resources
- January 2022: Start activities and quarterly monitoring via the board

Table of Contents

1. Introduction

1.1 Definitions

We define **CLARIAH Services** as follows:

- A CLARIAH service is a service offered by CLARIAH to scholars, here ‘service’ is used in the largest sense of the word, as long as it involves software and/or data in some form. In this sense, offering any form of software to users for local usage is also considered a service.
- A CLARIAH service implements 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.
- A CLARIAH service fits a (part of) a scholarly use case and must be directly usable by scholars. We distinguish the following types of services:
    1. *Shared core services* - CLARIAH+ services that provide core functionality for the CLARIAH infrastructure **as a whole**. These services serve scholarly use cases, 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 services* - These CLARIAH+ services provide important functionality for scholars, but are not considered central to the shared infrastructure. Such services may be of a more **domain-specific** nature. Ideally, it fulfills functionality that is pervasive in other use-cases (see above definition). (example: the need for provenance tracking)
    3. *Provisioning Services* - These are the exception to the rule in the sense that they are not scholarly services, but are important **facilitating services** for the core infrastructure. They provide low-level pervasive functionality required by most other services (such as authentication, deployment).
- A CLARIAH service may be described in terms of multiple scholarly user stories, these stories should be as minimal and generic as possible (i.e. not conflated with other user stories). It’s possible that implementations of the service do not cover all user stories as some describe extra/optional features (this should be explicitly indicated).
- A CLARIAH service 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 service is realized by one or more *implementations* (there may be multiple implementations which cover a generic user story). Each implementation consists of a set of *software components and/or data components* and/or interoperability standards that enables a certain workflow.
- The notion of a CLARIAH service does not correspond one-on-one with the stricter technical notion of a service, but is viewed from a scholarly goal-oriented

perspective. The implementation of CLARIAH service may consist of multiple services in the technical sense, which would be software components tied to specific *instances.*

We define **software components** as follows:

- Any software product that is reusable as-is (no unextractable component) and fit for a particular purpose, regardless of interface; so including web services, web applications, programming libraries, command-line tools and even desktop GUIs.
- The software components are described in the following terms:
    - The name of the actual software component. This should correspond to some software implementation and is usually
    - The function(s) of the software component in generic terms.
    - The instance where the software is deployed, in case it is software-as-a-service, and the provider (organization) who maintains that instance.

We define **data components** as follows:

- Any data collection, usually in a specific form and fit for a particular purpose.
- Data components are described in the following terms:
    - The name of the actual data component
    - The function/role the data performs in a specific implementation
    - The software instance that provides the data and/or the provider who maintains it.

We define **interoperability standards** as follows:

- Any agreement/protocol/data model/data format designed to facilitate interoperability between multiple software components.
- Interoperability standards are described in the following terms:
    - The name of the interoperability standard
    - The type of the standard or the role it fulfills, with respect to a specific implementation; e.g. a data format.
    - The authority that maintains the standard

In section 1.4 we will provide a template with clear instructions for all this.

1.2 Objectives

Objectives of this Shared Development Roadmap:

- Formulate (potential) CLARIAH services

- Identify cross-WP collaboration opportunities (CLARIAH services spanning components from multiple WPs)
- Get an accurate overview of how potential CLARIAH services 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
- We aim for “minimal viable” services, this means that we also aim for a minimal viable set of components that makes up a service.

1.3 Evaluation

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

1. 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. See Appendix C
2. The technology axis we define using the “**Technology Readiness Leve**l” (TRL), a measure that defines the development status of a service. See Appendix A
3. Compatibility with the **CLARIAH Software requirements** (appendix D) , the **CLARIAH Infrastructure requirements** (appendix E), the **Documentation Standard** (appendix F), and possible other future requirement specifications. The compliance levels themselves are defined in appendix B

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.

1.4 Instructions

We ask each WP to:

- Consider from the perspective of scholarly use cases what *minimum* viable CLARIAH service the scholar needs and describe these. We aim for generic or pervasive services only that are relevant in a common shared infrastructure.
- For each CLARIAH service, enter a title (heading 4, dark cyan) and fill the template that is outlined below according to the (blue) instructions in there.
- Work out the details for each software/data component in the separate stand-off listing. Multiple CLARIAH services can re-use the same software/data components, hence the stand-off method.
- Estimate a general Technology Readiness level for each component in the separated component list. Also estimate a TRL for the CLARIAH service as a whole (make sure to consult the appropriate appendix). It is also possible that the TRL level for a

software component differs in the context of a CLARIAH service, because a particular feature is needed that is not implemented yet, in that case, add an extra TRL level in the component list with the server.
- Make sure to link the software/data component to its source code repository, website or documentation. The *only* exception where you can omit links is when the Technology Readiness Level (TLR) is so low (<2) that there is nothing to link to yet.
- Describe in terms of services and components, the *cross-WP* infrastructure parts you think are important to develop to showcase the success of CLARIAH (within the scope of the main CLARIAH workplan). We will gradually fit the product/component/workflow descriptions into the CLaaS workflow as depicted here.

Special attention will go to CLARIAH services whose software/data components and/or workflows cross work package boundaries, e.g., pipelines that involve tools from different work packages, or components that serve needs in more than one work package (such as authentication).

Template for CLARIAH Services

[Please adhere to the following template, all instructions are in blue (remove them):]

(name of the person who created this service, WPx)
[^- fill the name and person who filed this service (not necessarily the owner) and the work package where it belongs, remove all blue instructions, like this one, when filling this template]

User story:
As a scholar, I ……. **in order to** ……   *[^- the user story must adhere to the above format and express a need from a user's perspective and the reason/motivation why the scholar might have that need. It is expressed from a scholarly perspective and describes a **minimum viable** service]*   (2) **As a scholar,** I …. **In order to** …..   *[^- in addition to the main user story at the core of this service, additional user stories that provide additional functionality that transcends the minimal viable nature may be specified. Prefix each with a number (and optionally a short identifier) so they can be referenced]*
Software Components & Implementations
Implementation 1: **(name)**   *[^- a user story can be covered by multiple independent/alternative/conflicting pipelines that implement it, give the implementation a number, a name to identify it and describe the components of the implementation in a table as follows (so you can have multiple tables,*

one for each implementation). The components are described by name, function and by the instance and provider that hosts them (in case it is a service), the order has no meaning]

Implements: (numbers of additional user stories)

[^- if multiple user stories are given (and numbered), explicitly indicate which are or are not supported by an implementation. Leave this out if there is only one user story]

Component	Function(s)	Instance @Provider
*[the name of the specific component, i.e. the name of the underlying software that implements the functions describes in the next column. This **must** recur in the stand-off component list in section 3, which provides further details. If the function is planned but there is no implementation yet, add the qualifier "**(proposed)**", the actual name may be empty in that case]*	*[the function/role of a software component in the pipeline, expressed in **generic** terms]*	*[The service instance that provides this functionality/integrates this component, and the provider (institution) that provides it. You **must include a link** to the instance at this provider* *Leave this empty if there is no implementation yet* *It is possible to specify a chain of services using the **via** keyword, the last service should be the most generic one covering the user story and the first service the most specific one for the component]*
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
[Estimate a TRL for the CLARIAH service per implementation (make sure to consult the appropriate appendix). It is also possible that the TRL level for a software component differs in the context of a particular CLARIAH service, because a particular	*[leave SRL empty for now]*	*[The compatibility level expresses to what degree the service in general adheres to the software requirements and infrastructure requirements, leave it empty for now as they're not finished yet]*

feature is needed that is not implemented yet, in that case, add an extra TRL level alongside the component name]		

Verdict
UNDECIDED *[^-- This should only be filled by the CTO (Roeland Ordelman), in discussion with the technical advisory committee, and determines whether this implementation of this service, as described in the template, is either UNDECIDED, ACCEPTED, REJECTED or NEEDS REVISION for further development/maintenance and inclusion in CLARIAH+]*

Data Components

Implementation 1: **(some identifier)** *[<- see before]*

Component (specific)	Function	Instance @Provider
[Name of the actual data component, e.g. a particular data collection or data standard]	*[generic function/role the data fulfills in a specific pipeline; e.g. input corpus, lexicon]*	*[The service instance that provides this data and the provider (institution) that provides it]*

Interoperability Standards

Implementation: (some identifier) *[<- see before]*

Component	Function	Instance @Provider
[Name of the actual standard]	*[Function/type of the standard, e.g. input data format, output data format, protocol, Web-API specification]*	*[The service instance that and its provider (institution) that makes use of this standard. In case of interoperability between two*

		instances, this may contain two linked instances]

Workflow schema

[if an implementation consists of many different components and their flow is not obvious, please draw a schema to illustrate the implementation(s)]

Wider Context

[here you can sketch how this service, either in general or specific implementations thereof, relate to a wider context, such as partner projects like CLARIN, NDE]

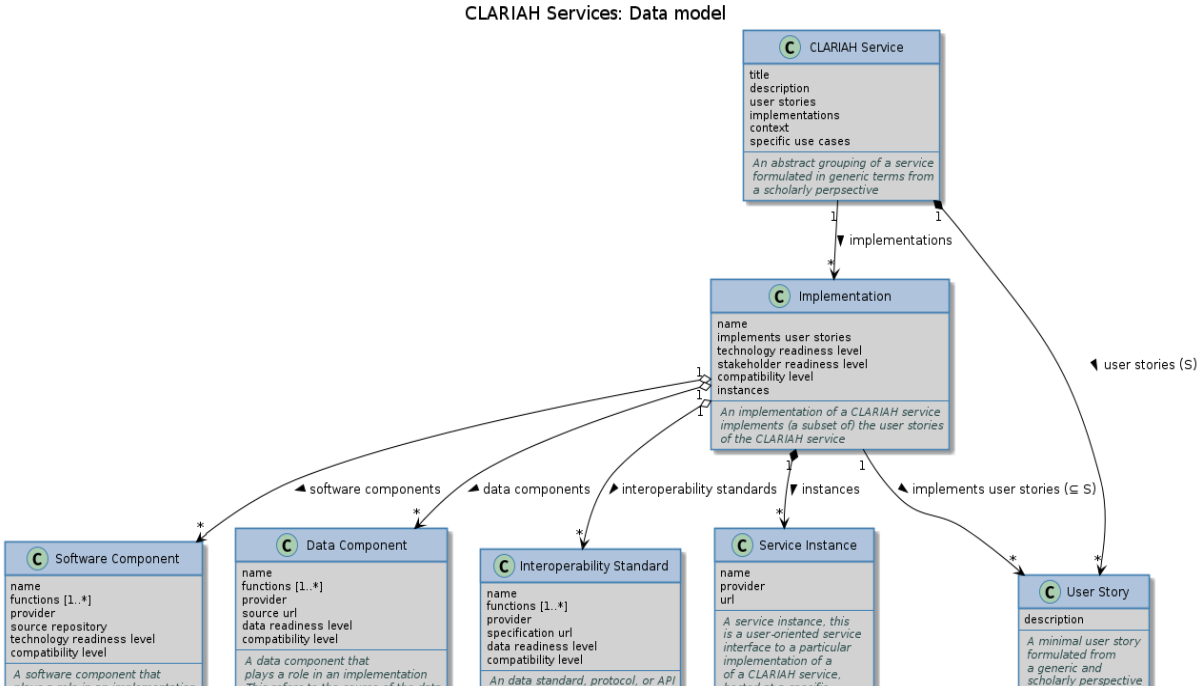
Use cases

- ….

[link to specific use cases for which this user story is relevant, the use cases must reside in https://github.com/CLARIAH/usecases and there should be at least one use case for every service]

Data Model

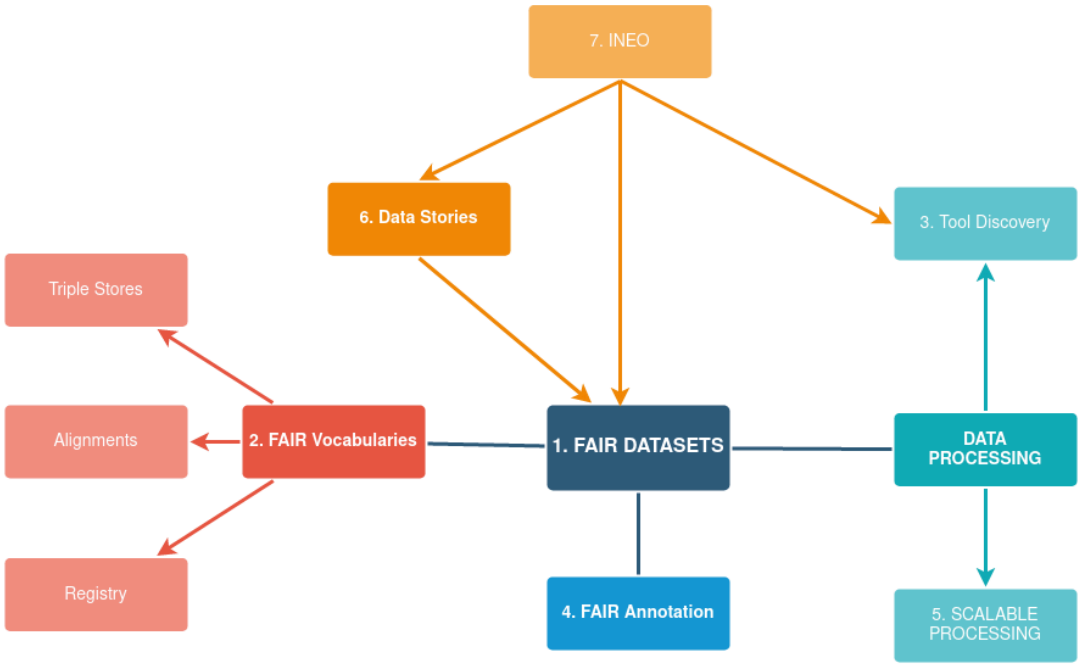
The following schematic (UML) illustrates the data model we use for describing CLARIAH services. It was presented on the CLARIAH tech day in October, [see the video here](#).



2 CLARIAH Services

2.1 Shared Services

Shared services are CLARIAH+ services that provide core functionality for the CLARIAH infrastructure as a whole. These services serve scholarly use cases, 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. For CLARIAH+ we identified the following Shared Services:

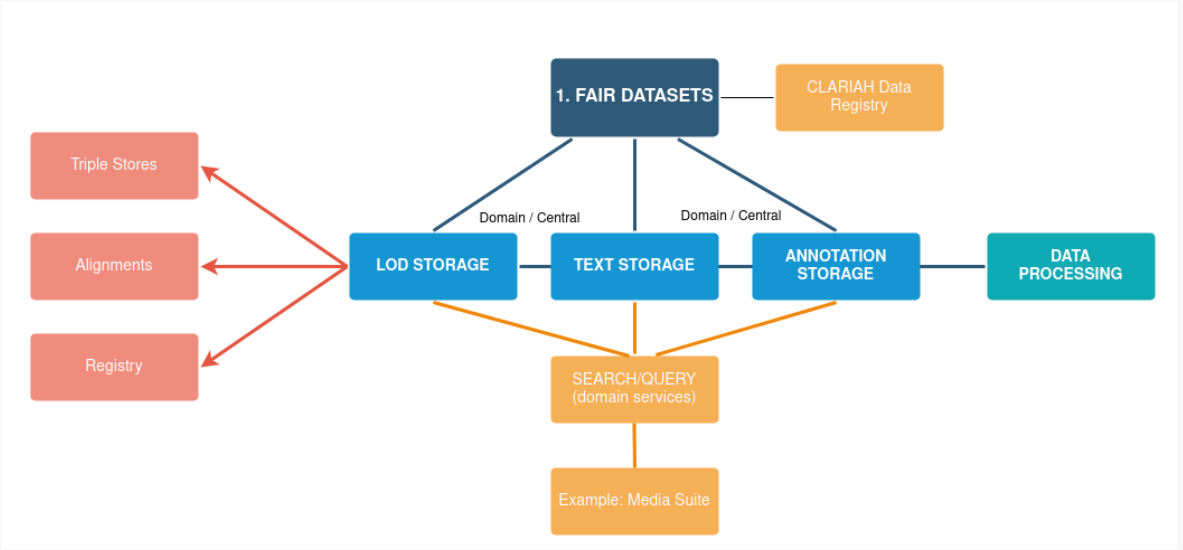


Meta view on CLARIAH Shared Services with FAIR datasets as central service (incorporating structured data organisation, annotation and data processing). INEO is central access point for data and tools, data stories a central facility for working with data.

2.1.1 FAIR Datasets (Dataset Registry)

Coordinator:
Work packages involved: WP5, WP3, WP4, WP6, WP2
Github projects link: https://github.com/orgs/CLARIAH/projects/2/
Rationale

Providing a view on datasets that could be of interest for scholarly research on a national level at research institutes or heritage institutes, is a core functionality of a research infrastructure. At least, potential should be "findable" and provide scholars with information on the contents of the data. In addition to making datasets findable, within CLARIAH we want to stimulate access to these collections and increase the levels of interoperability for reuse. The aim is to create a CLARIAH "dataset registry" (e.g., operated via Ineo) that requires entries to be findable, accessible, interoperable (formats, schema's) and reusable (persistent) within the CLARIAH infrastructure. As such, the FAIR Datasets service also includes work on making individual datasets accessible/interoperable/reusable via *domain services*, e.g., end-points/apis (sparql/search-api) or domain portals (Media Suite, Linguistic corpus search, Nederlab), but also a *shared service* such as Data Stories. A simplified overview (file):



Service description:

Enabling FAIR use of datasets within CLARIAH: a central data set discovery (provided by DCAT or schema.org formatted metadata on the dataset level, e.g. using CKAN as dataset registry), searching within data sets via domain services, accessing data sets via endpoints either on domain level or via a central facility, with persistent identifiers. The proposal is to migrate the CKAN registry that was implemented in WP5 (Media Suite) as a *domain registry service* to a *central registry service* and make an inventory of additional central services that are needed to enable FAIR use of the datasets.

User story:

As a scholar, I want to have an overview of datasets providing me with information from collection metadata, including the dataset's distribution (e.g. SPARQL endpoint, full text search API endpoint, RDF or CSV data dump file) where the dataset is distributed, the organization that publishes the dataset and the license under which it is published, **in order to** select interesting data sets of my research and access them either via domain

portals or central services, by downloading the content myself, or by visiting an organisation.

Software Components & Implementations

Implementation 1: **CLARIAH Dataset Registry**
Implements: 1

Component	Function(s)	Instance @Provider
CKAN (Docker extension)	Webservice and user interface for registering data and discovery	Mediasuite.data@NIBG
NDE Dataset Registry	Webservice and user interface for registering data and discovery (NDE alternative) -- (datasets must adhere to Requirements for Datasets)	demonstrator@NDE
Domain services (search)	Enable search within individual data sets. In principle, data sets are registered centrally, accessed locally.	• Medisuite@NIBG • Nederlab Portal @KNAW-HuC • AutoSearch @INT • GreTeL4 @UU • ...
Satosa instance	For authorisation on dataset registration activities	HuC
PID service?	Mint and resolve persistent identifiers to use for registering collections.	@WP2
INEO	Portal that brings end-users to the discovery service	@WP1/2
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
5-6		
Verdict		

Data Components

Implementation 1: **(some identifier)** *[<- see before]*

Component (specific)	Function	Instance @Provider
Varous registered collections from CLARIAH partners		
Metadata curation (WP3)		MPI, DANS, RU
[Name of the actual data component, e.g. a particular data collection or data standard]	*[generic function/role the data fulfills in a specific pipeline; e.g. input corpus, lexicon]*	*[The service instance that provides this data and the provider (institution) that provides it]*

Interoperability Standards

Implementation: (some identifier) *[<- see before]*

Component	Function	Instance @Provider
CMDI	Metadata standard	*
DCAT		
NDE Requirements for datasets and Dataset Register.		
schema.org		

Workflow schema

[if an implementation consists of many different components and their flow is not obvious, please draw a schema to illustrate the implementation(s)]

Wider Context

[here you can sketch how this service, either in general or specific implementations thereof, relate to a wider context, such as partner projects like CLARIN, NDE]
Use cases
• ….   *[link to specific use cases for which this user story is relevant, the use cases must reside in https://github.com/CLARIAH/usecases and there should be at least one use case for every service]*

2.1.2 FAIR Vocabularies

Coordinator:
Work packages involved: WP5, WP3, WP4, WP6, WP2
Github projects link: https://github.com/orgs/CLARIAH/projects/3/views/1
Rationale
The FAIR Vocabularies service concerns FAIR use of vocabularies, the alignment of entities and schemas across vocabularies, and the validation of alignments.

User story:
As a scholar, **I** want to have a place to store, safeguard or refer to my vocabularies, share them with others and tell how well they fit my use case, search through them or get them recommended based on my data needs, create crosslinks between them and access them through a common API **in order to** share the vocabularies I use in a FAIR way.
Implementations & Software Components

Implementation:

Component	Function(s)	Instance @Provider
Vocabulary registry (WP2)	Registry to describe, classify and rate vocabularies	KNAW HuC
Relation Registry (WP3)	Registry to store relation sets, e.g., linksets or alternative overlays, between vocabularies	
Vocab Recommender (WP4)	Recommend LD vocabularies to model structured data by inspecting the data	

Semantic Gateway (WP2)	Common access to vocabularies from Dataverse	DANS
CLARIAH CMDI Forms (WP3)	Edit CMDI records incl. Interaction with vocabularies	KNAW HuC
LDProxy (WP4)	Make archived vocabularies accessible via their original URL	
SKOSMOS (3rd party)	Vocabulary repository	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Data Components

Implementation:

Component	Function	Instance @Provider
CCR (WP3)	CLARIN Concept Registry	KNAW HuC
CLAVAS (WP3)	CLARIN Vocabulary Registry	
YALC (WP4)	Collection of (pointers to) LD vocabularies	Github
Awesome Humanities (WP4)	Collection of (pointers to) LD vocabularies	Github
FoLiA Set Definitions (SKOS) (WP3)	Vocabularies for text/linguistic annotation	Github
Entity Registries (WP4)		

Interoperability Standards

Implementation:

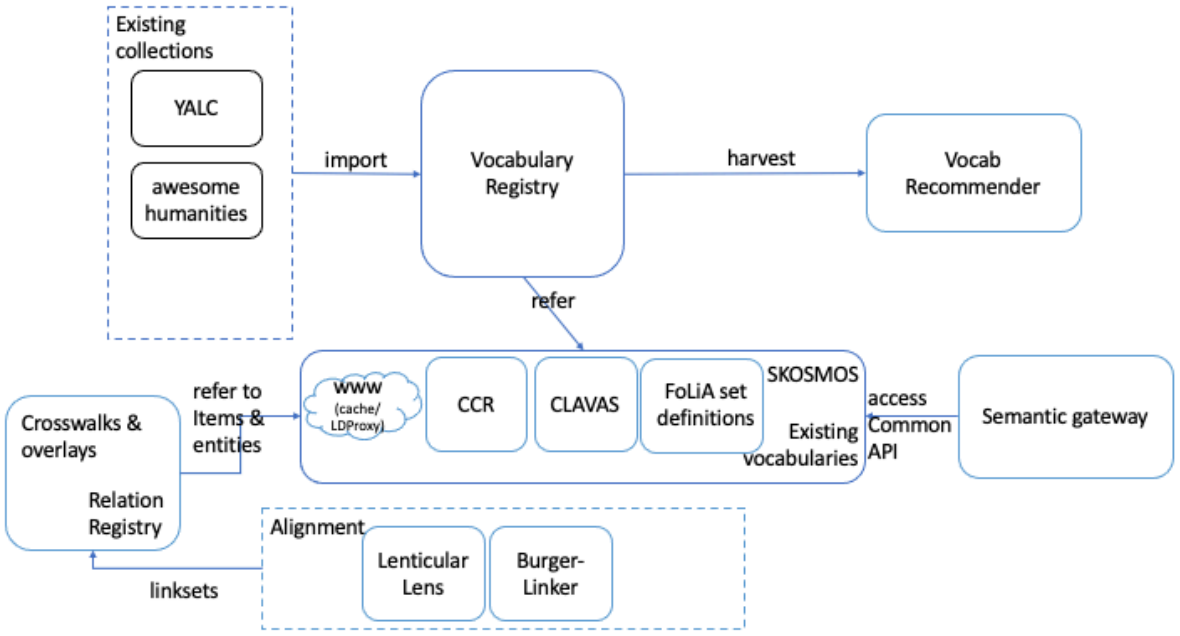
Component	Function	Instance @Provider
RDF		
SKOS		
OWL		
SHACL		

Wider Context

NDE's Termennetwerk, SSHOC Vocabularies Commons, Bartoc

Use cases

-



2.1.3 FAIR Tool Discovery

Coordinator: Maarten van Gompel

Work packages involved: WP3, WP4, WP5, WP6, WP2

Github projects link: *https://github.com/orgs/CLARIAH/projects/1/views/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.

User story:

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; data) **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; interface) **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; harvest) **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.

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.

Implementations & Software Components

Implementation 1: **CLARIAH Tool Discovery (proposed)**
Implements: 1, 2, 3, 4
Note: This implementation focuses on the backend and leaves room to be used with multiple frontends (e.g. Ineo, CLARIN Switchboard).

Component	Function(s)	Instance
Ineo?	Simple web front-end	Ineo @?
CLARIN Switchboard	* Browsable tool inventory * Basic search through tools * Tool recommendation based on analysis of uploaded data (web UI) (TRL 8)	CLARIN Switchboard @CLARIN-ERIC
CLARIAH Tool Harvester	Harvester for software & service	

[proposed]	metadata. Periodically queries all endpoints listed in the CLARIAH Tool Source Registry and updates the tool store.	
Codemetapy (KNAW HuC)	Conversion from various metadata schemes to codemeta	
CLARIAH Tool Store [proposed]	• Holds all tool metadata (as Linked Open Data) • API for updating (invoked by harvester) • API for querying (invoked by end-user, portals) *(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.* *Another option is to use the baserow database we use for components and instances, but here we don't have actual Linked Open Data)*	CLARIAH Tool Store @?
CMDI export [proposed]	Client using the tool store API and converting output to CMDI *(or added directly as an extension to the API)*	
Ineo export [proposed]	Client using the tool store API and exporting to a data format Ineo can ingest. *(or added directly as an extension to the API)*	
CLARIN Switchboard export [proposed]	Client using the tool store API and exporting to the CLARIN Switchboard Registry *(or added directly as an extension to the API)*	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
1		
Verdict		

UNDECIDED

Implementation 2: **CLARIN Switchboard**
Implements: 1, 2, 3

Component	Function(s)	Instance
CLARIN Switchboard	* Browsable tool inventory * Basic search through tools * Tool recommendation based on analysis of uploaded data (web UI)	CLARIN Switchboard @CLARIN-ERIC
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
8		
Verdict		
UNDECIDED		

Implementation 3: **LaMachine Portal**
Implements:1,3,4

Component	Function(s)	Instance
Labirinto	* Portal to all tools installed in a specific LaMachine instance. * Simple faceted search (Web UI)	*any LaMachine installation/deployment* Language & Speech portal @RUN
LaMachine	* Distribution & deployment solution * Introspection pipeline that automatically harvests and collects metadata for all installed tools * Holds the dynamically harvested tool registry	
codemetapy	* Tool to convert software metadata to codemeta * Aggregate metadata from different sources	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
8		

Verdict
UNDECIDED

Implementation 4: **CLAPOP**
Implements: 1,3

Component (specific)	Function	Instance
CLAPOP	• Browsable tool inventory • Faceted search	CLAPOP @UU
Technology Readiness Level (TRL)		
8?		
Verdict		
UNDECIDED		

Implementation 5: **CLARIN Resource Families**
Implements: 1

Component (specific)	Function	Instance
CLARIN Resource Families	• Limited but CLARIN-wide list for a few tool families (NLP focussed), manually curated	Resource Families @CLARIN-ERIC
Technology Readiness Level (TRL)		
8?		
Verdict		
UNDECIDED		

Data Components

Implementation 1: **CLARIAH Tool Discovery (proposed)**

Component (specific)	Function	Instance
CLARIAH Tool Source Registry	Simple registry of software source repositories and service endpoints.	

	Serves as input for the harvester. 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.	

Implementation 2 (+1): **CLARIN Switchboard**

Component (specific)	Function	Instance
CLARIN SwitchBoard Registry	Registry of all available and participating tools	CLARIN Switchboard @CLARIN-ERIC

Implementation 4: **CLAPOP**

Component (specific)	Function	Instance
Collection of software metadata in CMDI	Software metadata descriptions	CLAPOP @UU

Interoperability Standards

Implementation 1: **CLARIAH Tool Discovery (proposed)**

Component (specific)	Function	Instance
CodeMeta	Software metadata schema	*
OpenAPI Specification	Webservice specification schema	
CLARIAH Tool Metadata [proposed]	Additional tool metadata vocabulary agreed upon by CLARIAH WPs *(earlier work in the WP3 'metadata for tools project' can serve as valuable input here)*	

Implementation 3: **LaMachine Portal**

Component (specific)	Function	Instance
CodeMeta	Software metadata schema	*

Implementation 4: **CLAPOP**

Component (specific)	Function	Instance
CMDI	Software metadata schema	*

Wider 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.
- 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.
- The LaMachine Portal 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).

Use cases

2.1.4 FAIR Annotations (Web Annotation clients)

Coordinator: Hennie Brugman

Work packages involved: WP6, WP3, WP5, WP2

Github projects link: *https://github.com/orgs/CLARIAH/projects/4/views/1*

Rationale

The ability to create, store and reuse annotations is a key requirement for scholars and in the context of dispersed data sets and annotations that cross domain boundaries and media types, requires a centralised organisation that is flexible enough to cope with decentralized workflows. The “vision” represented in this service, is that scholars have their own “portable, digital rolodex” in which they can keep and share their personal annotations, generated with a variety of annotations clients, either on private collections or on “institutional” data sets (at DANS or CH institutes) without the need to be bothered with issues such as storage, persistency, provenance, and interoperability. Shared services that need to be provided include annotation storage, PID service, authentication/authorization

(2.3.1), dataset registry (2.1.1), etc. The service should be interoperable with data facilities and existing annotation tools.

User story:

(1) **As a scholar, I** want to annotate collection objects and object parts **in order to** execute one or more of a wide range of scholarly annotation scenarios. Inventories of such scenarios are already made available in CLARIAH.
(2) **As a scholar**, I want to support my research by autonomously collecting and annotating parts of a wide range of collections, from large online CH collections to non-digitized personal collections. I want to be able to store such references and annotations, and I want to be able to share them with my peers.
(3) **As a tool builder, I** want to see working examples of interactive annotation clients, scripts and Jupyter notebooks **in order to** learn how to create my own clients or extend these examples for my own tasks.

Software Components & Implementations

Implementation 1: **Annotation Rolodex**

As part of their research, scholars regularly collect references to relevant data from multiple collections, including collections that are not even digitized (the 'shoebox in the attic'). This service allows researchers to register such previously anonymous collections persistently, so that they can be annotated. Furthermore, it supports scholars to fill their 'research rolodex' with relevant references to a wide range of collections that are already online available by means of creation, management and sharing of web annotations.

Implements: 1, 2

Component	Function(s)	Instance @Provider
Rolodex web application	Autonomous web application that supports collection registration, annotation and annotation management.	@HuC
Collection Registration Service	CLARIAH registration for existing and new collections. Minimally provides a persistent identifier for a collection and a minimal metadata description.	@WP2?
Annotation repository	Store, share, manage, search	@HuC

with groupware functions	web annotations.	
PID service	Mint and resolve persistent identifiers to use from web annotations and for registering collections.	@WP2
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Implementation 2: **Showcase for micro-clients**

We already have a number of well described scholarly annotation scenarios available. Given that building large interactive apps is very expensive and that the lifespan of such apps is relatively short we aim for a set of ‘micro-clients’: small interactive apps and (jupyter) notebooks that can be build rapidly and by anybody, can be specialised and are so cheap that they can be deprecated when not useful anymore. We want to make a number of such micro-clients available for use by scholars, but also as examples for other tool builders.

Implements: 1, 3

Component	Function(s)	Instance @Provider
µAnnotator	Small interactive client that loads and shows a slice of text plus annotations and enables adding personal annotations	@HuC
Scripted annotation scenarios	diverse	@HuC
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		

UNDECIDED

Implementation 3: **Web Annotation extensions for existing CLARIAH clients**

This section lists a number of existing tools that support manual annotation: FLAT, COBALT, ELAN, MediaSuite. These tools might benefit from connecting them to one or more shared CLARIAH Web Annotation Repositories. These tools may need extensions for that.

Implements: 1

Component	Function(s)	Instance @Provider
Annotation repo support for FLAT, COBALT, ELAN, MediaSuite	Share and exchange web annotations between different CLARIAH annotation tools	@HuC, @INT, @MPI, @NISV
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Implementation 4: **extensions for existing external annotation tools**

Some external annotation tools are regularly used by several CLARIAH partners: Inception, BRAT, Recogito(-js), possibly others. To make these tools interoperable with the CLARIAH Annotation infrastructure we propose to make use of Web Annotations that can be stored and exchanged via a CLARIAH Annotation Repository. To do this, format converters or tool extensions are necessary.

Implements: 1

Component	Function(s)	Instance @Provider
Extension for recogito.js	i/o for web annotation format and storage	@HuC

Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Implementation 5: **TextFabric front-end**

Create, load and visualise annotation sets.

Implements: (numbers of additional user stories)

Component	Function(s)	Instance @Provider
TextFabric		@DANS
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Data Components

Implementation 1:

Component (specific)	Function	Instance @Provider

Interoperability Standards

Implementation: all

Component	Function	Instance @Provider
W3C Web Annotation data model, protocol and format	Protocol, API, Format	

Workflow schema

Wider Context

To enable and stimulate manual annotation scenarios we not only have to provide Annotation Repository services, but also have to implement and/or demonstrate a range of viable annotation scenarios with client software. These clients can be interactive or scripted using API's, they can be existing annotation tools or newly developed micro-clients, and they can be provided by CLARIAH or be external tools.

Use cases

- Micro-frontends for manual scholarly annotations

2.1.5 Scalable Multimedia Processing

Coordinator:

Work packages involved: WP3, WP5, WP6, WP2

Github projects link: *https://github.com/orgs/CLARIAH/projects/5/views/1*

Rationale

In many scholarly use cases in CLARIAH, multimedia processing tools play a crucial role. One of the main category of these tools are NLP tools, that in its broadest sense also includes tools such as OCR/HTR, speech recognition or even sign language recognition. But also tools such as computer vision can be seen as part of the multimedia processing suite that CLARIAH scholars would like to use. The development of the individual tools themselves often occurs outside the CLARIAH scope. In practice, the main concern of

CLARIAH is to enable the use of such tools in the context of scholarly use cases. The Fair Tool Registry service provides a mechanism to point scholars to individual tools that can be accessed locally (e.g., via LaMachine) or via a webservice (e.g., speech recognition service at RUN). This “individual execution” of (CLARIAH) tools is also part of the work in FAIR Tool Discovery. However, there are scholarly use cases that require the *scalable* execution of tools within the infrastructure, either as a bulk processing job that requires a High Performance Cluster (e.g., processing a large data set), as part of a complex workflow for a specific task, or as part of a “tools-to-data” use case (e.g., for Jupyter Notebook analysis). The service “Scalable Multimedia processing” is about enabling such use cases.

Description:

This service provides the ability to run processing tools on text/multimedia in a scalable fashion. It provides means of execution over multiple computing nodes in a computing cluster and execution on a large data collection.

User story:

(1a) **As a scholar,** I want to apply a processing tool on a large data set in the CLARIAH infrastructure, either using computational resources provided by the CLARIAH infrastructure **in order to** be able to do quick and efficient processing on large data sets without needing my own infrastructure.

(1b) **As a scholar,** I want to apply a processing tool on a large data set within my own infrastructure **in order to** take the tools to my (restricted) data and work in my own secure environment.

(2) **As a scholar**, I want to have access to information about the tool (provenance) **in order to** facilitate reproducibility.

(3) **As a scholar**, I want to run tools distributed over multiple machines **in order to** be able to efficiently process big data collections.

Implementations & Software Components

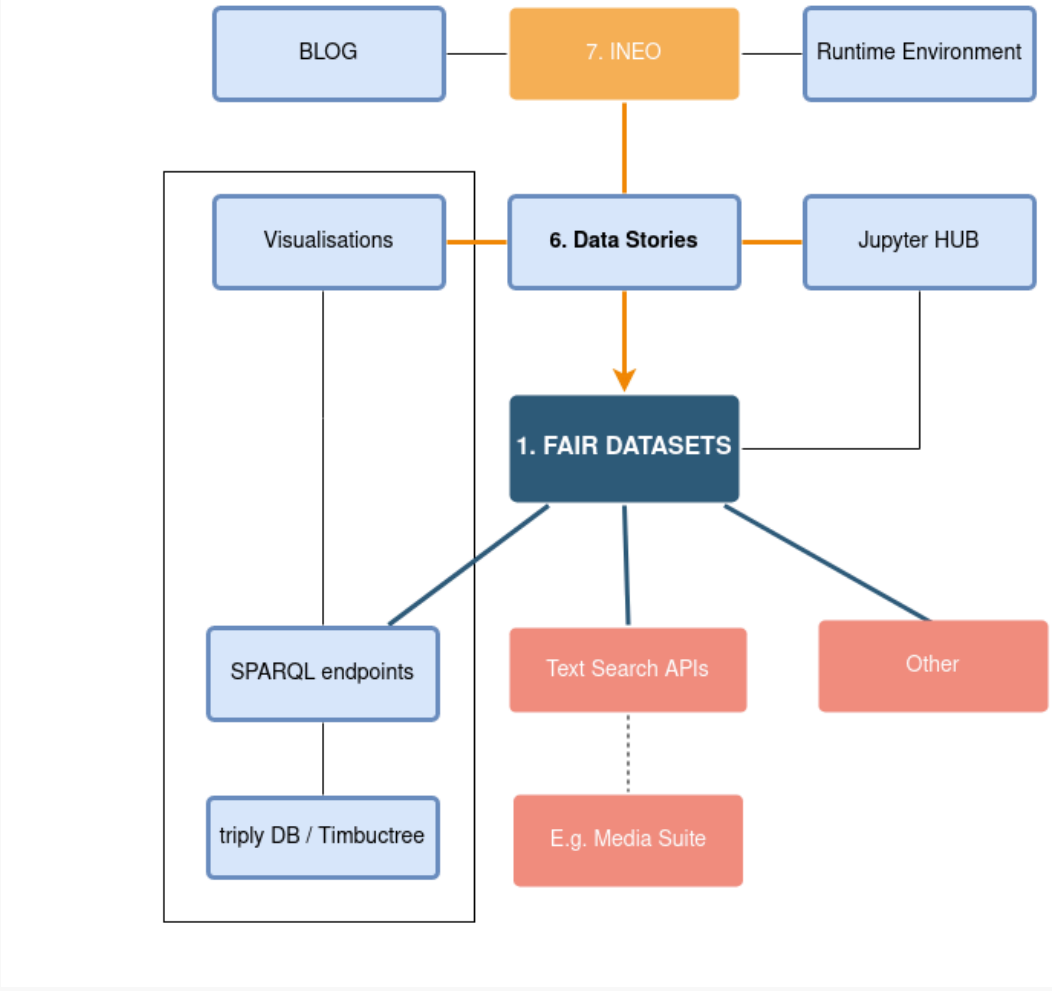
Implementation: Proposed implementation
Implements: 1a,1b,2,3
(not done yet)

Component	Function(s)	Instance @Provider
DANE-server (B&G)	Workflow manager (API, task scheduler and UI)	

Nextflow (3rd party)	Workflow manager (API, task scheduler, workers)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
2		
Verdict		
UNDECIDED		

2.1.6 Data Stories

Coordinator: Menzo Windhouwer
Work packages involved: WP4, WP2
Github projects link: *https://github.com/orgs/CLARIAH/projects/6/views/1*
Rationale
The Data Stories service facilitates the creation of stories (scientific output) based on the analysis and often visualisation of events (data points) in data sets. Scholars that create data stories, typically use low-level access methods such as APIs via Jupyter notebooks to do their analysis, sometimes referred to as "programming with data". The Data Stories can be published as notebooks that an end-user can run (requiring a run-time environment) or in written form (e.g. blog) that summarizes results from a notebook (and provides the notebook as a reference). The Data Stories service obviously has a strong relation with the FAIR datasets service, in particular the shared linked open data "stack". Simplified overview of the Data Stories (file):



User story:
As a scholar, I want to tell a story illustrated by live queries on data sources and attractive visualisations of the answers **in order to** present the results/value of my data work to the wider public.
Implementations & Software Components

Implementation:

Component	Function(s)	Instance @Provider
Data Stories(WP2)	Create data stories on top of a datastore	
Data Stories	Exchange format to make data stories	

interoperability (WP2)	interoperable	
Data Stories Visualisations (WP2)	Various visualisations of query results, e.g. maps, timelines, barcharts, pies	
Data Stories in Ineo (WP1)	Allow the embedding of a data story in Ineo as a dataset publication mechanism	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
2		
Verdict		
UNDECIDED		

Data Components

Implementation:

Component	Function	Instance @Provider
Notebooks (WP5)	Data stories already available in WP5	
Notebooks in LaMachine (WP3)	Data stories already available in WP3	
Data Stories in Druid (WP4)	Data stories already available in WP4	

Interoperability Standards

Implementation:

Component	Function	Instance @Provider
To be developed	Exchange format to make data stories interoperable	

Wider Context

Use cases

-

2.1.7 Ineo

Coordinator: Sebastiaan Fluitsma, Thomas Vermaut, Roeland Ordelman
Work packages involved: WP1, WP2
Github projects link: *https://github.com/orgs/CLARIAH/*
Rationale

User story:
As a scholar, I want to have a central place where I can go to, to find out about the possibilities of the CLARIAH infrastructure for my research **in order to** point me to data, tools, examples, best-practices and standards.
Implementations & Software Components

Implementation:

Service	Function(s)	Instance @Provider
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		

UNDECIDED

Data Components

Implementation:

Component	Function	Instance @Provider

Interoperability Standards

Implementation:

Component	Function	Instance @Provider

Wider Context

Use cases

-

2.2 Domain Services

Domain services are CLARIAH+ services that provide important functionality for scholars, but are not considered central to the shared infrastructure. Such services may be of a more domain-specific nature. Ideally, it fulfills functionality that is pervasive in other use-cases (see above definition).

2.2.1 OCR & HTR service

Coordinator:

Work packages involved: WP4, WP2

Github projects link: *https://github.com/orgs/CLARIAH/*
Rationale

User story:
As a scholar, I have scans (e.g. PDFs or images) which I want to upload and digitise using optical character recognition or handwritten text recognition (OCR/HTR) **in order to** make the data available for further research, whatever that may be, such as further enrichment, annotation, search, etc. (2) **As a scholar**, I want to process some images of written documents **in order to** extract data such as text and layout information for further actions later, such as NER, document classification, searching and textual analysis.
Implementations & Software Components

Implementation 1: **PICCL** (limited to OCR rather than HTR)

Component	Function(s)	Instance @Provider
Tesseract	OCR engine	PICCL @RUN or PICCL @INT *(outdated)*
PICCL/ocr.nf	Workflow around the OCR engine	
Nextflow	Workflow engine	
• FoLiAutils • foliatools	Data conversion	
PICCL webservice (powered by CLAM)	• Web API (REST) • Upload, processing web app	
FLAT	Front-end for visualisation of results (optional, document-based)	FLAT @RUN via PICCL @RUN
LaMachine	Distribution & Deployment solution (optional)	*
clamopener	Basic authentication provider (non-federated, to be replaced)	PICCL @RUN only
Shibboleth	Federated auth solution	via PICCL @INT only
Technology Readiness Level	Stakeholder Readiness Level (SRL)	Compatibility Level

(TRL)		
7		
Verdict		
[RO]: • **Not on SDR because TRL7** • **make discoverable in INEO (given TRL7 level)** • **possibly some small efforts for fine-tuning** • **Requirement: solve dependency on developer**		

Note: OCR post-correction is listed as a separate service, but both are implemented in PICCL and provided by the same instance

Implementation 2: **OCR workflow builder** (agnostic wrt OCR/HTR distinction)

Component	Function(s)	Instance @Provider
OCR workflow builder	OCR workflow construction tool, generating a workflow to be executed by the OCR-D container	@INT *(not online yet)*
OCR-D	Docker container implementing a variety of recent tools for layout analysis, image processing and text recognition	
Execution bridge	Server side component; validates and sends constructed workflow to instance for processing	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
3		
Verdict		
[RO]: • **Not on SDR because TRL3 and already existing (redundancy argument)** • **Possibly on WP workplan but requires argumentation[RO]:** • • **Not on SDR because TRL3 and already existing (redundancy argument)** • • **Possibly on WP workplan but requires argumentation**		

Implementation 3: **Pergamon IMages - PIM**
Implements: (2)

Component	Function(s)	Instance @Provider
Transkribus connection/interfacing	HTR	knaw-huc
Tesseract	OCR + training	knaw-huc
Kraken	OCR/HTR+ training	
PyLaia	HTR + training	
P2PaLA	Layout analysis + training	knaw-huc
Document classification (visual)(noname)	Document classification + training	knaw-huc/NA
Epub Generation(noname)	Epub Generation using tooling above	National Library
Baseline detector(noname)	Baseline detector + training	knaw-huc
HTR (noname)	HTR + training	
Layout Analysis (noname)	Structure labeling + training	knaw-huc
Connectivity to INT CLARIAH interface	Expose software above via INT OCR workflow builder	
Image repo + connectivity for harvesting web/IIIF/manual uploading	Storing links to images and their transcriptions. Optionally upload images.	knaw-huc
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
3/5/6/7		
Verdict		
[RO]: • **Not on SDR because of redundancy, involvement of industry** • **Possibly WP workplan but requires argumentation**		

Data Components
Implementation: PICCL

Component	Function	Instance @Provider
Tesseract OCR Models	OCR Models	*

Implementation: OCR workflow builder

Component	Function	Instance @Provider
OCR/HTR and layout analysis models	Models necessary to run the tools	*
Ground truth data	Data necessary to train the tools	

Implementation: PIM

Component (specific)	Function	Instance @Provider
Trained models for Kraken		
Trained models for Tesseract		
Trained models for PyLaia		

Interoperability Standards

Implementation: PICCL

Component	Function	Instance @Provider
FoLiA	Output data format	*
CLAM	RESTful web-API specification	*
Oauth2 & OpenID Connect	Authentication/Authorization Protocol	PICCL @INT (not enabled for PICCL @RUN)

Implementation: OCR workflow builder & PIM

Component	Function	Instance @Provider
Page XML[1]	Output data format	*

Implementation: PIM

[1] https://www.primaresearch.org/schema/PAGE/gts/pagecontent/

Component	Function	Instance @Provider
Google login		

Wider Context

Implementation PICCL: PICCL was developed in CLARIAH-CORE and CLARIAH-PLUS WP2 & WP3, successor of the earlier TICCLops in CLARIN-NL. However, current funding for PICCL has ended and a main developer has retired. Continuation of this implementation depends heavily on Martin Reynaert (UvT)

Platforms like Transkribus outperform the OCR of Tesseract, the downside is that these are proprietary solutions.

Deep-learning based models implemented in recent tools (calamari, kraken, ocropy, ..) will be deployed in the workflows generated by the OCR workflow builder; a systematic comparison of the performance of these tools to Transkribus is not yet available.

Use cases

- Seventeenth century newspaper collection (WP6 use case 2)
- Republic (WP2)
- Globalise (WP2)
- National Library (WP2)
- National Archive (WP2)
- Individual Scholars (WP2)

2.2.2 Speech Acquisition

Coordinator: Menzo Windhouwer

Work packages involved: WP3

Github projects link: *https://github.com/orgs/CLARIAH/*

Rationale

User story:

As a scholar, **I** want to send a questionnaire to the public to ask all kinds of questions including the recording of utterances **in order to** collect a balanced sample of (audio) responses.

Implementations & Software Components

Implementation: **SPAQ**

Component	Function(s)	Instance @Provider
JIT Lime Survey (WP3)	Spin up a dedicated LimeSurvey upon request	KNAW HuC
SPAQ (WP3)	Create and ask questions involving audio recording	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
5		
Verdict		
[RO]: • **Not on SDR as it is too specific for a single user community although link with surveys in ODISSEI context could be interesting but could be interesting for CLARIAH-3** • **I can imagine CLARIAH+ will bring this to TRL-7 on a WP-level** • **With TRL-7 make discoverable in INEO**		

Data Components

Implementation:

Component	Function	Instance @Provider

Interoperability Standards

Implementation:

Component	Function	Instance @Provider

Wider Context

Use cases
•

2.2.3 Geodata entry & management

Coordinator: Thomas Vermaut
Work packages involved: WP2
Github projects link: *https://github.com/orgs/CLARIAH/*
Rationale

User story:
As a scholar, I want to process some geographical data (p.e. a scanned historical map) **in order to** represent them in a single repository for scientific curation and potential further actions later, such as displaying the data on a map. (2) **As a scholar,** I want to process geographical data with the help of the crowd / collaborative editing **in order to** represent them in a single repository for scientific curation and potential further actions later, such as displaying the data on a map.
Software Components & Implementations

Implementation 1: **HisGIS**
Implements: (2)

Component	Function(s)	Instance @Provider
OSMAPI	Provide collaborative editing on geographical data; GUI + DB + API	OSM @ HisGIS
MAPNIK	Map rendering	
overpass	Geographical data-query	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level

8/9		
Verdict		
[RO]: • **Not on SDR as TRL > 7** • **Make discoverable in INEO** • **Work on data components (or fine tuning software for enabling use of data components) may be part of workplan but requires argumentation**		

Data Components

Implementation 1: **HisGIS**

Component (specific)	Function	Instance @Provider
HisGIS	Provide geographic historical cadastral reference data.	OSM @ HisGIS

Implementation 2: **Gemeentegeschiedenis**

Component (specific)	Function	Instance @Provider
Gemeentegeschiedenis (in progress)	Provide geographical reference data for Dutch municipalities and their boundaries.	

Implementation 1: **atlas**

Component (specific)	Function	Instance @Provider
Historical atlas of the low countries (idea)	Provide reference-data for heterogeneous administrative entities through time.	

Interoperability Standards

Implementation: (some identifier)

Component	Function	Instance @Provider
OSM-XML		
OSM-changeset	Specifying specific changes	

	between two versions of a dataset.	
overpass-query-format	A standard way of querying complex multi-tiered geographical datasets.	

Use cases

- HisGIS
- Gemeentegeschiedenis (in process)
- Historical atlas of the low countries (Rombert Stapel; idea)

2.2.4 FAIR Lexicons

Coordinator: Jesse de Does, Henk van den Heuvel

Work packages involved: WP3

Github projects link: *https://github.com/orgs/CLARIAH/*

Rationale

Description:

This is a service that allows look-up of words or phrases in one or more dictionaries.

User story:

As a scholar, I want to look up words in dictionaries **in order to** (1) obtain senses and definitions, (2) find word forms, synonyms/antonyms/hypernyms/hyponyms etc (3) find usages example *(not all need apply)*

(4) **As a scholar**, I am interested in looking up words for Dutch dialects and doing lookups for specific locations/dialects

(5) **As a scholar**, I am interested in browsing entire dictionaries

Implementations & Software Components

Implementation 1: **Dialect Dictionaries**
Implements: 1, 3, 4, 5

Component	Function	Instance @Provider
RU-wnd	Web application for dictionary lookups (no Web API)mathieu_concepts	e-WLD @RUN (Limburgs), e-WBD @RUN (Brabants), e-WGD @RUN (Gelders), e-WALD @RUN, (Achterhoeks)

Implementation 2: **dictionaries and lexica** available at INT (actually, different implementations)

Component	Function	Instance @Provider
Historical dictionaries	Web application (no public API)	Historical dictionary portal (ONW, VMNW, MNW, WNT, WFT)
Dictionary of contemporary Dutch	Web application (no public API)	Algemeen Nederlands Woordenboek
Dialect dictionaries	Web applications (no public API): - elektronische Woordenbank van de Nederlandse dialecten (eWND) - Integrated dictionary portal Database of the Southern Dutch Dialects	eWND @ INT DSDD @ INT
DiaMaNT	Historical semantic lexicon, user interface	DiaMaNT @INT
Webcelex	Select data from the well-known CELEX lexica	https://portal.clarin.inl.nl/webcelex/
Lexicon service	Web service for lexicon lookup in modern and historical morphosyntactic lexica	Available on request (contact INT)

Implementation 3: **dictionaries and lexica** available at Fryske Akademy (actually, different implementations)
Note: developed independently of CLARIAH

Component	Function	Instance @Provider
Frisian lexicon service	Web service for lookup in modern frisian data	Lexicon Service @Fryske Akademy

Frisian dictionary services	Web services and applications for lookup in frisian dictionaries	Dictionary Service @Fryske Akademy
Frisian language API	Graphql service for intelligent lookup of word information, text search and translation	LanguageAPI @Fryske Akademy

Data Components

Implementation 1: **Dialect Dictionaries**
Implements: 1, 4

Component	Function	Instance @Provider
Woordenboek van de Limburgse Dialecten	Lexicon	e-WLD @RUN
Woordenboek van de Brabantse Dialecten	Lexicon	e-WBD @RUN
Woordenboek van de Gelderse Dialecten	Lexicon	e-WGD @RUN
Woordenboek van de Achterhoekse en Liemerse Dialecten	Lexicon	e-WALD @RUN,

VERDICT

[RO]:
- Not part of SDR but discoverable in INEO could be part of WP workplan

2.2.5 Linguistic Corpus Search: Text & Annotation Search

(Maarten, WP3)

User story:

As a scholar, I want to perform complex searches in text collections/corpora and in the annotations on these collections **in order to** find patterns of specific (often linguistic) constructs for my research purpose.

(2) **As a scholar**, I want to view aggregated results over my results sets, such as distributions, grouped results and statistics **in order to** be able to analyse my data and identify common trends

(3) **As a scholar**, I want to provide my own text collections **in order to** have a platform that enables me to search in them.

(4) **As a scholar,** I want to search in syntactically annotated corpora (treebanks) **in order to** find linguistic patterns for my research purpose. *[this is a more specific instance of the main user story]*

(5) **As a scholar,** I want to automatically enrich my corpus with specific linguistic annotations **in order to** find linguistic patterns for my research purpose.

(6) **As a scholar,** I want uniform and rich access to a large and diverse set of corpora (possibly within a certain domain) **in order to** have a big enough data set to do searches on for my research.

Implementations & Software Components

Implementation 1: INT (implements all three stories, will implement level 4 functionality to a certain degree in the project (T83). Does not really implement 6)

Component	Function(s)	Instance @Provider
Blacklab (using Apache Lucene)	• Storage engine for text and annotations • Query & search engine • Indexer to process text corpora with annotations (in specific formats)	AutoSearch @INT OpenSoNaR @INT
Blacklab Server	• Web API	
Corpus-frontend	• A search front-end to formulate and execute queries • A results front-end to show matches in the corpus, complete with annotations • An upload front-end for users to add their own data	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level

8?		

Implementation 2: Nederlab (implements 6 (for historical dutch texts) and 2, does *not* implement: 3, 4, 5)

Component	Function(s)	Instance @Provider
MTAS	• Storage engine for text and annotations • Query & search engine • Web API • Indexer to process text corpora with annotations (in specific formats) • A search front-end to formulate and execute queries • A results front-end to show matches in the corpus, complete with annotations •	Nederlab Portal @KNAW-HuC
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
?		

Implementation 3: GreTeL (implements the more specific story 4 (treebanks) rather than the more generic story, also implements 2 & 3 & 5. Does not implement 6). This differs from implementation 4 (PaQu) in having more advanced query facilities.

Component	Function(s)	Instance @Provider
GreTeL	• Storage engine for treebanks • Query & search engine for treebanks • Indexer to process treebanks with annotations (in specific formats) • A search front-end to formulate and execute queries • A results front-end to show matches in the corpus, complete with annotations • An upload front-end for users to add their own data	GreTeL4 @UU
Alpino	• Automatic dependency parsing	
Technology	Stakeholder Readiness Level (SRL)	Compatibility

Readiness Level (TRL)		Level
8?		

Implementation 4: PaQu (implements the more specific story 4 (treebanks) rather than the more generic story, also implements 2 & 3 & 5. Does not implement 6)

Component	Function(s)	Instance @Provider
PaQu	• Storage engine for treebanks • Query & search engine for treebanks (using simply XPath) • A search front-end to formulate and execute (XPath) queries • A results front-end to show matches in the corpus, complete with annotations • An upload front-end for users to add their own data	PaQu @RUG
Alpino	• Automatic dependency parsing (see also 2.5.1)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
8?		

Data Components

Implementation 1: INT

Component (specific)	Function	Instance @Provider
SoNaR corpus	Indexed & searchable collection (with various linguistic annotations)	OpenSoNaR @INT
CGN corpus	Indexed & searchable collection (with various linguistic annotations)	OpenSoNaR @INT
Corpus hedendaags Nederlands	Regularly updated monitor corpus of modern Dutch, cf corpus composition	CHN @INT
Corpus Gysseling (13e eeuw); Corpus Middelnederlands (14e-16e eeuw);	Indexed & searchable collections; historical dutch corpora	

Brieven als Buit. Corpus of old Dutch forthcoming.		

Implementation 2: Nederlab

Component (specific)	Function	Instance @Provider
Approx 20 large and small text collections, in uniform format (FoLiA XML), spanning 8 centuries (18.5 billion words, approx 80 billion annotations)	Indexed & searchable collections (with various linguistic annotations)	Nederlab Portal @KNAW-HuC

Implementation 3: GreTeL

Component (specific)	Function	Instance @Provider
AnnCor CHILDES treebank for dutch	Indexed & searchable treebank	

Interoperability Standards

Implementations 1 & 2 (INT & Nederlab):

Component	Function	Provider
TEI XML	Data format supported for indexing (limited)	TEI Consortium
FoLiA XML	Data format supported for indexing (limited)	KNAW-HuC

Implementations 3 & 4 (GreTeL & PaQu):

Component	Function	Provider
CHAT	Data format for ingestionmathieu_concepts	CHILDES corpus project
Alpino XML	Data format for ingestion	RUG

FoLiA XML	Data format for ingestion (limited)	KNAW-HuC

Wider Context

Nederlab was an independent NWO project that has finished, it has strong relations with CLARIAH due to the partners involved being in both projects. The MTAS maintainer/lead developer has left KNAW-HuC so current MTAS maintenance/development & sustainability is an issue.

Use cases

-

VERDICT

[RO]:
- NOT in SDR, part of FAIR datasets and responsibility of domains/WPs
- Possibly part of WP workplan but requires argumentation

2.2.6 Media Suite

User story:

As a media scholar, I want to access data and tools **in order to** analyse large media archives and Oral History data.

Implementations & Software Components

Implementation:

Service	Function(s)	Instance @Provider
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level

Verdict		
UNDECIDED		

Data Components

Implementation:

Component	Function	Instance @Provider

Interoperability Standards

Implementation:

Component	Function	Instance @Provider

Wider Context

Use cases

-

2.3 Provisioning Services

These are the exception to the rule in the sense that they are not scholarly services, but are important facilitating services for the core infrastructure. They provide low-level pervasive functionality required by most other services (such as authentication, deployment)

2.3.1 Authentication & Authorization

2.4 Unclassified Services

These services have yet to be processed into one of the above categories, or are left here if we don't consider them part of the Shared Development Roadmap.

2.4.1 Metadata management

(Henk, WP3)

User story:
(1) **As a scholar**, I want to edit a metadata file for my deposited language resource/dataset/corpus via a web form **in order to** have rich metadata file description (2) **As a scholar,** I would like to have my metadata file harvested in the VLO **In order to** have it published in the CLARIN infrastructure

Software Components & Implementations

Implementation 1: **Collection Bank**
Implements: 1, 2

Component	Function(s)	Instance @Provider
Collbank	Webservice and user interface	*Collbank @RUN*
	Login registration facility	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
3		
Verdict		
[RO]: • **Not part of SDR** • **Possibly on WP workplan but requires argumentation**		

Implementation 2: **TLA-FLAT**
Implements: 1, 2

Component	Function(s)	Instance @Provider

CLARIAH CMDI Forms (WP3)	Edit CMDI records incl. Interaction with vocabularies	KNAW HuC
TLA-FLAT	Store CMDI records and resources, provide OAI access to the metadata	KNAW HuC/MI, MPI/TLA
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
3		
Verdict		
[RO]: • **Not part of SDR** • **Possibly on WP workplan but requires argumentation**		

Data Components

Implementation 1: **(some identifier)** *[<- see before]*

Component (specific)	Function	Instance @Provider
[Name of the actual data component, e.g. a particular data collection or data standard]	*[generic function/role the data fulfills in a specific pipeline; e.g. input corpus, lexicon]*	*[The service instance that provides this data and the provider (institution) that provides it]*

Interoperability Standards

Implementation: (some identifier) *[<- see before]*

Component	Function	Instance @Provider
CMDI	Metadata standard	*

Workflow schema

[if an implementation consists of many different components and their flow is not obvious, please draw a schema to illustrate the implementation(s)]
Wider Context
[here you can sketch how this service, either in general or specific implementations thereof, relate to a wider context, such as partner projects like CLARIN, NDE]
Use cases
• …. *[link to specific use cases for which this user story is relevant, the use cases must reside in https://github.com/CLARIAH/usecases and there should be at least one use case for every service]*

2.4.2 Federated linguistic corpus search: Text & Annotations

Description:

Enabling searching within collections over different institutes that maintain a search API.

User story:

As a scholar, I want to search within collections over different institutes of my choosing, to be used for further analysis and comparison

e.g.
(1) Federated search in token-based corpora
(2) Federated search in treebanks
(3) Federated search in lexica

Implementations & Software Components

Implementation : **CLARIN federated content search for corpora** (https://www.clarin.eu/blog/clarin-federated-content-search-version-2)

Component	Function(s)	Instance @Provider
Dutch corpora made available via FCS (by means of wrappers for blacklab and Nederlab)		(multiple instances) via FCS @CLARIN
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
?		
Verdict		
[RO]: • **Not on SDR as federated search is outside the scope of CLARIAH+ workplan**		

Implementation : **Federated search for treebanks** (https://www.clarin.eu/blog/clarin-federated-content-search-version-2)

Component	Function(s)	Instance @Provider

GreTeL	Treebank search supporting CLARIAH FCS	Gretel @INT via FCS @CLARIN
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
?		
Verdict		
[RO]: **• Not on SDR as federated search is outside the scope of CLARIAH+ workplan**		

Interoperability Standards

Component	Function	Instance @Provider
CLARIN FCS Specification	API Specification	*
Lemon-Ontolex	Linked data model for lexica	https://www.w3.org/2016/05/ontolex/

2.4.3 Alignment services

(Willem, WP5)

User story:
As a scholar I need to be able to align different models or ontologies, **in order to** be able to connect data from different (historical or research) sources, even if these datasets have different underlying data models or ontologies, **in order to** do cross collection research in structured data. (2) **As a scholar** I need to be able to load a data model and map the classes and properties to classes and properties from other data models, **in order to** produce an alignment file or linkset that allows me to normalize the data access.
Software Components & Implementations

• Cultuurlink (WP5) • Amalgame (WP5) • Lenticular Lens (WP2) • BurgerLinker (WP4) • Relation Registry (WP3)
Data Components

2.4.4 Reconciliation services

(Willem, WP5)

User story:
As a scholar, I want to be able to link named entities extracted from my (multimedia or text) source documents to identifiers from (terminology) sources like authority files or reference databases, **in order to** connect the enriched (multimedia) sources to other (historical) sources that are also linked to the same entities and identifiers. (2) **As a scholar** I want to be able to determine that named entities within a single source or within several different sources are identical, even if the lexical spelling of a name might be slightly different, **in order to** avoid mistakes with homonyms, e.g. persons having the same name, but representing completely different persons. (3) **As a scholar** I want to be able to assess the representation of some named entity in alternative sources, **in order to** get additional information that helps me in answering my research questions. (4) **As a scholar**, I want to be able to rethink my data and choose appropriate existing identifiers from some other sources, like authority files, thesauri or geocoding using a well known geographical database, **in order to** be able to do cross collection research. (5) **As a scholar** I need to be able to post-process transcriptions (either generated by manual transcription or AI) of multimedia sources, including text, images, audio and video, using reconciliation **in order to** facilitate search or filtering of my data for people, places or other named entities without having to tweak free text search queries.
Software components & Implementations
• OpenRefine • Network of terms • MixnMatch • OpenReconcile
Data Components
• Network of terms catalog

- Wikidata

2.4.5 Text & linguistic annotation

(Maarten WP3 & Jesse WP3)

User story:

As a scholar, I want to add (linguistic) annotations to text documents (either my own data or a data collection stored elsewhere), for instance part-of-speech tags, named entities, dependency relations, semantic roles, etc..

(2) **As a scholar**, I want to manually correct output from an automatic annotation system (or do manual spelling correction in general) **in order to** develop a ground truth set, for example.

(3) **As a scholar,** I want to do bulk assignment in a complete corpus (keyword in context) **in order to** quickly and efficiently tag larger collections

(4) **As a scholar,** I want to use a collaborative annotation environment **in order to** have multiple annotators edit the same data (or split data between multiple annotators), allowing me to compute things like inter-annotator agreement at a later stage

Implementations & Software Components

Implementation 1: **FLAT**
Implements, 1,2,4 (does not implement 3)

Component	Function(s)	Instance @Provider
FLAT	Web-based annotation environment for advanced linguistic annotation, document-based (frontend)	FLAT @RUN
foliadocserve	Document + annotation server, backend	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
8		

Verdict
UNDECIDED

Implementation 2: **COBALT** corpus-based annotation tool - new version
Implements: 1,2,3,4

Component	Function(s)	Instance @Provider
COBALT annotation tool	Annotation environment for basic linguistic annotation, oriented towards bulk processing	@INT *(not deployed yet)*
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
6?		
Verdict		
UNDECIDED		

Implementation 3: Annotation of corpus search results (WP3 T085)

Component	Function(s)	Instance @Provider
Annotation of corpus results	Remote annotation functionality on corpus search results; annotations are also made searchable in the corpus exploitation environment	@INT *(not deployed yet)*
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
2		
Verdict		
UNDECIDED		

Implementation 4: **TextFabric**

Data Components

Interoperability Standards

Implementations 1: **FLAT**

Component	Function	Instance @Provider
FoLiA	Input and output data format	*

Implementations 2 & 3: **COBALT**

Component	Function	Instance @Provider
TEI	Input and output data format	*

Wider context

- In addition to the tools developed in CLARIAH, there is a wide range of 3rd party annotation tools for text and annotations. A few notable examples are brat, webanno, doccano, recogito.

2.4.6 High resolution annotation of audiovisual data

(Roeland, WP5)

User story:

As a scholar, I want to do high-resolution (accurate frame selection) annotations in audiovisual content on a local machine on data in the CLARIAH infrastructure

Implementations & Software Components

Implementation 1: ELAN
Implements:

Component	Function	Instance @Provider
ELAN	Annotation tool for audio and video recordings (desktop GUI)	n/a
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
9?		
Verdict		
UNDECIDED		

Data Components
Implementation 1: Implements:

Component	Function	Instance @Provider

Interoperability Standards
Implementation:

Component	Function	Instance @Provider

Wider Context
Use cases
•

2.4.7 FAIR IIIF-style publication of collections

User story:
As a scholar, I ……. **in order to** ……

(1) **As a collection provider**, I want to publish my collections in such a way that anybody can persistently refer to the data and segments of the data, **in order to** enable users to annotate those with web annotations or to retrieve those data segments for further processing or analysis.

(2) **As a scholar,** I want to be able to select (parts of) collection objects **in order to** annotate these selections or to further process these selections.

(3) **As a scholar**, I want to be able to refer to these selections persistently **in order to** be sure that my annotations stay valid and resolvable for a long time.

Software Components & Implementations

Implementation 1: **W3C IIIF Image service**
W3C compliant externally developed open source implementations of the IIIF image API

Implements: 1, 2

Component	Function(s)	Instance @Provider
Externally developed open source IIIF image server	Retrieve or refer to images and image parts, complying to the W3C IIIF protocol	@HuC instance at https://images.diginfra.net/iiif/ Probably other partner host IIIF image servers
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Implementation 2: **W3C IIIF Presentation service**
W3C compliant externally developed open source implementations of the IIIF Presentation API. This service publishes IIIF Canvases organised in Manifests.

Implements: 1, 2

Component	Function(s)	Instance @Provider
IISG implementation of IIIF Presentation API	Publish image collections using Canvases and Manifests	
HuC-DI implementation of IIIF Presentation API	Publish image collections using Canvases and Manifests	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level

Verdict		
UNDECIDED		

Implementation 3: **W3C IIIF audiovisual extension**

Extensions of the IIIF Image protocol for audiovisual data, to be able to also address time intervals.

Implements: 1, 2

Component	Function(s)	Instance @Provider
Beeld en Geluid IIIF extensions	Retrieve and point at parts of audiovisual streams	@Beeld en Geluid?
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Implementation 4: **W3C IIIF geo extensions**

Extension of the IIIF Image protocol for geographical maps.

@Thomas Vermaut: can you fill in the details?

Implements: 1, 2

Component	Function(s)	Instance @Provider
?		

Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Implementation 5: **IIIF-style text service**

This service represents annotated or enriched text documents as a UTF-8 text stream that is segmented using (persistent) anchors between the text segments, plus stand-off annotations. These anchors can be used to attach web annotations.
Text segments can be retrieved or referred to using IIIF style urls, that are resolved by the text service.

Implements: 1, 2

Component	Function(s)	Instance @Provider
TextRepository	Store different serialisations and versions of text documents and provide IIIF-style API access to text segments.	@HuC
un-t-ann-gle	Extract a ‘text stream plus stand-off annotation’ representation of annotated text collections and documents	@HuC
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Implementation 6: **TextFabric**

Similar to implementation 5, TextFabric also represents annotated text as raw text plus

standoff annotations. It does not use Web Annotations but its' own simple and compact format. It also takes the approach to publish a text coordinate system that can be used to add independent sets of standoff annotations.

Implements: 2

Component	Function(s)	Instance @Provider
TextFabric	Convert text collections to text plus standoff annotations and add annotation sets.	@DANS
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Data Components

Implementation 1:

Component (specific)	Function	Instance @Provider

Interoperability Standards

Implementation: 1, 2, 3, 4

Component	Function	Instance @Provider
W3C IIIF Image Protocol	Protocol, API	
W3C IIIF Presentation	Protocol, API	

Protocol		
W3C Web Annotations	Format, Protocol	

Workflow schema

Wider Context

IIIF for image collections is used more and more by collection providers and provides a standard and solid basis for manual annotation of online collections of the CLARIAH partners and other collection providers. A clear omission that is highly relevant for scholars is the ability to include annotation of text collections. This service addresses this omission.

Use cases

- Annotatable collections

2.4.8 FAIR Annotation Repository

User story:

(1) **As a scholar,** I want to store, search, update, share, persistently publish, manage and exchange sets of web annotations.
(2) **As a collection manager,** I want to store, search, update, share, persistently publish, manage and exchange sets of web annotations.
(3) **As a builder of annotation tools,** I want to store, search, update, share, persistently publish, manage and exchange sets of web annotations.

Software Components & Implementations

Implementation 1: **Extended version of eLucidate Web Annotation Server**

Implements: 1, 2, 3

Component	Function(s)	Instance @Provider

Extended eLucidate Web Annotation Server	Implements W3C web annotation protocol and annotation search	@HuC
Python client for eLucidate	Provide easy access to elucidate API for Python developers	@HuC
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Implementation 2: **CLARIAH WP2 Web Annotation Server: annotation.clariah.nl**

Implements: 1, 2, 3

Component	Function(s)	Instance @Provider
Python server for scholarly web annotations	Store and search web annotations	@HuC
Demos with front-end	Proof of concepts	@HuC
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
Verdict		
UNDECIDED		

Data Components

Implementation 1:

Component (specific)	Function	Instance @Provider

Interoperability Standards

Implementation: 1, 2

Component	Function	Instance @Provider
W3C Web Annotation Data Model and Protocol standards	API, format	

Workflow schema

Wider Context

Use cases

- Store, share and search web annotations

2.4.9 Natural Language Processing (automatic annotation)

(Maarten, WP3)

User story:

As a scholar, I want to automatically enrich my texts with linguistic annotation **in order to** facilitate further processing/analysis/visualisation steps for my research.

(2; tok) **As a scholar**, I want to automatically tokenize my text
(3; pos) **As a scholar**, I want to automatically enrich my text with part-of-speech tags
(4; lem) **As a scholar**, I want to automatically lemmatise my text

(5; dep) **As a scholar**, I want to automatically extract dependency relations on my text
(6; syn) **As a scholar**, I want to automatically extract syntactic constituents in my text, e.g. syntax trees.
(7; mor) **As a scholar**, I want to automatically decompose tokens into morphemes
(8, ner) **As a scholar,** I want to automatically recognize named entities in my text
(9, nl) **As a scholar**, I want to automatically enrich *dutch* texts
(10, hist) **As a scholar**, I want to automatically enrich *historical* dutch texts
(11, en) **As a scholar**, I want to automatically enrich *english* texts
(12, fy) **As a scholar**, I want to automatically enrich *frisian* texts
(13, loc) **As a scholar**, I want to be able to run the enrichment tools myself on my own system, and not as a service **in order to** enable more efficient execution or integrate it in my own pipeline.
(14, id) **As a scholar**, I want to automatically identify the language of (parts of) a text

Note: Visualisation of the results is not included in this service, but is covered by a separate service: Corpus Search: Text & Annotation service. Further manual annotation/correction is also not covered by this service but by others.

Implementations & Software Components

Implementation 1: **Frog**
Implements: 2 (tok), 3 (pos), 4 (lem), 5 (dep), 7 (mor), 8 (ner), 9 (nl), 10 (hist, but only for 2 and 3), 13 (loc)
Note: Interoperability is provided with FLAT (see 2.4.5) for visualisation of results and further manual annotation

Component	Function(s)	Instance
Frog	Tagger for Automatic Linguistic Enrichment	Frog @RUN
Frog python binding	Use of frog from Python (optional)	
Frog webservice (powered by CLAM)	* Web API * Upload, processing web app (optional)	
Ucto	Tokenization library	
LaMachine	Distribution & Deployment solution (optional)	
clamopener	Basic authentication provider (non-federated, to be replaced)	
Technology Readiness	Stakeholder Readiness Level	Compatibility Level

Level (TRL)	(SRL)	
9		
Verdict		
UNDECIDED		

Implementation 2a: **Alpino @RUN**
Implements: 2 (tok), 3 (pos), 5 (dep), 9 (nl), 12 (vis), 13 (loc)
Note: Interoperability is provided with FLAT (see 2.4.5) for visualisation of results and further manual annotation

Component	Function(s)	Instance
Alpino	Dependency Parser for Dutch	Alpino @RUN
Alpino webservice (powered by CLAM)	* Web API * Upload, processing web app * Conversion from/to FoLiA (optional)	
LaMachine	Distribution & Deployment solution (optional)	
clamopener	Basic authentication provider (non-federated, to be replaced)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
9		
Verdict		
UNDECIDED		

Implementation 2b: **Alpino via PaQu @RUG**
Implements: 2 (tok), 3 (pos), 5 (dep), 9 (nl), 13 (loc)

Component	Function(s)	Instance
Alpino	Dependency Parser for Dutch	PaQu @RUG
PaQu	Upload, processing web app (no web API) (other functions are beyond the	

	scope of this user story, see corpus search)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
9		
Verdict		
UNDECIDED		

Implementation 3: **ucto**
Implements: 2 (tok), 9 (nl), 11 (en) , 10 (hist), 13 (loc), 14 (id)
Note: Interoperability is provided with FLAT (see 2.4.5) for visualisation of results and further manual annotation

Component	Function(s)	Instance
Ucto	Tokenizer	Ucto @RUN
Ucto python binding	Use of ucto from Python (optional)	
Ucto webservice (powered by CLAM)	* Web API * Upload, processing web app (optional)	
libexttextcat	Language identification	
LaMachine	Distribution & Deployment solution (optional)	
clamopener	Basic authentication provider (non-federated, to be replaced)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
9		
Verdict		
UNDECIDED		

Implementation 4: **UDPipe-Frysk**
Implements: 2 (tok), 4 (lem), 5 (dep), 12 (fy), 13 (loc)

Component	Function(s)	Instance

Frisian UDPipe (powered by UDPipe)	Tagger	?
UdPipeService	Web API for UDPipe-Frysk (optional)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
?		
Verdict		
UNDECIDED		

Implementation 5: **DeepFrog**
Implements: 2 (tok), 3 (pos), 4 (lem), 9 (nl), 13 (loc)
Note: Interoperability will be provided with FLAT (see 2.4.5) for visualisation of results and further manual annotation

Component	Function(s)	Instance
DeepFrog	Tagger for Automatic Linguistic Enrichment	Not available yet
LaMachine	Distribution & Deployment solution (optional)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
6		
Verdict		
UNDECIDED		

Implementation 6: **VU Named Entity Detection**
Implements: 8 (ner), 10 (hist)

Component	Function(s)	Instance
?		
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
?		

Verdict
UNDECIDED

Data Components

Implementation 1: **Frog**

Component (specific)	Function	Instance @Provider
frogdata	Models for Frog	Frog @RUN
uctodata	Tokenization rules for ucto	Frog @RUN

Implementation 3: **Ucto**

Component (specific)	Function	Instance @Provider
uctodata	Tokenization rules for ucto	Ucto @RUN

Implementation 4: **UDPipe-Frysk**

Component	Function	Instance @Provider
Frisian UDPipe Model	Tagging Model	?

Implementation 5: **DeepFrog**

Component	Function	Instance @Provider
RobBERT v2 Part-of-Speech (CGN tagset) for Dutch	Fine-tuned Transformer Model (Roberta) (compatible with huggingface transformers)	Not available yet
BERT Part-of-Speech (CGN tagset) for Dutch (uses BERTje)	Fine-tuned Transformer Model (BERT) (compatible with huggingface transformers)	Not available yet
RobBERT v2 SoNaR Named Entities for Dutch	Fine-tuned Transformer Model (Roberta) (compatible with huggingface transformers)	Not available yet

BERT SoNaR Named Entities for Dutch (uses BERTje)	Fine-tuned Transformer Model (Roberta) (compatible with huggingface transformers)	Not available yet

Interoperability Standards

Implementation 1, 2a & 3:

Component	Function	Instance @Provider
FoLiA	Output data format	*
CLAM	RESTful web-API specification	*
Oauth2 & OpenID Connect	Authentication/Authorization Protocol	Implemented but not used yet

Implementation 2a & 2b: Alpino

Component	Function	Instance @Provider
Alpino XML	Output data format	*

Implementation 5: Deepfrog

Component	Function	Instance @Provider
FoLiA	Output data format	*
Huggingface Transformers / Pytorch	Neural model format	*

Wider Context

- There is a vast amount of 3rd party NLP tools available, the implementations in this CLARIAH service concern only tools that were either built in CLARIAH (or predecessors) or are hosted in a CLARIAH context.

Use cases

- Automatic linguistic enrichment for Dutch texts using Frog

2.4.9.1 NLP Suite

Definition: We define a natural language processing *suite* as a high-level application (usually a web-based platform) geared towards scholarly end-users (with limited technical skills) that gives the user the ability to select and possibly compose one or more NLP processing components integrated within the suite; additionally the integration offers additional facilities such as a built-in visualisation/analysis component for the NLP output.

Note: This has a certain overlap with tool discovery (2.7.1), but focus here is on the NLP aspect and the integration rather than on tool discovery as such. There may also be overlap with CLARIAH services on annotation and search.

(Jesse, WP3, edited by Maarten)

User story:

As a scholar, I am looking for an integrated application that allows me to apply various automatic linguistic enrichments on data I provide and allows me to visualize the results

(2; eval) **As a scholar,** I am looking for a platform that allows me to compare tagged data with a gold standard, and present an analysis **in order to** help me evaluate a tagging model.

(3; pos) **As a scholar**, I want to automatically enrich my text with part-of-speech tags
(4; lem) **As a scholar**, I want to automatically lemmatise my text

(5, hist) **As a scholar**, I want to automatically enrich *historical* dutch texts

Implementations & Software Components

Implementation 1: **Galahad platform**
Implements: 2,3,4,5
Note: This platform is specifically focused on historical dutch

Component	Function(s)	Instance @Provider
Galahad	• Management platform for enrichment and evaluation. • Provides a non-expert user interface for annotation and evaluation	Galahad @INT (not yet deployed)
Frog (cf previous, both modern and historical)	Tagging and lemmatization (option)	
PIE	Tagging and lemmatization	

	(option)	
?Helmut Schmid tagger	Tagging and lemmatization (option)	
INT historical tagger	Tagging and lemmatization (option)	
?Huggingface basic transformer-based tagger using historical BERT models[2] (proposed)	Tagging and lemmatization (option)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
4?		
Verdict		
UNDECIDED		

Data Components

Implementation: Galahad

Component	Function	Instance @Provider
PIE, frogdata and other models	Models necessary to run the tools	*
Ground truth data	Data necessary to train the tools	

Wider Context

- There are also various 3rd party NLP suites that would fit this definition and integrate multiple tools. Within CLARIN-D, Weblicht comes to mind as a notable example. GATE may also be a contender in a broader context.
- (Failed) attempts towards a WP3 Virtual Research Environment would also partially fall in this category.
- The NLP tools offered in LaMachine, most notably Frog in combination with FLAT for visualisation and/or further annotation, but also PICCL, could also be considered a kind of NLP suite, but they have not been separately listed here as another more minimally described CLARIAH service felt more fitting for htose.

[2] (to be developed when historical Dutch BERT model is available)

2.4.9.2 Spelling correction/normalisation (OCR/HTR post-correction)

(Maarten, WP3)

User story:

As a scholar, I want to correct or normalize text **in order to** make it more suitable (less errors/noise) for further-processing, whatever that may be; e.g. indexing for search, NLP processing, etc..

(2) **As a scholar**, I want to automatically clean up errors in OCR/HTR output

(3) **As a scholar**, I want to normalize diachronic variation in spelling

Implementations & Software Components

Implementation 1: **PICCL**
Implements: 1,2,3

Component	Function(s)	Instance @Provider
TICCL (part of PICCL)	OCR post-correction (workflow script)	PICCL @RUN or PICCL @INT (outdated)
TICCL-tools	OCR post-correction (set of tools)	
Nextflow	Workflow engine	
• FoLiAutils (hOCR to folia) • foliatools	Data conversion	
PICCL service, powered by CLAM	* RESTful webservice layer * Upload, processing front-end	
clamopener	Basic authentication provider (non-federated, to be replaced)	PICCL @RUN
FLAT (document-based)	Result visualisation front-end (optional)	FLAT @RUN via PICCL @RUN
LaMachine	Distribution & Deployment	

	solution (optional)	
Shibboleth	Federated auth solution	Via PICCL @INT
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
7		
Verdict		
UNDECIDED		

Note: OCR is listed as a separate service, but both are implemented as part of PICCL

Implementation 2: **Analiticcl**
Implements: 1,2,3
Note: This is a reimplementation of the core ideas of ticcl

Component	Function(s)	Instance @Provider
Analiticcl	OCR-post-correction (matching spelling variants against preferably validated lexica)	None yet
Analiticcl service, powered by CLAM	Webservice and user interface	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
5		
Verdict		
UNDECIDED		

Data Components

Implementation: PICCL

Component (specific)	Function	Instance @Provider
Aspell lexicons	Lexicons	*

Interoperability Standards

Implementation: PICCL

Component	Function	Instance @Provider
FoLiA	Output data format	*
CLAM	RESTful web-API specification	*
Oauth2 & OpenID Connect	Authentication/Authorization Protocol	(Available but not used)

Wider Context

- Implementation PICCL: This was developed in CLARIAH-CORE and CLARIAH-PLUS WP2 & WP3, successor of the earlier TICCLops in CLARIN-NL. However, current funding for PICCL has ended and a main developer has retired. Continuation of this implementation depends on Martin Reynaert (UvT)

Use cases

2.4.9.3 Grapheme to Phoneme Conversion

(Maarten, WP3)

User story:

As a scholar, I want to get a phonetic representation of a text **in order to** use it for further analysis or speech synthesis

(2) **As a scholar**, I want dutch phoneme conversion
(3) **As a scholar**, I want english phoneme conversion

Implementations & Software Components

Implementation 1: **g2p**
Implements: 1,2,3

Component	Function(s)	Instance @Provider
Phonetisaurus G2P	Backend system	G2P @RUN
g2p service, powered by CLAM	* RESTful webservice layer * Upload, processing front-end	
clamopener	Basic authentication provider (non-federated, to be replaced)	
LaMachine	Distribution & Deployment	

	solution (optional)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
7		
Verdict		
UNDECIDED		

Data Components

Interoperability Standards

Implementation: PICCL

Component	Function	Instance @Provider
CLAM	RESTful web-API specification	*
Oauth2 & OpenID Connect	Authentication/Authorization Protocol	(Available but not used)

Wider Context

-

Use cases

2.4.10 (Annotated) Text Conversion

(Maarten, WP3)

Description:

Convert (annotated) text documents between various formats.

User story:

As a scholar, I want to convert my (annotated) text document from one format to another. **In order to** use my data with a tool that requires a different format, or because I want to store and archive it in a different format.

Implementations & Software Components

Implementation 1: **Piereling**

Component	Function(s)	Instance
Foliatools • tei2folia (TEI to FoLiA conversion, limited) (TRL 7) • Conllu2folia • folia2columns - Simple conll-like output • Folia2txt • Folia2html • folia2rst - FoLiA to ReStructuredText • folia2salt - FoLiA to Salt (TRL 3) • rst2folia -ReStructuredText to FoLiA XML	Conversion tools around FoLiA (cli)	Piereling @RUN
FoLiAutils • FoLiA-page - Page XML to FoLiA XML • FoLiA-alto - ALTO XML to FoLiA XML • FoLiA-hocr - Tesseract hOCR to FoLiA XML • FoLiA-abby - Abby to FoLiA XML • FoLiA-2text	Conversion tools around FoLiA (cli)	
• Folia2naf (TRL 5) • Naf2folia (TRL 3)	Conversion tools for FoLiA<>NAF	
Pandoc (3rd party)	Conversion tool/library for various common document formats. Used for OpenDocument, Word,	

	Markdown, ReStructuredText	
Piereling (powered by CLAM)	* Web API * Upload facilities (web app) * Conversion pipeline/workflow	
clamopener	Basic authentication provider (non-federated, to be replaced)	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
8		
Verdict		
UNDECIDED		

Implementation 2: **OpenConvert**

Component	Function(s)	Instance
OpenConvert	Conversion library, web API and web app Supports: • FoLiA (TRL 3) • TEI (TRL 5?) • Alto • Word	OpenConvert @INT
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
5		

Interoperability Standards

Implementations 1 & 2:

Component	Function	Instance @Provider
FoLiA XML	I/O Data format for annotated text	*
TEI P5	I/O Data format for annotated text (input only for implementation 1)	*

Word	I/O Document format	*
OpenDocument	I/O Document format	*
ALTO XML	Input data format for layout and text objects	*
Plain text	I/O Most elemental text format	*

Implementation 1 only: Piereling

Component	Function	Instance @provider
ReStructuredText	I/O Text markup format	*
Markdown	I/O Text markup format	*
Abbyy XML	OCR-engine data format (ABBYY Finereader) (input only)	*
hOCR	OCR-engine data format (Tesseract) (input only)	*
Salt	Meta-model for linguistic annotation (output only for now)	*
CONLL-U	Column-based format for linguistic annotation (I/O)	*
HTML	Hypertext markup format (output only)	*
OAuth2 & OpenID Connect	Auth protocol	(implemented but not used yet)

Wider Context

- OpenConvert is significantly outdated when it comes to FoLiA support, the work in tei2folia, as included in Piereling, is a continuation of the conversion efforts that started at INT.

2.4.11 Linguistic Diagnostics Database (LIDIA)

(JO, WP3 LIDIA)

Service description:
Easily find arguments for or against a certain linguistic property
User story:
As a scholar, I want to quickly find all arguments from the literature for or against a linguistic property of a word or construction
Evaluation
• **Technology Readiness Level (TLR):** 2, working towards 3 • **Compatibility Level (CL):** ? • **Stakeholder Readiness Level (SRL)** ?
Software Components
• PICCL (to extract text from PDFs) • Annotation: annotate regions (in scans or PDFs) and associated text as a linguistic argument, with appropriate metadata • Database with arguments and associated metadata (to be developed) • User interface (to be developed) • Software to automate finding pieces of text that forms a linguistic argument (to be developed in a different/ successor project
Data Components
• Linguistic articles and books in PDF, Word etc.

2.4.12 Glossing Service

(JO, WP3: EXCALIBUR)

Service description:
Generate a gloss and a translation (into English) for an example sentence
User story:
As a scholar, I want to use Dutch examples in an English article and the gloss and translation should be added automatically
Evaluation
• **Technology Readiness Level (TLR):** 2, working towards 3 • **Compatibility Level (CL):** ? • **Stakeholder Readiness Level (SRL)** ?
Software Components
• PICCL (to extract example sentences from PDFs)

- Example sentence identifier (to be developed)
- Translation Memory- like techniques (to be developed in CP)
- Database with glossed examples and associated metadata (to be developed)
- User interface (to be developed)
- Language identification

Data Components

- Linguistic articles and books in PDF, Word etc.

2.4.13 Speech Recognition Services

(Roeland, WP5; Henk WP3)

Description:

There are several speech recognition related services requested by scholars, either directly (e.g., processing personal collections) or indirectly (e.g., enhance searchability of AV collections or browsing AV content).

User story:

As a scholar, I want to create a speech transcripts for an audiofile so that I have a textual representation of it for browsing/close reading

(2) **As a scholar**, I want to create speech transcripts for the audiovisual collection I'm interested in using automatic speech recognition (ASR) so that I can better search data that typically have limited descriptive metadata. The data I want to apply ASR on, could be a personal collection or (be part of) an institutional collection.

(3) **As a scholar,** I want to align my audio with its transcription, i.e. find what segments of the audio correspond with which text in the transcription.

(4;nl) **As a scholar,** I want to do Dutch speech recognition
(5;en) **As a scholar,** I want to do English speech recognition
(6;fy) **As a scholar,** I want to do Frisian speech recognition
(7;dl) **As a scholar,** I want to do Dutch dialect speech recognition

Implementations & Software Components

Implementation 1: Bulk processing
Implements: 2,4

Component	Function	Instance @Provider
DANE	Workflow manager	NIBG

Kaldi_NL worker	Kaldi implementation in DANE compatible worker format	NIBG
Elastic Search index	Needed for DANE to select data, retrieve audio and store transcripts	NIBG
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
8		
Verdict		
UNDECIDED		

Implementation 2: **Dutch ASR aka Oral History**, Stand-alone webservice/webapp for personal collections
Implements: 1,4

Component	Function	Instance @Provider
Kaldi-NL (powered by Kaldi)	ASR backend	Oral history @RUN
LaMachine	Deployment solution	
Oral History webservice (powered by CLAM)	Webservice front-end & web application	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
8		
Verdict		
UNDECIDED		

Implementation 3: **Forced alignment service**
Implements: 3

Component	Function	Instance @Provider
Forced Alignment	Webservice front-end & web application	Forced alignment

webservice (powered by CLAM)		@RUN
LaMachine	Deployment solution	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
7?		
Verdict		
UNDECIDED		

Implementation 5: English ASR for personal collections
Implements: 1,5

Component	Function	Instance @Provider
Kaldi	ASR backend	English ASR @RUN
LaMachine	Deployment solution	
English ASR webservice (powered by CLAM)	Webservice front-end & web application	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
8?		
Verdict		
UNDECIDED		

Implementation 6: Frisian-Dutch ASR for personal collections
Implements: 1,4,6
Note: multilingual implementation, supports code switching

Component	Function	Instance @Provider
Kaldi	ASR backend	Frisian-Dutch ASR @RUN
LaMachine	Deployment solution	
Frisian-Dutch ASR	Webservice front-end & web application	

webservice (powered by CLAM)		
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
7?		
Verdict		
UNDECIDED		

Implementation 7: Dutch dialect ASR for personal collections
Implements: 1,7

Component	Function	Instance @Provider
Kaldi	ASR backend	• To be created
LaMachine	Deployment solution	
Dialect-Dutch ASR webservice (powered by CLAM)	Webservice front-end & web application	
Technology Readiness Level (TRL)	Stakeholder Readiness Level (SRL)	Compatibility Level
1 -> 7		
Verdict		
UNDECIDED		

Data Components

Implementation 1 & 2

Component	Function	Instance @Provider
Kaldi-NL	ASR models (AM, LM, DCT) for dutch and generic	Oral history @RUN

Implementation 5

Component	Function	Instance @Provider
eng_ASR models	ASR models (AM, LM, DCT) for english	*

Implementation 6

Component	Function	Instance @Provider
Fy-nl-asr models	ASR models (AM, LM, DCT) for dutch and frisian	*

Implementation 7

Component	Function	Instance @Provider
dl-nl-asr models	ASR models (AM, LM, DCT) for dutch dialects (not yet available)	*

Interoperability Standards
Implementation 2,5,7:

Component	Function	Instance @Provider
CLAM	RESTful web-API specification	*
Oauth2 & OpenID Connect	Authentication/Authorization Protocol	Implemented but not used yet

Wider Context

Use cases
•

2.4.14 Computer Vision

(Roeland, WP5)

Description:
Using computer vision to automatically label video or images with images features such as objects, colour, shots, etc.
User story:
As a scholar, I want to use computer vision to explore data collections based on image features such as objects that are visible, shots or colours. The data I want to apply computer vision on, could be a personal collection or (be part of) an institutional collection.
Implementations & Software Components
Implementation 1: Bulk processing Implements:

Component	Function	Instance @Provider
DANE (task assignment and file storage for the automatic annotation of content)	Workflow manager	

DANE workers (see list of available computer vision DANE workers)	Computer Vision algorithm implementation in DANE compatible worker format	
Elastic Search index	Needed for DANE to select data, retrieve files and store transcripts	

Implementation 2:
Implements:

Component	Function	Instance @Provider

Data Components

Implementation 1:
Implements:

Component	Function	Instance @Provider

Interoperability Standards

Implementation:

Component	Function	Instance @Provider

Wider Context

Use cases
•

3 Software Components

This is a list of software components that are used by one or more CLARIAH services.
Instructions:

- Link the name of the software component to he software/service source repository, documentation, link instances to the service itself. The *only* exception where you can omit links is when the Technology Readiness Level (TLR) is so low (<2) that there is nothing to link to yet.
- Separate back-end components from front-end components wherever possible
- Separate specific *instances* of the software (services in the technical senses, hosted at a specific place) from the underlying software
- Components are categorized per work package (WP), either the WP where they are effectively developed, or the WP where a suggested new component should be housed.
- All existing components should already be listed in the CLARIAH Work Plan. The only components here which are not in the work plan should be a) components that are proposed for the future (mark them with the label “proposed”) and b) partner/3rd party components that are not developed in CLARIAH. The latter are in a separate subsection.
- The components are listed in no particular order (we will alphabetize it probably)

3.1 WP1

Name & Link	Description	TRL	CL	Partner
Ineo Ineo Instance	CLARIAH Portal			KNAW-Huc?

3.2 WP2

Name & Link	Description	TRL	CL	Partner
treafik	External proxy-layer K8s	3		KNAW-HuC
Rancher instance	Kubernetes orchestration platform	3		KNAW-HuC
K3s instance	Light-weight Kubernetes	3		KNAW-HuC
Satosa instance	OpenId/SAML Gateway	3		KNAW-HuC
GoHarbor	A docker registry and HELM registry for CLARIAH, including vulnerability scanning.	4		KNAW-HuC
Open annotation environment & offline annotation store (src)	Annotation FE and BE for creating and storing web annotations. (Koolen et al.)	4		KNAW-HuC
Kibana instance	A monitoring service for CLARIAH	4		KNAW-HuC
Lenticular Lens	Tool that allows users to construct linksets between entities from different Timbuctoo datasets (so called data-alignment or reconciliation).	7 / 8		KNAW-HuC
Datastories		2		KNAW-HuC
Datastories visualisation		2		KNAW-HuC

Datastories interoperability		2		KNAW-HuC
Datarepository	Describe or demonstrate how Small institutes/research groups are able to build a clariah compatible repository for storing objects (datasets, media, etc). This means that data is shareable within the Clariah community.	2		KNAW-HuC

3.3 WP3

Name & Link	Description	TRL	CL	Partner
CLAM	Webservice framework	9		KNAW-Huc
Ucto	Tokenizer and sentence splitter for text	9		KNAW-HuC
Ucto service instance	Webservice and Web UI for ucto (powered by CLAM)	8		KNAW-HuC & CLST RUN (current hoster)
Python-ucto	Python binding for ucto	9		KNAW-Huc
Frog	NLP Suite for dutch: (tokenization (via ucto),PoS,lemmatization, named-entity recognition, dependency parsing, constituent parsing,いややはり持って帰ることにすると言っ shallow parsing (chunking), morphological analysis)	9		KNAW-HuC
Frog service instance	Webservice and Web-UI for Frog (powered by CLAM)	8		KNAW-HuC & CLST RUN (current hoster)
Python-frog	Python binding for Frog	9		KNAW-Huc
Foliapy	Python library for working with FoLiA. Dependency for various other tools.	9		KNAW-Huc

libfolia	C++ library for working with FoLiA. Dependency for Frog and other tools.	9		KNAW-Huc
Folia-rust	Rust library for working with FoLiA. Dependency for DeepFrog.	5		KNAW-Huc
DeepFrog	NLP-suite for Dutch based on deep-learning: part-of-speech tagging, named entity recognition, lemmatization	5		KNAW-HuC
DeepFrog service	Webservice and web UI for DeepFrog (powered by CLAM)	2		KNAW-HuC
foliatools	Commanverdictd-line/python tools for working with FoLiA documents. (Python-based)	9		KNAW-Huc
foliautils	Command-line tools for working with FoLiA (C++), largely complementary and only partially overlapping with foliatools.	8		KNAW-Huc
Alpino	Dutch Dependency parser (including Part-of-Speech tagging, lemmatization)	9		RUG
Alpino service instance	Webservice and web UI for Alpino, with FoLiA support (powered by CLAM)	8		RUN
Frisian UDPipe	Part-of-Speech tagger for Frisian, R script and RESTful webservice	8		FA
udpipeService	Language independent rest service around UDpipe	8		FA
PICCL	OCR and post-OCR normalisation workflow using tesseract and TICCL.	6		UvT, RUN (hoster), INT (out of date hoster)
TICCL-tools	Low-level post-OCR normalisation tools that make up the TICCL workflow.	6		UvT
Blacklab	Backend for search over large text collections, including annotations	9		INT
Corpus frontend	Generic search frontend for blacklab	8?		INT
AutoSearch	Specific deployment of Corpus frontend for CLARIAH.	8?		INT
GrETEL	Search in syntactically annotated	8?		UU

	corpora (treebanks)			
PaQu	Search in syntactically annotated corpora (treebanks),	8?		RUG
FLAT	Collaborative web-based linguistic annotation tool (document-based, using FoLiA)	8		KNAW-Huc & CLST RUN (hoster)
foliadocserve	FoLiA Document Server, back-end for FLAT	9		KNAW-Huc & CLST RUN (hoster)
LaMachine	Meta-distribution / deployment-solution for a large amount of WP3 software. Both for end-users and hosters.	8		KNAW-Huc
e-WLD.nl	Web-based dictionary lookup of limburgish dialects	8		CLST RUN
e-WBD.nl	Web-based dictionary lookup of brabantish dialects	8		CLST RUN
e-WGD.nl	Web-based dictionary lookup of geldre dialects	8		CLST RUN
SPAQ	Speech acquisition using Surveys	?		KNAW-Huc
clam2switchboard	Tool for conversion of CLAM webservice specification to format expected by the switchboard registry. Enables semi-automatic harvesting.	7		KNAW-Huc / CLST RUN
Piereling service instance	Conversion between annotated text documents (focus on FoLiA)	7		KNAW-Huc / CLST RUN
PICCL	Web app, webservice and set of workflows for OCR and OCR post-correction (spelling correction)	7		UvT
Ticcltools	Set of command-line tools that make up the post-OCR correction system TICCL, used by PICCL.	6		UvT
spacy2folia	Thin wrapper around spaCy for FoLiA input/output support	6		KNAW Huc
RU-wnd	Web application for various dialect dictionaries	8?		CLST RUN
	Vocabulary overlays and alignments	?		KNAW HuC

Fluentd	Monitoring WP3 services			KNAW HuC
Clamopener + instance	Basic (legacy) authentication provider (non-federated, to be replaced)	8	D	CLST RUN
Dataverse				DANS
Collection Bank	Web application for metadata creation/curation			RUN

3.4 WP4

Name & Link	Description	TRL	CL	Partner
CSV On the Web (COW)	Command-line tool to convert CSV to RDF.	8		IISG
CATTLE	The server-based version of COW, which comes with a basic web UI to facilitate remote use.	8	D	
GRLC	A self-hosted web service to store and share SPARQL queries, facilitating easy reuse, and which offers a simple API. Actively used within and outside of CLARIAH. Still in active development.	9		KCL/eScience/IISG
Druid	A triple store with an intuitive frontend which provides tools for end-users to browse, query, and visualize RDF datasets, and which set the seed for what would later become TriplyDB. Still in active development.	9		Triply
LDWizard	Web-based alternative to COW with minimal user interaction and an intuitive interface. Still in active development.	8		Kadaster/NDE/Triply/IISG
burgerLinker	Tool to link records from different	7		IISG/HuC-DI

	civilian registries using the Levenshtein distance between names from birth, death, and marriage certificates, as well as a set of domain-specific heuristic rules. This tool expects linked data as input, and offers utility functions to convert CSV to that format.			
Vocab Recommender	Tool to suggest which vocabularies might be interesting to use when transforming CSV datasets to linked data. Suggestions are formed based on existing linked datasets, by querying the server using SPARQL or a REST API. Other sources of recommendations are community-maintained lists, such as the 'vocabulary registry'.	1		IISG/HuC-DI
Hypothesis Generator	A tool to identify interesting patterns in linked datasets that scholars and domain experts can use as starting points to form new research hypotheses, or as support for existing ones.	0		VU/IISG/HuC-DI
LDProxy	Proxy to access archived LD via there original URI, while this URI haa become dead on the WWW			KNAW HuC
data stories	A tool to make blog like texts with images, tables and graphs (incl. geographic representations), where such visual output is the result of a query, that can be edited on the fly.			Triply/IISG

3.5 WP5

Name & Link	Description	TRL	CL	Partner
DANE	Task assignment and file storage for the	7		NISV

	automatic annotation of content; includes these two repos: • DANE-server (API, task scheduler and UI) • DANE (Python library for the server and the workers)			
dane-download-worker	Downloads DANE input data (audio, video, text, etc) via HTTP URLs	7		NISV
dane-delete-worker	Deletes downloaded input data after processing is done	5		NISV
dane-asr-worker	Calls dane-kaldi-nl-api to process audiovisual content with Automatic Speech Recognition	6		NISV
dane-kaldi-nl-api	KaldiNL instance with API access; used by dane-asr-worker	5		NISV
dane-asr-to-folia-worker	Converts ASR output text into FoLiA format	5		NISV
dane-folia-to-ner-worker	Runs different NER tools on FoLiA input	4		NISV
dane-shot-detection-worker	Detects shots and keyframes (using Hecate)	6		NISV
dane-keypoint-worker	Detects keypoints (coordinates of joints in a body) as well as bounding boxes around persons	6		NISV
dane-object-classification-worker	Classifies images using the ResNet50 model as well as apply over a 1000 ImageNet classes	6		NISV
dane-pose-embedding-worker	Extracts embeddings (obtained via the dane-keypoint-worker) for pose retrieval	6		NISV
dane-colour-histogram-worker	Generates colour histograms from images	6		NISV
dane-image-embedding-worker	Detects features in images using the CLIP model	6		NISV
dane-optical-flow-worker	Detects optical flow in images	5		NISV
dane-dominant-colour-worker	Detects a ranking of the most dominant colours in an image	5		NISV
dane-text-detection-worker	Detects text in images (e.g. for frames from a video clip)	6		NISV

dane-cinemanet-worker	Detects cinematic features in images	5		NISV
DANE environments	Kubernetes configurations for setting up DANE processing environments (CLARIAH service candidates) involving one or more DANE workers: ● AV → ASR ● AV → ASR → FoLiA → NER Suitable for the CLAAS Helm repository	7		NISV
Search-API	This API supports authentication with SATOSA for the following endpoints: ● Collection registry access ● Elasticsearch query access ● Front-end dev friendly query access (to ES) ● Read ES documents ● SPARQL query access ● Grlc query access	9		NISV
Annotation API	This API supports authentication with SATOSA and is loosely based on the Web Annotation Protocol and Model. It offers: ● CRUD on annotations ● Simple search on annotations ● Bookmark groups ● Annotations can be grouped by project (see Workspace API)	9		NISV
Workspace API	This API supports authentication with SATOSA and provides a user workspace to: ● Store personal projects ● Store named queries (supported by the Search API)	9		NISV
Playout proxy	This proxy supports authentication with SATOSA and enables authorization for different content servers (e.g. B&G collection, EYE film collection, any other external media server): ● Secure play-out of AV streams ● Supports proxying IIIF image servers	9		NISV
Media Suite	Running CLARIAH service for the WP5 domain (and several generic use-cases	9		NISV

	therein). Built on top of the search, annotation & workspace APIs and playout proxy. Media Suite end-user tools implemented in React component library, which consists of: ● Resource viewer with manual annotation tool for AV and images ● Query compare tool ● Collection search tool ● Collection inspection tool			
MediaSuite-Collection Registry	CKAN instance mostly listing collections that are available in the Media Suite	9		NISV
BenG-LOD-server	Hosts B&G catalogue as Linked Open Data; includes the following: ● Dereferencing resources from DAAN catalogue ● SPARQL endpoint ● YASGUI/Comunica query editor with example sparql queries	7		NISV
Jupyter Hub connecting to Media Suite APIs	Python notebooks for (media) researchers that like to look beyond the Media Suite's capabilities. Accessible for a limited number of users on the B&G premises.	4		NISV
Media Suite data-stories	Stories from Dutch multimedia archives created using the Media Suite infrastructure (data and APIs)	9		NISV
ELAN	Annotation tool for audio and video recordings (desktop GUI)	9		MPI

3.6 WP6

Name & Link	Description	TRL	CL	Partner
Nederlab portal	Search & analysis environment for large set of diachronic text corpora			knaw-huc
MTAS (Nederlab search backend)	Advanced & scalable search engine for annotated text, implemented as plugin for SOLR			knaw-huc
un-t-ann-gle	Split annotated text documents in utf-8 text stream, stand-off annotations and metadata.			knaw-huc
Text Repository	Repository for (versions and serializations) of text documents. API extension to refer to/retrieve text segments (analogous to IIIF for images)			knaw-huc
PID services	Essential service for persistence of annotations.			wp2?
Showcase for Interactive micro-clients	Small/inexpensive clients for annotation scenarios, using text and web annotation services.			knaw-huc, all
Jupyter notebook micro-client	Small/inexpensive clients for annotation scenarios, using text and web annotation services.			knaw-huc, all
Rolodex web app	Web application that allows humanities scholars to collect collection references and annotate those. Supports registration of (not digitized, not online) collections, supports sharing of annotations.			knaw-huc
Collection registration service (proposed)	Simple service that allows registration of any collection (even if non-digital), to make a reference to it ‘resolvable’, and to support reference from annotations.			wp2?
Web annotation server	Elucidate web annotation server, with some improvements and extensions.			3rd party, knaw-huc
Python client for elucidate ann server	Python client library to support access to eLucidate server from a Python context.			knaw-huc

TextFabric	Collection annotation environment supporting independent standoff annotation layers for any user.			DANS
NER pipeline for historical text collections (per, loc, org, ships, commodities, values, amounts)	Derived from and applied to Generale Missieven			VU
Entity identification, context identification, co-occurrance for historical text	On Generale Missieven			VU
Surf Data Exchange	Platform for 'trusted data sharing'			3rd party, KB
Manual annotation tool (COBALT)	… for annotation of text corpora			INT
Docere	Generic framework for visualising text editions			HuC-DI
Text complexity module	Remark @hennie : add this to text processing services			VU

3.7 Partner Projects

Software from partner projects that have a relation with CLARIAH

Name & Link	Description	TRL	CL	Partner
Kaldi-NL	ASR Backend scripts, powered by Kaldi	7		SoS
Oral History Webservice	Dutch ASR system (webservice+webapp)	7		CLST RUN / SoS
Forced alignment	Forced alignment of audio and transcriptions (webservice+webapp)	7		CLST RUN
Grapheme-to-phoneme webservice	Phonetisaurus G2P (webservice+webapp)	7		CLST RUN

CLARIN Switchboard	Find and access suitable webservices/web applications (CLARIN-wide), usually given some user-uploaded input data.	8		CLARIN ERIC
Digital Methods Initiative	Consortium of new media researchers loosely affiliated with WP5. DMI also lists a great number of tools	8		UvA

3.8 3rd Party

Software from notable third-party projects that have no relation with CLARIAH and are being used as-is

Name & Link	Description	TRL	CL	3rd Party
TriplyDB				Triply
Tesseract	OCR system (used by PICCL)	9		
pandoc	Text conversion (used by Piereling)	9		
Jupyter Notebooks / Hub / Lab	Live coding in documents (notebooks)	9		
Kaldi	ASR system	9		
spaCy	NLP library for Python	9		
CKAN	Open source data management system	9		
Nextflow	Workflow engine for data-driven computational pipelines	9		
SKOSMOS	Web-based frontend for searching and browsing SKOS vocabularies	9		
Fluentd	Data collector for unified logging	9		
Harbor	OpenSource Registry for Kubernetes & Docker			
Kubernetes	Automated container deployment, scaling, and management	9		
Satosa	Proxy translating between different authentication protocols (SAML2, OpenID	8?		

	Connect and OAuth2)			
Kibana		9		
LimeSurvey		9		
ElasticSearch		9		
postgresql		9		
Matlab PyViz OpenStreetMap NetworkX GraphViz D3				
OpenRefine Network of terms MixnMatch OpenReconcile				
Cultuurlink Amalgame				
Clarin tools list	Named entity recognition software			

4 Data Components

4.2 WP2

Name & Link	Description	DRL	Partner

4.3 WP3

Name & Link	Description	DRL	Partner
Frog Data	Various trained models for dutch NLP tasks		KNAW-Huc
FoLiA	Schemas/model and XML-based Format for Linguistic Annotation.		KNAW-Huc
Ucto Data	Tokenization rules for various languages		KNAW-Huc
UDPipe-Frysk	Frisian part-of-speech model for UDpipe		FA
RobBERT v2 Part-of-Speech (CGN tagset) for Dutch	Fine-tuned Transformer Model (Roberta) (compatible with huggingface transformers; developed for DeepFrog)		RUN
BERT Part-of-Speech (CGN tagset) for Dutch (uses BERTje)	Fine-tuned Transformer Model (BERT) (compatible with huggingface transformers; developed for DeepFrog)		RUN
RobBERT v2 SoNaR Named Entities for Dutch	Fine-tuned Transformer Model (Roberta) (compatible with huggingface transformers; developed for DeepFrog)		RUN

BERT SoNaR Named Entities for Dutch (uses BERTje)	Fine-tuned Transformer Model (Roberta) (compatible with huggingface transformers; developed for DeepFrog)		RUN
CMDI Software metadata descriptions	Manually composed software metadata descriptions *(may be out of date!)*		UU

4.4 WP4

Name & Link	Description	DRL	Partner
Civil Registries schema	A linked data ontology to encode civil registry records with.	?	
Dutch Municipalities through Time	A linked geodataset about the history of municipalities from 1812 to present day.	?	DANS
CShapes	A linked geodataset that provides historical maps of state boundaries and capitals in the post-World War II period.	?	
hkh-maids-burgerLinker	A linked dataset of civil registry records. Used as tutorial dataset for BurgerLinker.	?	Historisch Kenniscentrum Harderwijk
Amsterdam Time Machine	A datastory to time travel through Amsterdam with linguistic, entertainment and social-economic data	?	
Growth and inequality	A datastory that asks questions about why some countries are rich and others are poor.	?	
Strikes	A datastory that looks into collective action in times of economic progress.	?	
The value of Occupations	A datastory that analyses ‘high’ and ‘low’ occupations in the past.	?	

The 'Spanish' Flu in The Netherlands	Discover the spatial, temporal, and social distribution of the 1918-19 flu epidemic in this datastory.	?	
The wealth of the Renaissance	A datastory about the wealth of the inhabitants of the Republic of Florence.	?	
Roman Catholics per municipality	A datastory that looks into the spatial distribution of Roman Catholics.	?	

4.5 WP5

Name & Link	Description	DRL	Partner
GTAA GTAA data	Dutch thesaurus for audiovisual archives	?	NISV
Open Data Lab	Open datasets from NISV: ● Open Beelden (Open Images) ● NISV metadata as LOD and SPARQL endpoint		NISV
Media Suite data	Metadata & AV content accessible via the Media Suite (requires login via CLARIAH/CLARIN federation/IdP): ● NISV catalogue (Dutch radio & television ● NISV photo archive ● NISV 2e kamer debates ● NISV Radio & TV ratings ● KB Delpher ● Dutch Television program guides (Metamorfoze) ● EYE Desmet film ● EYE Desmet posters ● EYE Desmet paper archive ● DANS oral history ● Open data sets from the Open Data Lab		NISV / KB / EYE / DANS

4.6 WP6

Name & Link	Description	DRL	Partner
		?	

4.7 Partner Projects

Data from partner projects that have a relation with CLARIAH

Name & Link	Description	DRL	Partner
Kaldi-NL	Dutch ASR models		SoS
hisgis.nl	Oorspronkelijk kadaster NL 1832		KNAW-HuC -DI + Fryske Akademy
CLARIN Switchboard Tool Registry	Tool registry for the CLARIN switchboard		CLARIN ERIC

4.8 3rd Party

Data from third-party projects have no relation with CLARIAH and are being used as-is

Name & Link	Description	DRL	3rd party
spaCy models	Various NLP models for various languages		
Wikidata Wikidata query service	Wikidata is a free and open knowledge base that can be read and edited by both humans and machines. The content of Wikidata is available under a free license, exported using standard formats, and can be interlinked to other open data sets on the linked data web.		Wikimedia Foundation

Appendix A: Technology Readiness Levels

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 colours and stages in the right column correspond to the vocabulary already used in the CLARIAH-PLUS workplan. We generally aim for at least TRL7.

TRL	Description	Stage
0	**Idea -** Unproven, untested and largely unformulated concept	**Planning (pre-alpha)**
1	**Initial Research -** Basic (scholarly) needs observed and reported	
2	**Concept Formulated** - Initial technology/application has been concept formulated	
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.	**PoC (alpha)**
4	**Validated PoC** - 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.	
5	**Early Prototype** - Technology validated in target setting (e.g. with potential end-users)	**Experimental (beta)**
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.	
8	**Complete** - Technology complete and qualified, released for all end-users in scholarly environments.	**Production (stable)**
9	**Proven -** Technology complete and proven in practice by real users.	

Appendix B: Compatibility Levels

CL	Description
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.

Appendix C: 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)

Appendix D: Infrastructure Requirements

(See https://github.com/CLARIAH/IG-DevOps/issues/4 & https://github.com/CLARIAH/IG-DevOps/pull/5 , work in progress)

Appendix E: Software Requirements

(See https://github.com/CLARIAH/IG-DevOps/issues/4 &
https://github.com/CLARIAH/IG-DevOps/pull/5 , work in progress)

Appendix F: Documentation Standard

(work in progress?)

Appendix G: Data Readiness Levels

(to be done)

Appendix H: Mission

[data] - [services] - [interfaces] - [community] - [stakeholders]

[data]
- ckan.clariah.nl

[services]
- See above

[interfaces]
- INEO
- CLARIAH Data Registry
- CLARIAH Digital Rolodex - CLARIAH VRE
- CLARIAH Media Suite
- CLARIAH Data Stories

[community]
- Mastodon.clariah.nl
- github/clariah

[stakeholders]
- clariah.nl