FAIR Tool Discovery

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

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 Stories

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

User stories 1-3 are not directly implemented by this epic/service, but are facilitated by it. This epic focusses on the data provisioning pipeline that makes these user stories possible.

Needs & Dependencies

- Compliance to the software/infrastructure requirements as described in the next section

- Cross-WP agreement on some additional vocabulary
- WP4 involvement

Requirements

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

Service Description

This core service provides infrastructure for finding tools, where the metadata for the tools is automatically and periodically harvested and converted to a unified representation. 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.

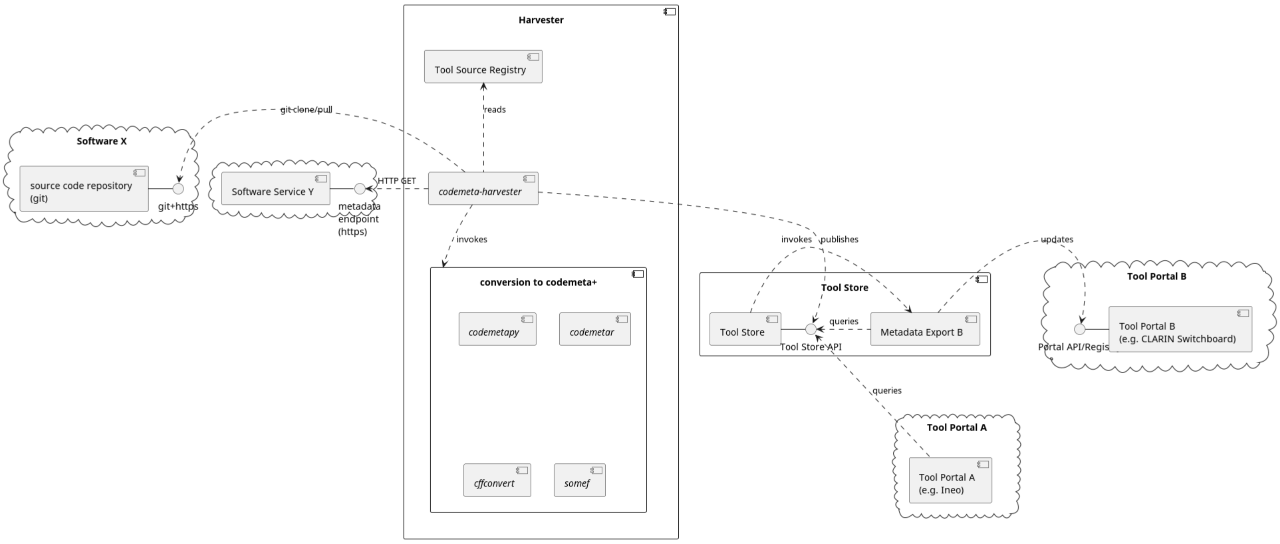
Components

Num	Name	Type	Description	Producer	Deps	URL
1	tool store	service	Queryable backend holding all tools; with simple front-end; facitilates user stories 1,3	KNAW HuC	5,3,11-14	https://tools.dev.clariah.nl
2	tool source registry	data	Input data for the harvester	KNAW HuC		https://github.com/CLARIAH/tool-discovery/tree/master/source-registry
3	codemeta-harvester	software	Harvester pipeline; ties togethervarious harvesting components. Implements story 4	KNAW HuC	4,6,7,11-14	https://github.com/proycon/codemeta-harvester
4	codemetapy	software	Conversion from various metadata formats to codemeta	KNAW HuC	4,11-14,16	https://github.com/proycon/codemetap
5	codemeta-server	software	triple store with API; SPARQL endpoint; simple web frontend	KNAW HuC	4,11-14	https://github.com/proycon/codemeta-server
6	cffconvert	software	CITATION.cff to codemeta conversion	3rd party	11-12	https://github.com/citation-file-format/cff-converter-python
7	codemetar	software	R to codemeta conversion	3rd party	11-12	https://github.com/ropensci/codemetar
8	somef	software	README parsing and to codemeta conversion	3rd party	11-13	https://github.com/KnowledgeCapture
9	CMDI export	software	Conversion/export from codemeta to CMDI. Implements story 5	undetermined	1	not started yet
10	CLARIN switchboard export	sofware	Conversion for CLARIN switchboard registry. Implements story 5, facilitates 2	undetermined	1	not started yet

Num	Name	Type	Description	Producer	Deps	URL
11	codemeta	standard	Linked Open Data vocabulary for describing software source code metadata	3rd party	12,15	https://codemeta.github.io
12	schema.org	standard	Linked Open Data vocabulary	3rd party		https://schema.org
13	repostatus.org	standard	Linked Open Data vocabulary for software status	3rd party		https://repostatus.org
14	spdx.org	standard	Linked Open Data vocabulary for software licenses	3rd party		https://spdx.org
15	CMDI	standard	Component Metadata Infrastructure used by CLARIN	CLARIN-ERIC		https://www.clarin.eu/content/cmdi-12
16	OpenAPI	standard	Web API specification	3rd party		https://www.openapis.org/

Note: Interoperability with Ineo will be handled at the Ineo side: https://github.com/CLARIAH/clariah-plus/issues/35

Workflow Schema



Evaluation

end Q1 2022 - Initial propotype has been delivered

Context

- Ineo is meant 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, as done by this epic.
- A system called CLAPOP was developed in CLARIN and uses manually crafted software metadata descriptions in CMDI (no harvesting) with rich information for scholars. The information is largely outdated however.
- A LaMachine Portal (labirinto) was developed as a solution to provide a portal page for any LaMachine installation/deployment, automatically harvesting the tools available within. It uses CodeMeta which is more generic but less specific for scholars.
- The CLARIN Switchboard is developed by CLARIN-ERIC and gives users the option to select tools from a wider CLARIN ecosystem, based on the data they upload. It is largely limited to singular data (single files).

Use cases

n/a

Deliverables

1. **Document:** Software Metadata Requirements
2. **Software + documentation:** codemetapy
3. **Software + documentation:** codemeta-harvester
4. **Software + documentation:** codemeta-server
5. **Service + documentation:** Tool Store

Planning

We identify the following primary tasks, more or less in chronological order, and add a rough indication (PM=person months) of the work we expect and who will perform it.

- Define cross-WP team for tool discovery
- Define components, standards, requirements for tool discovery - 0.5PM (68 hours)
- Define extra vocabulary for tool discovery - 0.5PM (68 hours) - entire team
  - Development status vocabulary
- Implement codemeta validation component for tool discovery, against clariah requirements - +0.5PM (+68 hours) - NDE (David de Boer)
- Write software metadata requirements - 0.5PM (68 hours) - KNAW HuC (Maarten vG and others)
- Implement harvester component for tool discovery - 1.5PM (204 hours) - KNAW HuC (Maarten vG)
  - Includes work on the underlying conversion component (codemetapy)
  - Extract software metadata from the web (service endpoints and/or webpages)
- Implement tool store for tool discovery - 2.0PM (272 hours) - KNAW HuC (Maarten vG)
  - Implement simple visualisation (portal page) for tool discovery
- Implement Ineo export/import for tool discovery - 0.5PM (68 hours) - KNAW HuC (Maarten vG)
  - This involves only support for the Ineo developers to make the link. The actual implementation for connectivity between tool discovery and ineo is on the Ineo side and requires more hours. The hours listed here are only for supporting the Ineo developers from our side.
- Implement Switchboard export for tool discovery - 0.5PM (68 hours) - Not assigned yet
- Implement CMDI export for tool discovery - 1PM (136 hours) - Not assigned yet
- Compose and publish metadata for all software/services - 3PM??? (very hard to estimate, this is a function of the number of tools, this is spread over all participating institutes)
  - Populate tool source registry

Total: 10.5PM (1428 hours)

Resources

Some comments on resource allocation:

- The "compose metadata for all software/services" subtask may be argued to be financed from already existing WP budgets rather than this one.
- Any funds still open for the WP3 MD4T project (Utrecht University) should be reallocated on behalf of this shared epic.
- Composing metadata for software/service should probably come from the existing WP budgets that maintain the software