OpenMinTeD Interoperability Specification

OpenMinTeD Interoperability Team (T5.2)

Version 1.1.0-SNAPSHOT

Contents

This document will host the interoperability specification for the OpenMinTeD platform. Currently, there is not much to see here. For the time being, we use this place to dump intermediate information that may or may not end up in the final specification.

Introduction

Requirement metadata

ID

Every requirement has an ID. We start counting from 1 and every new requirement increments the ID by 1. The ID is encoded in the requirement filename, e.g. `1.adoc`.

Concreteness

The OpenMinTeD platform aims to be open and inclusive. As such, it aims to support multiple popular technologies and standards. As popularity is changing over time and as new standards and technologies are evolving, the platform will have to evolve as well. The distinction between abstract and concrete interoperability requirements that we make here allows us to answer two questions:

- How difficult is it for a new technology or standard to be incorporated into the platform?
- How difficult is it to integrate new components based on already supported technologies and standards into the platform.

Abstract requirements are agnostic to concrete technologies and standards and help assessing compliance with them; helps answering the first question. Concrete requirements refer to specific implementation details and help answering the second question.

Requirement concreteness values

- **Abstract** - the requirement specifies a need, but does not go into details how this need must be fulfilled. The requirement may provide examples of techniques or implementations that fulfill the requirement, but does not mandate their use.
- **Concrete** - the requirement specifies a need and prescribes the use of specific techniques, standards, implementations, etc.

Strength

Requirement strength values

- **Mandatory** - compliance with a mandatory requirement is obligatory. Non-compliance with any mandatory requirement entails non-compliance with the specification as a whole.
- **Recommended** - compliance with a recommended requirement is not obligatory but strongly desired.
- **Optional** - compliance with an optional requirement is not obligatory and not strongly desired, but considered beneficial.

Status

The requirement status indicates how far it has proceeded in its lifecycle. If and which changes may be made to a requirement depends on this status.

Requirement status values

- **Draft** - the requirement is a suggestion and can be changed substantially in any respect.
- **Final** - the requirement is ready for release. Changes to a final requirement are only allowed if they do not

affect the compliance status of any product, component, format, etc. that as already been evaluated against the requirement specification. If a change would trigger a change in any compliance status, instead of changing an existing requirement, a new requirement must be created under a new ID and compliance must be evaluated against this new requirement specification in the next iteration. The previous requirement must be moved to deprecated status.

- **Deprecated** - the requirement is no longer to be used for compliance assessment. The requirement specification must not be changed. An exception are amendments adding pointers to potential new versions of the requirement and providing a rationale for the deprecation.

Category

The category of a requirement is used to anchor it in the document structure of the interoperability specification. The actual document structure is kept in a separate file to faciliate its refactoring. A requirement may be in multiple categories which must be provided as a comma-separated list.

Compliance assessment

Requirement compliance levels

- **Full** - fully compliant
- **Partial** - partially compliant. E.g. some parts of a product are compliant but not all. This is typically the case if a product is in a state of transition from a non-compliant to a compliant state.
- **No** - not compliant.
- **N/A** - not applicable. This is expected to occur mainly for concrete requirements if a certain requirement is not applicable for a certain implementation, e.g. a requirement on remote API access on a tool which does not offer a remote API. Abstract requirements should be formulated in such a way that they are always applicable.

Overall compliance levels

- **None** - not compliant at all
- **Silver** - fully compliant with all mandatory requirements
- **Gold** - fully compliant with all mandatory and recommended requirements
- **Platin** - fully compliant with all mandatory, recommended, and optional requirements

Overall compliance levels are maintained separately for abstract and concrete requirements. A product may attain a higher compliance level on abstract requirements than on concrete requirements. For example, an abstract mandatory requirement may be to provide metadata in a machine-readable format while a related concrete requirement may prescribe a certain metadata format. If a product offers metadata in a machine-readable format, it would be fully compliant with the abstract requirement, but if it does not use the metadata format prescribed by the concrete requirement, it is not compliant here. The product may thus attain **silver** compliance on the abstract level, but **no** overall compliance level on the concrete level.

Compliance profile

A compliance profile is a machine readable file that describes the compliance of a product, component, format, etc. with the interoperability requirements.

NOTE	The format and schema of this file is still to be determined. YAML may be a good format candidate.

Files and IDs

Requirements are maintained in the folder `req`, one requirement per file, and files are numbered starting at 1. The number corresponds to the ID of the requirement.

Template

This is the template we presently use for recording the requirements. It contains the requirement metadata as described above and a description. Links to other requirements can be embedded in the description.

```
== Here goes the requirement title

[%hardbreaks]
[small]#*_Concreteness:_* __abstract|concrete__#
[small]#*_Strength:_*     __mandatory|recommended|optional__#
[small]#*_Category:_*     __WG1__,__WG2__,__WG3__,__WG4__#
[small]#*_Status:_*       __draft|final|deprecated__#

This is where the description of the requirement goes. If you wish to reference another requirement from
here, you can do it like link:{include-dir}req/1.adoc[this].

// Below is an example of how a compliance evaluation table could look. This is presently optional
// and may be moved to a more structured/principled format later maintained in separate files.
[cols="2,1,1,4,1"]
|====
|Product|Version|Compliant|Justification|Status

| Alvis
|
| Unknown
| Unknown
| Draft

| ARGO
| 0.5
| Unknown
| Unknown
| Draft

| DKPro Core
| 1.8.0
| Unknown
| Unknown
| Draft

| GATE
| 8.2
| Unknown
| Unknown
| Draft

| ILSP
|
| Unknown
| Unknown
| Draft
|====
src/main/asciidoc/openminted-interoperability-spec/req/TEMPLATE.adoc
```

And this is how the requirement is rendered in this document (except the heading which doesn't work in this particular example).

== Here goes the requirement title

Concreteness: *abstract|concrete*
Strength: *mandatory|recommended|optional*
Category: *WG1,WG2,WG3,WG4*
Status: *draft|final|deprecated*

This is where the description of the requirement goes. If you wish to reference another requirement from here, you can do it like this.

Product	Version	Compliant	Justification	Status
Alvis		Unknown	Unknown	Draft
ARGO	0.5	Unknown	Unknown	Draft
DKPro Core	1.8.0	Unknown	Unknown	Draft
GATE	8.2	Unknown	Unknown	Draft
ILSP		Unknown	Unknown	Draft

src/main/asciidoc/openminted-interoperability-spec/req/TEMPLATE.adoc

Overviews

By WG

WG1 (42)

ID	Requirement	Concreteness	Status	Strength	WG's
4	URL to actual content must be discoverable	abstract	final	mandatory	WG1, WG2 (27), WG3 (25)
13	Citation information for component should be included in the metadata	abstract	draft	recommended	WG1, WG4 (45)
33	Licensing information must be included in the metadata	abstract	final	mandatory	WG1, WG3 (25)
34	Licensing information should be expressed in a machine-readable form	abstract	final	recommended	WG1, WG3 (25)
36	Classification metadata should be included, where applicable, in the metadata record of the resource	abstract	final	recommended	WG1, WG2 (27)
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	abstract	final	recommended	WG1
38	Access mode of resources must be included in the metadata	abstract	final	mandatory	WG1, WG2 (27), WG4 (45)
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	abstract	final	mandatory	WG1
41	Content resources must include metadata on their language(s)	abstract	final	mandatory	WG1, WG2 (27)
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	abstract	final	mandatory	WG1, WG4 (45)
44	Statistical metadata that allow monitoring of resource versions may accompany resources	abstract	final	optional	WG1, WG2 (27)
45	S/W (tools, web services, workflows) must indicate format of their output	abstract	final	mandatory	WG1, WG4 (45)
47	Information on funding of resources may be included in the metadata	abstract	final	optional	WG1, WG2 (27), WG3 (25), WG4 (45)
50	Documentation references should be versioned	abstract	final	recommended	WG1, WG2 (27), WG3 (25), WG4 (45)
74	Machine-readable metadata for UIMA components	concrete	draft	mandatory	WG1, WG4 (45)

ID	Requirement	Concreteness	Status	Strength	WG's
75	Embedding UIMA component metadata into the source code	concrete	draft	mandatory	WG1, WG4 (45)
76	Separating UIMA metadata from the component	concrete	draft	mandatory	WG1, WG4 (45)
78	Specifying input and output types of UIMA components	concrete	draft	mandatory	WG1, WG4 (45)
79	Documentation of UIMA components	concrete	draft	mandatory	WG1, WG4 (45)
81	Embedding GATE component metadata into the source code	concrete	draft	mandatory	WG1, WG4 (45)
83	Documentation of GATE components	concrete	draft	mandatory	WG1, WG4 (45)
84	Separating GATE metadata from the component	concrete	draft	mandatory	WG1, WG4 (45)
87	Embedding language capability in UIMA component metadata	concrete	draft	mandatory	WG1, WG4 (45)
88	Embedding output format in UIMA component metadata	concrete	draft	mandatory	WG1, WG4 (45)
89	Version documentation in parallel with component/resource	concrete	draft	recommended	WG1, WG2 (27), WG3 (25), WG4 (45)
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	concrete	draft	mandatory	WG1, WG4 (45)
91	Encoding citable publications (for scholarly attribution) in resource metadata records	concrete	draft	recommended	WG1, WG2 (27), WG4 (45)
92	Including license text in resource packages	concrete	draft	recommended	WG1, WG3 (25), WG4 (45)
96	Unique identifiers and versions for components using Maven	concrete	draft	mandatory	WG1, WG4 (45)
97	Declaring scaleout capability in UIMA	concrete	draft	mandatory	WG1, WG4 (45)
99	Encoding in the metadata a direct access link for content resources	concrete	draft	mandatory	WG1
100	Providing access to content resources (sharing/exposing and transferring)	concrete	draft	mandatory	WG1
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	concrete	draft	recommended	WG1, WG2 (27), WG4 (45)

ID	Requirement	Concreteness	Status	Strength	WG's
102	Adding version information in the metadata descriptions of all resources	concrete	draft	mandatory	WG1, WG2 (27), WG3 (25), WG4 (45)
103	Specifying access mode of resources and encoding it in the metadata descriptions	concrete	draft	mandatory	WG1, WG2 (27), WG4 (45)
104	Encoding funding information in the metadata descriptions of all resources	concrete	draft	recommended	WG1, WG2 (27), WG4 (45)
105	Encoding of format in the metadata description of content resources	concrete	draft	mandatory	WG1
106	Encoding licensing terms in the metadata description of the resource	concrete	draft	mandatory	WG1, WG3 (25)
107	Encoding metadata on domain/subject/ classification for all resources when applicable	concrete	draft	recommended	WG1, WG2 (27)
108	Encoding language information in the metadata of content resources	concrete	draft	mandatory	WG1, WG2 (27)
109	Encoding statistical information in the content resources	concrete	draft	mandatory	WG1, WG2 (27)
110	Assigning a unique persistent identifier for all resources	concrete	draft	mandatory	WG1, WG2 (27), WG3 (25)

WG2 (27)

ID	Requirement	Concreteness	Status	Strength	WG's
4	URL to actual content must be discoverable	abstract	final	mandatory	WG1 (42), WG2, WG3 (25)
10	Components should specify the types of the annotations that they input and output	abstract	draft	mandatory	WG4 (45), WG2
36	Classification metadata should be included, where applicable, in the metadata record of the resource	abstract	final	recommended	WG1 (42), WG2
38	Access mode of resources must be included in the metadata	abstract	final	mandatory	WG1 (42), WG2, WG4 (45)

ID	Requirement	Concreteness	Status	Strength	WG's
41	Content resources must include metadata on their language(s)	abstract	final	mandatory	WG1 (42), WG2
44	Statistical metadata that allow monitoring of resource versions may accompany resources	abstract	final	optional	WG1 (42), WG2
47	Information on funding of resources may be included in the metadata	abstract	final	optional	WG1 (42), WG2, WG3 (25), WG4 (45)
50	Documentation references should be versioned	abstract	final	recommended	WG1 (42), WG2, WG3 (25), WG4 (45)
67	Knowledge Resource Element Id	abstract	final	recommended	WG2
68	Data Category Linking Vocabulary	abstract	final	recommended	WG2
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	abstract	final	recommended	WG2
70	All KR content elements need to be added as text annotations within a TDM workflow.	abstract	final	mandatory	WG2
71	The KR should be ingestible through a URI	abstract	final	recommended	WG2
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	abstract	final	recommended	WG2
89	Version documentation in parallel with component/resource	concrete	draft	recommended	WG1 (42), WG2, WG3 (25), WG4 (45)
91	Encoding citable publications (for scholarly attribution) in resource metadata records	concrete	draft	recommended	WG1 (42), WG2, WG4 (45)
93	Provide identifiers for knowledge resource elements	concrete	draft	recommended	WG2
94	Data Category Linking Vocabulary	concrete	draft	recommended	WG2
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	concrete	draft	recommended	WG2

ID	Requirement	Concreteness	Status	Strength	WG's
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	concrete	draft	recommended	WG1 (42), WG2, WG 4 (45)
102	Adding version information in the metadata descriptions of all resources	concrete	draft	mandatory	WG1 (42), WG2, WG 3 (25) , WG4 (45)
103	Specifying access mode of resources and encoding it in the metadata descriptions	concrete	draft	mandatory	WG1 (42), WG2, WG 4 (45)
104	Encoding funding information in the metadata descriptions of all resources	concrete	draft	recommended	WG1 (42), WG2, WG 4 (45)
107	Encoding metadata on domain/subject/ classification for all resources when applicable	concrete	draft	recommended	WG1 (42), WG2
108	Encoding language information in the metadata of content resources	concrete	draft	mandatory	WG1 (42), WG2
109	Encoding statistical information in the content resources	concrete	draft	mandatory	WG1 (42), WG2
110	Assigning a unique persistent identifier for all resources	concrete	draft	mandatory	WG1 (42), WG2, WG 3 (25)

WG3 (25)

ID	Requirement	Concreteness	Status	Strength	WG's
4	URL to actual content must be discoverable	abstract	final	mandatory	WG1 (42), WG2 (27), WG3
33	Licensing information must be included in the metadata	abstract	final	mandatory	WG1 (42), WG3
34	Licensing information should be expressed in a machine-readable form	abstract	final	recommended	WG1 (42), WG3

ID	Requirement	Concreteness	Status	Strength	WG's
47	Information on funding of resources may be included in the metadata	abstract	final	optional	WG1 (42), WG2 (27), WG3, WG4 (45)
50	Documentation references should be versioned	abstract	final	recommended	WG1 (42), WG2 (27), WG3, WG4 (45)
51	License should be attached	abstract	draft	recommended	WG3
53	Licensor must be entitled to grant license	abstract	draft	recommended	WG3
54	Licensees should remain with a copy of the license	abstract	draft	recommended	WG3
55	Standard licenses should be used	abstract	draft	recommended	WG3
56	License should be machine readable	abstract	draft	recommended	WG3
57	License should be understandable by non-lawyers	abstract	draft	recommended	WG3
58	TDM must be explicitly allowed	abstract	draft	recommended	WG3
59	Right for (temporary) reproduction must be granted	abstract	draft	recommended	WG3
60	Boundary for derivative work must be clearly defined	abstract	draft	recommended	WG3
61	No restrictions on TDM results which are not derived works	abstract	draft	recommended	WG3
62	World-wide and irrevocable license grant	abstract	draft	recommended	WG3
63	License must qualify for Open Access rights	abstract	draft	recommended	WG3
64	License must qualify for Open Access uses	abstract	draft	recommended	WG3
65	License must qualify for Open Access must not restrict use in any way	abstract	draft	recommended	WG3
66	License must qualify for Open Access may include attribution requirements	abstract	draft	recommended	WG3

ID	Requirement	Concreteness	Status	Strength	WG's
89	Version documentation in parallel with component/resource	concrete	draft	recommended	WG1 (42), WG2 (27), WG3, WG4 (45)
92	Including license text in resource packages	concrete	draft	recommended	WG1 (42), WG3, WG4 (45)
102	Adding version information in the metadata descriptions of all resources	concrete	draft	mandatory	WG1 (42), WG2 (27), WG3, WG4 (45)
106	Encoding licensing terms in the metadata description of the resource	concrete	draft	mandatory	WG1 (42), WG3
110	Assigning a unique persistent identifier for all resources	concrete	draft	mandatory	WG1 (42), WG2 (27), WG3

WG4 (45)

ID	Requirement	Concreteness	Status	Strength	WG's
1	Components must be described by machine-readable metadata	abstract	final	mandatory	WG4
2	Component metadata have to be embedded into the component source code	abstract	final	mandatory	WG4
3	Component metadata must be separable from the component	abstract	final	mandatory	WG4
5	Components must detail all their environmental requirements for execution	abstract	draft	mandatory	WG4
6	Components should have a unique identifier and a version number	abstract	draft	mandatory	WG4
7	Components must have a fully qualified name that follows the Java class naming conventions	concrete	final	mandatory	WG4
8	Components must associate themselves with categories defined by the OpenMinTeD project	abstract	final	mandatory	WG4
9	Components must declare their annotation schema dependencies	abstract	final	mandatory	WG4
10	Components should specify the types of the annotations that they input and output	abstract	draft	mandatory	WG4, WG2 (27)

ID	Requirement	Concreteness	Status	Strength	WG's
11	Components must declare whether they can be scaled within a workflow	abstract	draft	mandatory	WG4
12	Components should provide documentation describing their functionality	abstract	final	recommended	WG4
13	Citation information for component should be included in the metadata	abstract	draft	recommended	WG1 (42), WG4
16	Models/resources should be useable across different component collections/platforms	abstract	final	recommended	WG4
17	Components should be stateless	concrete	final	recommended	WG4
21	Configuration and parametrizable options of the components should be identified and documented	abstract	final	recommended	WG4
26	It should be possible to determine the source of an annotation/assigned category	abstract	final	recommended	WG4
27	Components should handle failures gracefully	abstract	final	recommended	WG4
28	Processing components should be downloadable	abstract	final	recommended	WG4
38	Access mode of resources must be included in the metadata	abstract	final	mandatory	WG1 (42), WG2 (27), WG4
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	abstract	final	mandatory	WG1 (42), WG4
45	S/W (tools, web services, workflows) must indicate format of their output	abstract	final	mandatory	WG1 (42), WG4
47	Information on funding of resources may be included in the metadata	abstract	final	optional	WG1 (42), WG2 (27), WG3 (25), WG4
50	Documentation references should be versioned	abstract	final	recommended	WG1 (42), WG2 (27), WG3 (25), WG4
73	Stick to widely used data compression formats	concrete	draft	best practice	WG4
74	Machine-readable metadata for UIMA components	concrete	draft	mandatory	WG1 (42), WG4

ID	Requirement	Concreteness	Status	Strength	WG's
75	Embedding UIMA component metadata into the source code	concrete	draft	mandatory	WG1 (42), WG4
76	Separating UIMA metadata from the component	concrete	draft	mandatory	WG1 (42), WG4
78	Specifying input and output types of UIMA components	concrete	draft	mandatory	WG1 (42), WG4
79	Documentation of UIMA components	concrete	draft	mandatory	WG1 (42), WG4
81	Embedding GATE component metadata into the source code	concrete	draft	mandatory	WG1 (42), WG4
83	Documentation of GATE components	concrete	draft	mandatory	WG1 (42), WG4
84	Separating GATE metadata from the component	concrete	draft	mandatory	WG1 (42), WG4
87	Embedding language capability in UIMA component metadata	concrete	draft	mandatory	WG1 (42), WG4
88	Embedding output format in UIMA component metadata	concrete	draft	mandatory	WG1 (42), WG4
89	Version documentation in parallel with component/resource	concrete	draft	recommended	WG1 (42), WG2 (27), WG3 (25), WG4
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	concrete	draft	mandatory	WG1 (42), WG4
91	Encoding citable publications (for scholarly attribution) in resource metadata records	concrete	draft	recommended	WG1 (42), WG2 (27), WG4
92	Including license text in resource packages	concrete	draft	recommended	WG1 (42), WG3 (25), WG4

ID	Requirement	Concreteness	Status	Strength	WG's
96	Unique identifiers and versions for components using Maven	concrete	draft	mandatory	WG1 (42), WG4
97	Declaring scaleout capability in UIMA	concrete	draft	mandatory	WG1 (42), WG4
98	Publishing components via software repositories (Maven, Docker)	concrete	draft	mandatory	WG4
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	concrete	draft	recommended	WG1 (42), WG2 (27), WG4
102	Adding version information in the metadata descriptions of all resources	concrete	draft	mandatory	WG1 (42), WG2 (27), WG3 (25), WG4
103	Specifying access mode of resources and encoding it in the metadata descriptions	concrete	draft	mandatory	WG1 (42), WG2 (27), WG4
104	Encoding funding information in the metadata descriptions of all resources	concrete	draft	recommended	WG1 (42), WG2 (27), WG4

By Status

deprecated (25)

ID	Requirement	Concreteness	Strength	WG's
14	Components must maintain License information	abstract	mandatory	WG4 (45)
15	Human readable information should be provided by each resource	abstract	recommended	WG1 (42), WG4 (45)
18	Workflows should be described using an uniform language	abstract	recommended	WG4 (45)
19	Components that use external knowledge resources should delegate access to a resource adapter instead of handling it themselves	abstract	optional	WG2 (27), WG4 (45)
20	Workflow engines should not require to see data	concrete	recommended	WG2 (27), WG4 (45)
22	The Workflow Engine Should Permit Saving Experimental Conditions in a Workflow	abstract	recommended	WG1 (42), WG4 (45)
23	The Workflow Engine should permit Licence Aggregation in Workflows	abstract	recommended	WG3 (25), WG4 (45)
24	Using/treating workflows as components	abstract	mandatory	WG4 (45)
25	Incorporation of multiple resources in parallel	abstract	recommended	WG4 (45)
29	The actual content of all content resources must be discoverable	abstract	mandatory	WG1 (42), WG2 (27), WG3 (25)
30	Metrics for the confidence level of the TDM operation should be included in the metadata	abstract	optional	WG1 (42), WG4 (45)
31	Metrics for the performance of the TDM operation should be included in the metadata	abstract	optional	WG1 (42), WG4 (45)
32	Version must be included in the metadata description for all resources	abstract	mandatory	WG1 (42), WG2 (27), WG3 (25), WG4 (45)

ID	Requirement	Concreteness	Strength	WG's
35	All resources must include a unique persistent identifier	abstract	mandatory	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
40	Component metadata must include standardised categories/tags that make them easy to discover	abstract	mandatory	WG1 (42), WG4 (45)
42	The metadata can include the information on which projects/workflows involve the resource	abstract	optional	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
46	Output resources of web services/workflows must be accompanied by provenance metadata	abstract	mandatory	WG1 (42), WG4 (45)
48	All resource metadata records must include a reference to the metadata schema used for their description	abstract	mandatory	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
49	Metadata of tools should contain information about the models available for them	abstract	recommended	WG1 (42), WG4 (45)
52	License information must be in metadata	abstract	recommended	WG1 (42), WG3 (25)
77	Unique identifiers and a versions for UIMA components	concrete	mandatory	WG1 (42), WG4 (45)
80	Common elements to represent/describe an executable workflow	abstract	concrete	recommended
WG4 (45)	82	Unique identifiers and a versions for GATE components	concrete	mandatory

ID	Requirement	Concreteness	Strength	WG's
WG1 (42), WG4 (45)	85	Unique identifier and version for components in the OMTD-SHARE schema	concrete	mandatory
WG1 (42), WG4 (45)	86	Attaching format properties to the description of the inputs and outputs that use file	concrete	recommended

draft (53)

ID	Requirement	Concreteness	Strength	WG's
5	Components must detail all their environmental requirements for execution	abstract	mandatory	WG4 (45)
6	Components should have a unique identifier and a version number	abstract	mandatory	WG4 (45)
10	Components should specify the types of the annotations that they input and output	abstract	mandatory	WG4 (45), WG2 (27)
11	Components must declare whether they can be scaled within a workflow	abstract	mandatory	WG4 (45)
13	Citation information for component should be included in the metadata	abstract	recommended	WG1 (42), WG4 (45)
51	License should be attached	abstract	recommended	WG3 (25)
53	Licensor must be entitled to grant license	abstract	recommended	WG3 (25)
54	Licensees should remain with a copy of the license	abstract	recommended	WG3 (25)
55	Standard licenses should be used	abstract	recommended	WG3 (25)

ID	Requirement	Concreteness	Strength	WG's
56	License should be machine readable	abstract	recommended	WG3 (25)
57	License should be understandable by non-lawyers	abstract	recommended	WG3 (25)
58	TDM must be explicitly allowed	abstract	recommended	WG3 (25)
59	Right for (temporary) reproduction must be granted	abstract	recommended	WG3 (25)
60	Boundary for derivative work must be clearly defined	abstract	recommended	WG3 (25)
61	No restrictions on TDM results which are not derived works	abstract	recommended	WG3 (25)
62	World-wide and irrevocable license grant	abstract	recommended	WG3 (25)
63	License must qualify for Open Access rights	abstract	recommended	WG3 (25)
64	License must qualify for Open Access uses	abstract	recommended	WG3 (25)
65	License must qualify for Open Access must not restrict use in any way	abstract	recommended	WG3 (25)
66	License must qualify for Open Access may include attribution requirements	abstract	recommended	WG3 (25)
73	Stick to widely used data compression formats	concrete	best practice	WG4 (45)
74	Machine-readable metadata for UIMA components	concrete	mandatory	WG1 (42), WG4 (45)
75	Embedding UIMA component metadata into the source code	concrete	mandatory	WG1 (42), WG4 (45)
76	Separating UIMA metadata from the component	concrete	mandatory	WG1 (42), WG4 (45)
78	Specifying input and output types of UIMA components	concrete	mandatory	WG1 (42), WG4 (45)
79	Documentation of UIMA components	concrete	mandatory	WG1 (42), WG4 (45)
81	Embedding GATE component metadata into the source code	concrete	mandatory	WG1 (42), WG4 (45)
83	Documentation of GATE components	concrete	mandatory	WG1 (42), WG4 (45)

ID	Requirement	Concreteness	Strength	WG's
84	Separating GATE metadata from the component	concrete	mandatory	WG1 (42), WG4 (45)
87	Embedding language capability in UIMA component metadata	concrete	mandatory	WG1 (42), WG4 (45)
88	Embedding output format in UIMA component metadata	concrete	mandatory	WG1 (42), WG4 (45)
89	Version documentation in parallel with component/resource	concrete	recommended	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	concrete	mandatory	WG1 (42), WG4 (45)
91	Encoding citable publications (for scholarly attribution) in resource metadata records	concrete	recommended	WG1 (42), WG2 (27), WG4 (45)
92	Including license text in resource packages	concrete	recommended	WG1 (42), WG3 (25), WG4 (45)
93	Provide identifiers for knowledge resource elements	concrete	recommended	WG2 (27)
94	Data Category Linking Vocabulary	concrete	recommended	WG2 (27)
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	concrete	recommended	WG2 (27)
96	Unique identifiers and versions for components using Maven	concrete	mandatory	WG1 (42), WG4 (45)
97	Declaring scaleout capability in UIMA	concrete	mandatory	WG1 (42), WG4 (45)
98	Publishing components via software repositories (Maven, Docker)	concrete	mandatory	WG4 (45)
99	Encoding in the metadata a direct access link for content resources	concrete	mandatory	WG1 (42)
100	Providing access to content resources (sharing/exposing and transferring)	concrete	mandatory	WG1 (42)

ID	Requirement	Concreteness	Strength	WG's
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	concrete	recommended	WG1 (42), WG2 (27), WG4 (45)
102	Adding version information in the metadata descriptions of all resources	concrete	mandatory	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
103	Specifying access mode of resources and encoding it in the metadata descriptions	concrete	mandatory	WG1 (42), WG2 (27), WG4 (45)
104	Encoding funding information in the metadata descriptions of all resources	concrete	recommended	WG1 (42), WG2 (27), WG4 (45)
105	Encoding of format in the metadata description of content resources	concrete	mandatory	WG1 (42)
106	Encoding licensing terms in the metadata description of the resource	concrete	mandatory	WG1 (42), WG3 (25)
107	Encoding metadata on domain/subject/ classification for all resources when applicable	concrete	recommended	WG1 (42), WG2 (27)
108	Encoding language information in the metadata of content resources	concrete	mandatory	WG1 (42), WG2 (27)
109	Encoding statistical information in the content resources	concrete	mandatory	WG1 (42), WG2 (27)
110	Assigning a unique persistent identifier for all resources	concrete	mandatory	WG1 (42), WG2 (27), WG3 (25)

final (32)

ID	Requirement	Concreteness	Strength	WG's
1	Components must be described by machine-readable metadata	abstract	mandatory	WG4 (45)

ID	Requirement	Concreteness	Strength	WG's
2	Component metadata have to be embedded into the component source code	abstract	mandatory	WG4 (45)
3	Component metadata must be separable from the component	abstract	mandatory	WG4 (45)
4	URL to actual content must be discoverable	abstract	mandatory	WG1 (42), WG2 (27), WG3 (25)
7	Components must have a fully qualified name that follows the Java class naming conventions	concrete	mandatory	WG4 (45)
8	Components must associate themselves with categories defined by the OpenMinTeD project	abstract	mandatory	WG4 (45)
9	Components must declare their annotation schema dependencies	abstract	mandatory	WG4 (45)
12	Components should provide documentation describing their functionality	abstract	recommended	WG4 (45)
16	Models/resources should be useable across different component collections/platforms	abstract	recommended	WG4 (45)
17	Components should be stateless	concrete	recommended	WG4 (45)
21	Configuration and parametrizable options of the components should be identified and documented	abstract	recommended	WG4 (45)
26	It should be possible to determine the source of an annotation/assigned category	abstract	recommended	WG4 (45)
27	Components should handle failures gracefully	abstract	recommended	WG4 (45)
28	Processing components should be downloadable	abstract	recommended	WG4 (45)
33	Licensing information must be included in the metadata	abstract	mandatory	WG1 (42), WG3 (25)
34	Licensing information should be expressed in a machine-readable form	abstract	recommended	WG1 (42), WG3 (25)
36	Classification metadata should be included, where applicable, in the metadata record of the resource	abstract	recommended	WG1 (42), WG2 (27)
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	abstract	recommended	WG1 (42)
38	Access mode of resources must be included in the metadata	abstract	mandatory	WG1 (42), WG2 (27), WG4 (45)

ID	Requirement	Concreteness	Strength	WG's
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	abstract	mandatory	WG1 (42)
41	Content resources must include metadata on their language(s)	abstract	mandatory	WG1 (42), WG2 (27)
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	abstract	mandatory	WG1 (42), WG4 (45)
44	Statistical metadata that allow monitoring of resource versions may accompany resources	abstract	optional	WG1 (42), WG2 (27)
45	S/W (tools, web services, workflows) must indicate format of their output	abstract	mandatory	WG1 (42), WG4 (45)
47	Information on funding of resources may be included in the metadata	abstract	optional	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
50	Documentation references should be versioned	abstract	recommended	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
67	Knowledge Resource Element Id	abstract	recommended	WG2 (27)
68	Data Category Linking Vocabulary	abstract	recommended	WG2 (27)
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	abstract	recommended	WG2 (27)
70	All KR content elements need to be added as text annotations within a TDM workflow.	abstract	mandatory	WG2 (27)
71	The KR should be ingestible through a URI	abstract	recommended	WG2 (27)
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	abstract	recommended	WG2 (27)

By Strength

[[STR-best practice]] === best practice (1)

ID	Requirement	Concreteness	Status	WG's
73	Stick to widely used data compression formats	concrete	draft	WG4 (45)

mandatory (52)

ID	Requirement	Concreteness	Status	WG's
1	Components must be described by machine-readable metadata	abstract	final	WG4 (45)
2	Component metadata have to be embedded into the component source code	abstract	final	WG4 (45)
3	Component metadata must be separable from the component	abstract	final	WG4 (45)
4	URL to actual content must be discoverable	abstract	final	WG1 (42), WG2 (27), WG3 (25)
5	Components must detail all their environmental requirements for execution	abstract	draft	WG4 (45)
6	Components should have a unique identifier and a version number	abstract	draft	WG4 (45)
7	Components must have a fully qualified name that follows the Java class naming conventions	concrete	final	WG4 (45)
8	Components must associate themselves with categories defined by the OpenMinTeD project	abstract	final	WG4 (45)
9	Components must declare their annotation schema dependencies	abstract	final	WG4 (45)
10	Components should specify the types of the annotations that they input and output	abstract	draft	WG4 (45), WG2 (27)
11	Components must declare whether they can be scaled within a workflow	abstract	draft	WG4 (45)
14	Components must maintain License information	abstract	deprecated	WG4 (45)
24	Using/treating workflows as components	abstract	deprecated	WG4 (45)

ID	Requirement	Concreteness	Status	WG's
29	The actual content of all content resources must be discoverable	abstract	deprecated	WG1 (42), WG2 (27), WG3 (25)
32	Version must be included in the metadata description for all resources	abstract	deprecated	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
33	Licensing information must be included in the metadata	abstract	final	WG1 (42), WG3 (25)
35	All resources must include a unique persistent identifier	abstract	deprecated	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
38	Access mode of resources must be included in the metadata	abstract	final	WG1 (42), WG2 (27), WG4 (45)
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	abstract	final	WG1 (42)
40	Component metadata must include standardised categories/tags that make them easy to discover	abstract	deprecated	WG1 (42), WG4 (45)
41	Content resources must include metadata on their language(s)	abstract	final	WG1 (42), WG2 (27)
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	abstract	final	WG1 (42), WG4 (45)
45	S/W (tools, web services, workflows) must indicate format of their output	abstract	final	WG1 (42), WG4 (45)
46	Output resources of web services/workflows must be accompanied by provenance metadata	abstract	deprecated	WG1 (42), WG4 (45)
48	All resource metadata records must include a reference to the metadata schema used for their description	abstract	deprecated	WG1 (42), WG2 (27), WG3 (25), WG4 (45)

ID	Requirement	Concreteness	Status	WG's
70	All KR content elements need to be added as text annotations within a TDM workflow.	abstract	final	WG2 (27)
74	Machine-readable metadata for UIMA components	concrete	draft	WG1 (42), WG4 (45)
75	Embedding UIMA component metadata into the source code	concrete	draft	WG1 (42), WG4 (45)
76	Separating UIMA metadata from the component	concrete	draft	WG1 (42), WG4 (45)
77	Unique identifiers and a versions for UIMA components	concrete	deprecated	WG1 (42), WG4 (45)
78	Specifying input and output types of UIMA components	concrete	draft	WG1 (42), WG4 (45)
79	Documentation of UIMA components	concrete	draft	WG1 (42), WG4 (45)
81	Embedding GATE component metadata into the source code	concrete	draft	WG1 (42), WG4 (45)
82	Unique identifiers and a versions for GATE components	concrete	deprecated	WG1 (42), WG4 (45)
83	Documentation of GATE components	concrete	draft	WG1 (42), WG4 (45)
84	Separating GATE metadata from the component	concrete	draft	WG1 (42), WG4 (45)
85	Unique identifier and version for components in the OMTD-SHARE schema	concrete	deprecated	WG1 (42), WG4 (45)
87	Embedding language capability in UIMA component metadata	concrete	draft	WG1 (42), WG4 (45)
88	Embedding output format in UIMA component metadata	concrete	draft	WG1 (42), WG4 (45)
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	concrete	draft	WG1 (42), WG4 (45)
96	Unique identifiers and versions for components using Maven	concrete	draft	WG1 (42), WG4 (45)

ID	Requirement	Concreteness	Status	WG's
97	Declaring scaleout capability in UIMA	concrete	draft	WG1 (42), WG4 (45)
98	Publishing components via software repositories (Maven, Docker)	concrete	draft	WG4 (45)
99	Encoding in the metadata a direct access link for content resources	concrete	draft	WG1 (42)
100	Providing access to content resources (sharing/exposing and transferring)	concrete	draft	WG1 (42)
102	Adding version information in the metadata descriptions of all resources	concrete	draft	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
103	Specifying access mode of resources and encoding it in the metadata descriptions	concrete	draft	WG1 (42), WG2 (27), WG4 (45)
105	Encoding of format in the metadata description of content resources	concrete	draft	WG1 (42)
106	Encoding licensing terms in the metadata description of the resource	concrete	draft	WG1 (42), WG3 (25)
108	Encoding language information in the metadata of content resources	concrete	draft	WG1 (42), WG2 (27)
109	Encoding statistical information in the content resources	concrete	draft	WG1 (42), WG2 (27)
110	Assigning a unique persistent identifier for all resources	concrete	draft	WG1 (42), WG2 (27), WG3 (25)

optional (6)

ID	Requirement	Concreteness	Status	WG's
19	Components that use external knowledge resources should delegate access to a resource adapter instead of handling it themselves	abstract	deprecated	WG2 (27), WG4 (45)
30	Metrics for the confidence level of the TDM operation should be included in the metadata	abstract	deprecated	WG1 (42), WG4 (45)

ID	Requirement	Concreteness	Status	WG's
31	Metrics for the performance of the TDM operation should be included in the metadata	abstract	deprecated	WG1 (42), WG4 (45)
42	The metadata can include the information on which projects/workflows involve the resource	abstract	deprecated	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
44	Statistical metadata that allow monitoring of resource versions may accompany resources	abstract	final	WG1 (42), WG2 (27)
47	Information on funding of resources may be included in the metadata	abstract	final	WG1 (42), WG2 (27), WG3 (25), WG4 (45)

recommended (51)

ID	Requirement	Concreteness	Status	WG's
12	Components should provide documentation describing their functionality	abstract	final	WG4 (45)
13	Citation information for component should be included in the metadata	abstract	draft	WG1 (42), WG4 (45)
15	Human readable information should be provided by each resource	abstract	deprecated	WG1 (42), WG4 (45)
16	Models/resources should be useable across different component collections/platforms	abstract	final	WG4 (45)
17	Components should be stateless	concrete	final	WG4 (45)
18	Workflows should be described using an uniform language	abstract	deprecated	WG4 (45)
20	Workflow engines should not require to see data	concrete	deprecated	WG2 (27), WG4 (45)
21	Configuration and parametrizable options of the components should be identified and documented	abstract	final	WG4 (45)
22	The Workflow Engine Should Permit Saving Experimental Conditions in a Workflow	abstract	deprecated	WG1 (42), WG4 (45)

ID	Requirement	Concreteness	Status	WG's
23	The Workflow Engine should permit Licence Aggregation in Workflows	abstract	deprecated	WG3 (25), WG4 (45)
25	Incorporation of multiple resources in parallel	abstract	deprecated	WG4 (45)
26	It should be possible to determine the source of an annotation/assigned category	abstract	final	WG4 (45)
27	Components should handle failures gracefully	abstract	final	WG4 (45)
28	Processing components should be downloadable	abstract	final	WG4 (45)
34	Licensing information should be expressed in a machine-readable form	abstract	final	WG1 (42), WG3 (25)
36	Classification metadata should be included, where applicable, in the metadata record of the resource	abstract	final	WG1 (42), WG2 (27)
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	abstract	final	WG1 (42)
49	Metadata of tools should contain information about the models available for them	abstract	deprecated	WG1 (42), WG4 (45)
50	Documentation references should be versioned	abstract	final	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
51	License should be attached	abstract	draft	WG3 (25)
52	License information must be in metadata	abstract	deprecated	WG1 (42), WG3 (25)
53	Licensor must be entitled to grant license	abstract	draft	WG3 (25)
54	Licensees should remain with a copy of the license	abstract	draft	WG3 (25)
55	Standard licenses should be used	abstract	draft	WG3 (25)
56	License should be machine readable	abstract	draft	WG3 (25)
57	License should be understandable by non-lawyers	abstract	draft	WG3 (25)
58	TDM must be explicitly allowed	abstract	draft	WG3 (25)

ID	Requirement	Concreteness	Status	WG's
59	Right for (temporary) reproduction must be granted	abstract	draft	WG3 (25)
60	Boundary for derivative work must be clearly defined	abstract	draft	WG3 (25)
61	No restrictions on TDM results which are not derived works	abstract	draft	WG3 (25)
62	World-wide and irrevocable license grant	abstract	draft	WG3 (25)
63	License must qualify for Open Access rights	abstract	draft	WG3 (25)
64	License must qualify for Open Access uses	abstract	draft	WG3 (25)
65	License must qualify for Open Access must not restrict use in any way	abstract	draft	WG3 (25)
66	License must qualify for Open Access may include attribution requirements	abstract	draft	WG3 (25)
67	Knowledge Resource Element Id	abstract	final	WG2 (27)
68	Data Category Linking Vocabulary	abstract	final	WG2 (27)
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	abstract	final	WG2 (27)
71	The KR should be ingestible through a URI	abstract	final	WG2 (27)
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	abstract	final	WG2 (27)
80	Common elements to represent/describe an executable workflow	abstract	concrete	deprecated
WG4 (45)	86	Attaching format properties to the description of the inputs and outputs that use file	concrete	deprecated

ID	Requirement	Concreteness	Status	WG's
WG1 (42), WG4 (45)	89	Version documentation in parallel with component/resource	concrete	draft
WG1 (42), WG2 (27), WG3 (25), WG4 (45)	91	Encoding citable publications (for scholarly attribution) in resource metadata records	concrete	draft
WG1 (42), WG2 (27), WG4 (45)	92	Including license text in resource packages	concrete	draft
WG1 (42), WG3 (25), WG4 (45)	93	Provide identifiers for knowledge resource elements	concrete	draft
WG2 (27)	94	Data Category Linking Vocabulary	concrete	draft

ID	Requirement	Concreteness	Status	WG's
WG2 (27)	95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	concrete	draft
WG2 (27)	101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	concrete	draft
WG1 (42), WG2 (27), WG4 (45)	104	Encoding funding information in the metadata descriptions of all resources	concrete	draft

ID	Requirement	Concreteness	Status	WG's
WG1 (42), WG2 (27), WG4 (45)	107	Encoding metadata on domain/subject/classification for all resources when applicable	concrete	draft

By Concreteness

abstract (69)

ID	Requirement	Status	Strength	WG's
1	Components must be described by machine-readable metadata	final	mandatory	WG4 (45)
2	Component metadata have to be embedded into the component source code	final	mandatory	WG4 (45)
3	Component metadata must be separable from the component	final	mandatory	WG4 (45)
4	URL to actual content must be discoverable	final	mandatory	WG1 (42), WG2 (27), WG3 (25)
5	Components must detail all their environmental requirements for execution	draft	mandatory	WG4 (45)
6	Components should have a unique identifier and a version number	draft	mandatory	WG4 (45)
8	Components must associate themselves with categories defined by the OpenMinTeD project	final	mandatory	WG4 (45)
9	Components must declare their annotation schema dependencies	final	mandatory	WG4 (45)
10	Components should specify the types of the annotations that they input and output	draft	mandatory	WG4 (45), WG2 (27)
11	Components must declare whether they can be scaled within a workflow	draft	mandatory	WG4 (45)
12	Components should provide documentation describing their functionality	final	recommended	WG4 (45)
13	Citation information for component should be included in the metadata	draft	recommended	WG1 (42), WG4 (45)
14	Components must maintain License information	deprecated	mandatory	WG4 (45)
15	Human readable information should be provided by each resource	deprecated	recommended	WG1 (42), WG4 (45)
16	Models/resources should be useable across different component collections/platforms	final	recommended	WG4 (45)
18	Workflows should be described using an uniform language	deprecated	recommended	WG4 (45)

ID	Requirement	Status	Strength	WG's
19	Components that use external knowledge resources should delegate access to a resource adapter instead of handling it themselves	deprecated	optional	WG2 (27), WG4 (45)
21	Configuration and parametrizable options of the components should be identified and documented	final	recommended	WG4 (45)
22	The Workflow Engine Should Permit Saving Experimental Conditions in a Workflow	deprecated	recommended	WG1 (42), WG4 (45)
23	The Workflow Engine should permit Licence Aggregation in Workflows	deprecated	recommended	WG3 (25), WG4 (45)
24	Using/treating workflows as components	deprecated	mandatory	WG4 (45)
25	Incorporation of multiple resources in parallel	deprecated	recommended	WG4 (45)
26	It should be possible to determine the source of an annotation/assigned category	final	recommended	WG4 (45)
27	Components should handle failures gracefully	final	recommended	WG4 (45)
28	Processing components should be downloadable	final	recommended	WG4 (45)
29	The actual content of all content resources must be discoverable	deprecated	mandatory	WG1 (42), WG2 (27), WG3 (25)
30	Metrics for the confidence level of the TDM operation should be included in the metadata	deprecated	optional	WG1 (42), WG4 (45)
31	Metrics for the performance of the TDM operation should be included in the metadata	deprecated	optional	WG1 (42), WG4 (45)
32	Version must be included in the metadata description for all resources	deprecated	mandatory	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
33	Licensing information must be included in the metadata	final	mandatory	WG1 (42), WG3 (25)
34	Licensing information should be expressed in a machine-readable form	final	recommended	WG1 (42), WG3 (25)

ID	Requirement	Status	Strength	WG's
35	All resources must include a unique persistent identifier	deprecated	mandatory	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
36	Classification metadata should be included, where applicable, in the metadata record of the resource	final	recommended	WG1 (42), WG2 (27)
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	final	recommended	WG1 (42)
38	Access mode of resources must be included in the metadata	final	mandatory	WG1 (42), WG2 (27), WG4 (45)
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	final	mandatory	WG1 (42)
40	Component metadata must include standardised categories/tags that make them easy to discover	deprecated	mandatory	WG1 (42), WG4 (45)
41	Content resources must include metadata on their language(s)	final	mandatory	WG1 (42), WG2 (27)
42	The metadata can include the information on which projects/workflows involve the resource	deprecated	optional	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	final	mandatory	WG1 (42), WG4 (45)
44	Statistical metadata that allow monitoring of resource versions may accompany resources	final	optional	WG1 (42), WG2 (27)
45	S/W (tools, web services, workflows) must indicate format of their output	final	mandatory	WG1 (42), WG4 (45)
46	Output resources of web services/workflows must be accompanied by provenance metadata	deprecated	mandatory	WG1 (42), WG4 (45)
47	Information on funding of resources may be included in the metadata	final	optional	WG1 (42), WG2 (27), WG3 (25), WG4 (45)

ID	Requirement	Status	Strength	WG's
48	All resource metadata records must include a reference to the metadata schema used for their description	deprecated	mandatory	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
49	Metadata of tools should contain information about the models available for them	deprecated	recommended	WG1 (42), WG4 (45)
50	Documentation references should be versioned	final	recommended	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
51	License should be attached	draft	recommended	WG3 (25)
52	License information must be in metadata	deprecated	recommended	WG1 (42), WG3 (25)
53	Licensor must be entitled to grant license	draft	recommended	WG3 (25)
54	Licensees should remain with a copy of the license	draft	recommended	WG3 (25)
55	Standard licenses should be used	draft	recommended	WG3 (25)
56	License should be machine readable	draft	recommended	WG3 (25)
57	License should be understandable by non-lawyers	draft	recommended	WG3 (25)
58	TDM must be explicitly allowed	draft	recommended	WG3 (25)
59	Right for (temporary) reproduction must be granted	draft	recommended	WG3 (25)
60	Boundary for derivative work must be clearly defined	draft	recommended	WG3 (25)
61	No restrictions on TDM results which are not derived works	draft	recommended	WG3 (25)
62	World-wide and irrevocable license grant	draft	recommended	WG3 (25)
63	License must qualify for Open Access rights	draft	recommended	WG3 (25)
64	License must qualify for Open Access uses	draft	recommended	WG3 (25)

ID	Requirement	Status	Strength	WG's
65	License must qualify for Open Access must not restrict use in any way	draft	recommended	WG3 (25)
66	License must qualify for Open Access may include attribution requirements	draft	recommended	WG3 (25)
67	Knowledge Resource Element Id	final	recommended	WG2 (27)
68	Data Category Linking Vocabulary	final	recommended	WG2 (27)
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	final	recommended	WG2 (27)
70	All KR content elements need to be added as text annotations within a TDM workflow.	final	mandatory	WG2 (27)
71	The KR should be ingestible through a URI	final	recommended	WG2 (27)
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	final	recommended	WG2 (27)

[[CONC-abstract|concrete]] === abstract|concrete (1)

ID	Requirement	Status	Strength	WG's
80	Common elements to represent/describe an executable workflow	deprecated	recommended	WG4 (45)

concrete (40)

ID	Requirement	Status	Strength	WG's
7	Components must have a fully qualified name that follows the Java class naming conventions	final	mandatory	WG4 (45)
17	Components should be stateless	final	recommended	WG4 (45)
20	Workflow engines should not require to see data	deprecated	recommended	WG2 (27), WG4 (45)
73	Stick to widely used data compression formats	draft	best practice	WG4 (45)
74	Machine-readable metadata for UIMA components	draft	mandatory	WG1 (42), WG4 (45)
75	Embedding UIMA component metadata into the source code	draft	mandatory	WG1 (42), WG4 (45)

ID	Requirement	Status	Strength	WG's
76	Separating UIMA metadata from the component	draft	mandatory	WG1 (42), WG4 (45)
77	Unique identifiers and a versions for UIMA components	deprecated	mandatory	WG1 (42), WG4 (45)
78	Specifying input and output types of UIMA components	draft	mandatory	WG1 (42), WG4 (45)
79	Documentation of UIMA components	draft	mandatory	WG1 (42), WG4 (45)
81	Embedding GATE component metadata into the source code	draft	mandatory	WG1 (42), WG4 (45)
82	Unique identifiers and a versions for GATE components	deprecated	mandatory	WG1 (42), WG4 (45)
83	Documentation of GATE components	draft	mandatory	WG1 (42), WG4 (45)
84	Separating GATE metadata from the component	draft	mandatory	WG1 (42), WG4 (45)
85	Unique identifier and version for components in the OMTD-SHARE schema	deprecated	mandatory	WG1 (42), WG4 (45)
86	Attaching format properties to the description of the inputs and outputs that use file	deprecated	recommended	WG1 (42), WG4 (45)
87	Embedding language capability in UIMA component metadata	draft	mandatory	WG1 (42), WG4 (45)
88	Embedding output format in UIMA component metadata	draft	mandatory	WG1 (42), WG4 (45)
89	Version documentation in parallel with component/resource	draft	recommended	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	draft	mandatory	WG1 (42), WG4 (45)

ID	Requirement	Status	Strength	WG's
91	Encoding citable publications (for scholarly attribution) in resource metadata records	draft	recommended	WG1 (42), WG2 (27), WG4 (45)
92	Including license text in resource packages	draft	recommended	WG1 (42), WG3 (25), WG4 (45)
93	Provide identifiers for knowledge resource elements	draft	recommended	WG2 (27)
94	Data Category Linking Vocabulary	draft	recommended	WG2 (27)
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	draft	recommended	WG2 (27)
96	Unique identifiers and versions for components using Maven	draft	mandatory	WG1 (42), WG4 (45)
97	Declaring scaleout capability in UIMA	draft	mandatory	WG1 (42), WG4 (45)
98	Publishing components via software repositories (Maven, Docker)	draft	mandatory	WG4 (45)
99	Encoding in the metadata a direct access link for content resources	draft	mandatory	WG1 (42)
100	Providing access to content resources (sharing/exposing and transferring)	draft	mandatory	WG1 (42)
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	draft	recommended	WG1 (42), WG2 (27), WG4 (45)
102	Adding version information in the metadata descriptions of all resources	draft	mandatory	WG1 (42), WG2 (27), WG3 (25), WG4 (45)
103	Specifying access mode of resources and encoding it in the metadata descriptions	draft	mandatory	WG1 (42), WG2 (27), WG4 (45)

ID	Requirement	Status	Strength	WG's
104	Encoding funding information in the metadata descriptions of all resources	draft	recommended	WG1 (42), WG2 (27), WG4 (45)
105	Encoding of format in the metadata description of content resources	draft	mandatory	WG1 (42)
106	Encoding licensing terms in the metadata description of the resource	draft	mandatory	WG1 (42), WG3 (25)
107	Encoding metadata on domain/subject/ classification for all resources when applicable	draft	recommended	WG1 (42), WG2 (27)
108	Encoding language information in the metadata of content resources	draft	mandatory	WG1 (42), WG2 (27)
109	Encoding statistical information in the content resources	draft	mandatory	WG1 (42), WG2 (27)
110	Assigning a unique persistent identifier for all resources	draft	mandatory	WG1 (42), WG2 (27), WG3 (25)

Compliance

By Product

Numbers exclude deprecated requirements.

ARGO (46)

Compliance	#	%
Full	8	17
No	16	35
Partial	22	48

ID	Requirement	Compliance
1	Components must be described by machine-readable metadata	Full
2	Component metadata have to be embedded into the component source code	No
3	Component metadata must be separable from the component	Partial
5	Components must detail all their environmental requirements for execution	Partial
6	Components should have a unique identifier and a version number	Partial
7	Components must have a fully qualified name that follows the Java class naming conventions	Partial
8	Components must associate themselves with categories defined by the OpenMinTeD project	Partial
9	Components must declare their annotation schema dependencies	Full
10	Components should specify the types of the annotations that they input and output	Partial
11	Components must declare whether they can be scaled within a workflow	Full
12	Components should provide documentation describing their functionality	Partial
13	Citation information for component should be included in the metadata	No
16	Models/resources should be useable across different component collections/platforms	Partial
17	Components should be stateless	Partial
21	Configuration and parametrizable options of the components should be identified and documented	Full
26	It should be possible to determine the source of an annotation/assigned category	No
27	Components should handle failures gracefully	Partial
28	Processing components should be downloadable	No
33	Licensing information must be included in the metadata	Partial
36	Classification metadata should be included, where applicable, in the metadata record of the resource	No
38	Access mode of resources must be included in the metadata	Partial

ID	Requirement	Compliance
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	No
45	S/W (tools, web services, workflows) must indicate format of their output	Partial
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	No
74	Machine-readable metadata for UIMA components	Partial
75	Embedding UIMA component metadata into the source code	Partial
76	Separating UIMA metadata from the component	Full
78	Specifying input and output types of UIMA components	Partial
79	Documentation of UIMA components	Partial
87	Embedding language capability in UIMA component metadata	Partial
88	Embedding output format in UIMA component metadata	No
89	Version documentation in parallel with component/resource	No
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	No
91	Encoding citable publications (for scholarly attribution) in resource metadata records	No
92	Including license text in resource packages	Partial
96	Unique identifiers and versions for components using Maven	Full
97	Declaring scaleout capability in UIMA	Full
98	Publishing components via software repositories (Maven, Docker)	No
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	No
102	Adding version information in the metadata descriptions of all resources	Partial
103	Specifying access mode of resources and encoding it in the metadata descriptions	Partial
104	Encoding funding information in the metadata descriptions of all resources	No
106	Encoding licensing terms in the metadata description of the resource	Partial
107	Encoding metadata on domain/subject/ classification for all resources when applicable	No
110	Assigning a unique persistent identifier for all resources	Full

Agrovoc (27)

Compliance	#	%
Full	18	67
No	8	30

Compliance	#	%
Partial	1	4

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
33	Licensing information must be included in the metadata	Full
36	Classification metadata should be included, where applicable, in the metadata record of the resource	Full
38	Access mode of resources must be included in the metadata	Full
41	Content resources must include metadata on their language(s)	No
44	Statistical metadata that allow monitoring of resource versions may accompany resources	Full
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	No
67	Knowledge Resource Element Id	Full
68	Data Category Linking Vocabulary	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	Full
71	The KR should be ingestible through a URI	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
89	Version documentation in parallel with component/resource	No
91	Encoding citable publications (for scholarly attribution) in resource metadata records	No
93	Provide identifiers for knowledge resource elements	Full
94	Data Category Linking Vocabulary	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	No
102	Adding version information in the metadata descriptions of all resources	Partial
103	Specifying access mode of resources and encoding it in the metadata descriptions	Full
104	Encoding funding information in the metadata descriptions of all resources	No
106	Encoding licensing terms in the metadata description of the resource	Full
107	Encoding metadata on domain/subject/ classification for all resources when applicable	Full
108	Encoding language information in the metadata of content resources	No
109	Encoding statistical information in the content resources	Full

ID	Requirement	Compliance
110	Assigning a unique persistent identifier for all resources	Full

Alvis (38)

Compliance	#	%
Full	7	18
No	16	42
Partial	13	34
Unknown	1	3
partial	1	3

ID	Requirement	Compliance
1	Components must be described by machine-readable metadata	Full
2	Component metadata have to be embedded into the component source code	No
3	Component metadata must be separable from the component	Full
5	Components must detail all their environmental requirements for execution	Partial
6	Components should have a unique identifier and a version number	Partial
7	Components must have a fully qualified name that follows the Java class naming conventions	Full
8	Components must associate themselves with categories defined by the OpenMinTeD project	Partial
9	Components must declare their annotation schema dependencies	No
10	Components should specify the types of the annotations that they input and output	Partial
11	Components must declare whether they can be scaled within a workflow	No
12	Components should provide documentation describing their functionality	Partial
13	Citation information for component should be included in the metadata	No
16	Models/resources should be useable across different component collections/platforms	Partial
17	Components should be stateless	Partial
21	Configuration and parametrizable options of the components should be identified and documented	Full
26	It should be possible to determine the source of an annotation/assigned category	Partial
27	Components should handle failures gracefully	No
28	Processing components should be downloadable	Full
33	Licensing information must be included in the metadata	No

ID	Requirement	Compliance
36	Classification metadata should be included, where applicable, in the metadata record of the resource	No
38	Access mode of resources must be included in the metadata	No
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	No
45	S/W (tools, web services, workflows) must indicate format of their output	No
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	No
89	Version documentation in parallel with component/resource	No
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	Unknown
91	Encoding citable publications (for scholarly attribution) in resource metadata records	No
92	Including license text in resource packages	partial
96	Unique identifiers and versions for components using Maven	Partial
98	Publishing components via software repositories (Maven, Docker)	Partial
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	Partial
102	Adding version information in the metadata descriptions of all resources	Partial
103	Specifying access mode of resources and encoding it in the metadata descriptions	Full
104	Encoding funding information in the metadata descriptions of all resources	No
106	Encoding licensing terms in the metadata description of the resource	Partial
107	Encoding metadata on domain/subject/ classification for all resources when applicable	No
110	Assigning a unique persistent identifier for all resources	Full

CLARIN CCR (8)

Compliance	#	%
Full	4	50
No	4	50

ID	Requirement	Compliance
67	Knowledge Resource Element Id	Full
68	Data Category Linking Vocabulary	No
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	No

ID	Requirement	Compliance
71	The KR should be ingestible through a URI	No
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
93	Provide identifiers for knowledge resource elements	Full
94	Data Category Linking Vocabulary	No
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full

CORE (19)

Compliance	#	%
Full	1	5
Partial	8	42
Unknown	10	53

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Partial
33	Licensing information must be included in the metadata	Partial
36	Classification metadata should be included, where applicable, in the metadata record of the resource	Partial
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	Partial
38	Access mode of resources must be included in the metadata	Full
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	Partial
41	Content resources must include metadata on their language(s)	Partial
47	Information on funding of resources may be included in the metadata	Partial
99	Encoding in the metadata a direct access link for content resources	Unknown
100	Providing access to content resources (sharing/exposing and transferring)	Unknown
102	Adding version information in the metadata descriptions of all resources	Partial
103	Specifying access mode of resources and encoding it in the metadata descriptions	Unknown
104	Encoding funding information in the metadata descriptions of all resources	Unknown
105	Encoding of format in the metadata description of content resources	Unknown
106	Encoding licensing terms in the metadata description of the resource	Unknown
107	Encoding metadata on domain/subject/ classification for all resources when applicable	Unknown
108	Encoding language information in the metadata of content resources	Unknown

ID	Requirement	Compliance
109	Encoding statistical information in the content resources	Unknown
110	Assigning a unique persistent identifier for all resources	Unknown

DKPro Core (47)

Compliance	#	%
Full	24	51
N/A	2	4
No	6	13
Partial	15	32

ID	Requirement	Compliance
1	Components must be described by machine-readable metadata	Full
2	Component metadata have to be embedded into the component source code	Full
3	Component metadata must be separable from the component	Full
5	Components must detail all their environmental requirements for execution	Partial
6	Components should have a unique identifier and a version number	Partial
7	Components must have a fully qualified name that follows the Java class naming conventions	Full
8	Components must associate themselves with categories defined by the OpenMinTeD project	No
9	Components must declare their annotation schema dependencies	Partial
10	Components should specify the types of the annotations that they input and output	Full
11	Components must declare whether they can be scaled within a workflow	Full
12	Components should provide documentation describing their functionality	Full
13	Citation information for component should be included in the metadata	No
16	Models/resources should be useable across different component collections/platforms	Full
17	Components should be stateless	Partial
21	Configuration and parametrizable options of the components should be identified and documented	Full
26	It should be possible to determine the source of an annotation/assigned category	Partial
27	Components should handle failures gracefully	N/A
28	Processing components should be downloadable	Full
33	Licensing information must be included in the metadata	Partial
34	Licensing information should be expressed in a machine-readable form	Partial

ID	Requirement	Compliance
36	Classification metadata should be included, where applicable, in the metadata record of the resource	Partial
38	Access mode of resources must be included in the metadata	Full
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	Partial
45	S/W (tools, web services, workflows) must indicate format of their output	Partial
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	Full
74	Machine-readable metadata for UIMA components	Full
75	Embedding UIMA component metadata into the source code	Partial
76	Separating UIMA metadata from the component	Full
78	Specifying input and output types of UIMA components	Full
79	Documentation of UIMA components	Full
87	Embedding language capability in UIMA component metadata	Partial
88	Embedding output format in UIMA component metadata	Full
89	Version documentation in parallel with component/resource	Full
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	No
91	Encoding citable publications (for scholarly attribution) in resource metadata records	No
92	Including license text in resource packages	Partial
96	Unique identifiers and versions for components using Maven	Full
97	Declaring scaleout capability in UIMA	Full
98	Publishing components via software repositories (Maven, Docker)	Full
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	Partial
102	Adding version information in the metadata descriptions of all resources	Full
103	Specifying access mode of resources and encoding it in the metadata descriptions	Full
104	Encoding funding information in the metadata descriptions of all resources	N/A
106	Encoding licensing terms in the metadata description of the resource	Partial
107	Encoding metadata on domain/subject/ classification for all resources when applicable	No
110	Assigning a unique persistent identifier for all resources	Full

Frontiers (17)

Compliance	#	%
Full	6	35
Partial	2	12
Unknown	9	53

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
33	Licensing information must be included in the metadata	Full
36	Classification metadata should be included, where applicable, in the metadata record of the resource	Partial
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	Full
38	Access mode of resources must be included in the metadata	Full
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	Full
41	Content resources must include metadata on their language(s)	Full
47	Information on funding of resources may be included in the metadata	Partial
102	Adding version information in the metadata descriptions of all resources	Unknown
103	Specifying access mode of resources and encoding it in the metadata descriptions	Unknown
104	Encoding funding information in the metadata descriptions of all resources	Unknown
105	Encoding of format in the metadata description of content resources	Unknown
106	Encoding licensing terms in the metadata description of the resource	Unknown
107	Encoding metadata on domain/subject/ classification for all resources when applicable	Unknown
108	Encoding language information in the metadata of content resources	Unknown
109	Encoding statistical information in the content resources	Unknown
110	Assigning a unique persistent identifier for all resources	Unknown

GATE (36)

Compliance	#	%
Full	11	31
No	11	31
Partial	8	22
Unknown	5	14
no	1	3

ID	Requirement	Compliance
1	Components must be described by machine-readable metadata	Full
2	Component metadata have to be embedded into the component source code	Partial
3	Component metadata must be separable from the component	Partial
5	Components must detail all their environmental requirements for execution	Partial
6	Components should have a unique identifier and a version number	Partial
7	Components must have a fully qualified name that follows the Java class naming conventions	Full
8	Components must associate themselves with categories defined by the OpenMinTeD project	no
9	Components must declare their annotation schema dependencies	No
10	Components should specify the types of the annotations that they input and output	Partial
11	Components must declare whether they can be scaled within a workflow	Full
12	Components should provide documentation describing their functionality	Full
13	Citation information for component should be included in the metadata	No
16	Models/resources should be useable across different component collections/platforms	Full
17	Components should be stateless	No
21	Configuration and parametrizable options of the components should be identified and documented	Full
26	It should be possible to determine the source of an annotation/assigned category	Partial
27	Components should handle failures gracefully	No
28	Processing components should be downloadable	Full
33	Licensing information must be included in the metadata	Partial
36	Classification metadata should be included, where applicable, in the metadata record of the resource	Partial
38	Access mode of resources must be included in the metadata	Full
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	No
45	S/W (tools, web services, workflows) must indicate format of their output	No
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	No
89	Version documentation in parallel with component/resource	Unknown
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	Unknown
91	Encoding citable publications (for scholarly attribution) in resource metadata records	Unknown

ID	Requirement	Compliance
92	Including license text in resource packages	Unknown
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	No
102	Adding version information in the metadata descriptions of all resources	Full
103	Specifying access mode of resources and encoding it in the metadata descriptions	Unknown
104	Encoding funding information in the metadata descriptions of all resources	No
106	Encoding licensing terms in the metadata description of the resource	Full
107	Encoding metadata on domain/subject/ classification for all resources when applicable	No
110	Assigning a unique persistent identifier for all resources	Full

ILSP (19)

Compliance	#	%
Full	1	5
No	3	16
Partial	3	16
Unknown	12	63

ID	Requirement	Compliance
33	Licensing information must be included in the metadata	Partial
36	Classification metadata should be included, where applicable, in the metadata record of the resource	No
38	Access mode of resources must be included in the metadata	Full
43	S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output	Partial
45	S/W (tools, web services, workflows) must indicate format of their output	Partial
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	No
89	Version documentation in parallel with component/resource	Unknown
91	Encoding citable publications (for scholarly attribution) in resource metadata records	Unknown
96	Unique identifiers and versions for components using Maven	Unknown
97	Declaring scaleout capability in UIMA	Unknown
98	Publishing components via software repositories (Maven, Docker)	Unknown
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	Unknown

ID	Requirement	Compliance
102	Adding version information in the metadata descriptions of all resources	Unknown
103	Specifying access mode of resources and encoding it in the metadata descriptions	Unknown
104	Encoding funding information in the metadata descriptions of all resources	Unknown
106	Encoding licensing terms in the metadata description of the resource	Unknown
107	Encoding metadata on domain/subject/ classification for all resources when applicable	Unknown
110	Assigning a unique persistent identifier for all resources	Unknown

JATS (15)

Compliance	#	%
Full	4	27
No	7	47
Partial	4	27

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Partial
33	Licensing information must be included in the metadata	Partial
38	Access mode of resources must be included in the metadata	No
41	Content resources must include metadata on their language(s)	No
44	Statistical metadata that allow monitoring of resource versions may accompany resources	No
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	No
67	Knowledge Resource Element Id	Full
68	Data Category Linking Vocabulary	No
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	Full
71	The KR should be ingestible through a URI	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Partial
93	Provide identifiers for knowledge resource elements	Full
94	Data Category Linking Vocabulary	No
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Partial

LAPPS (16)

Compliance	#	%
Full	9	56
No	7	44

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
33	Licensing information must be included in the metadata	No
38	Access mode of resources must be included in the metadata	No
41	Content resources must include metadata on their language(s)	No
44	Statistical metadata that allow monitoring of resource versions may accompany resources	No
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	No
67	Knowledge Resource Element Id	Full
68	Data Category Linking Vocabulary	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	Full
71	The KR should be ingestible through a URI	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
89	Version documentation in parallel with component/resource	No
93	Provide identifiers for knowledge resource elements	Full
94	Data Category Linking Vocabulary	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full

Licences (5)

Compliance	#	%
Full	1	20
No	1	20
Partial	2	40
Unknown	1	20

ID	Requirement	Compliance
33	Licensing information must be included in the metadata	Full
34	Licensing information should be expressed in a machine-readable form	Partial

ID	Requirement	Compliance
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	Partial
89	Version documentation in parallel with component/resource	Unknown

OLiA (26)

Compliance	#	%
Full	10	38
No	14	54
Partial	1	4
Unknown	1	4

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
33	Licensing information must be included in the metadata	No
38	Access mode of resources must be included in the metadata	No
41	Content resources must include metadata on their language(s)	No
44	Statistical metadata that allow monitoring of resource versions may accompany resources	No
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	No
67	Knowledge Resource Element Id	Full
68	Data Category Linking Vocabulary	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	Full
71	The KR should be ingestible through a URI	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
89	Version documentation in parallel with component/resource	No
91	Encoding citable publications (for scholarly attribution) in resource metadata records	No
93	Provide identifiers for knowledge resource elements	Full
94	Data Category Linking Vocabulary	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	No
102	Adding version information in the metadata descriptions of all resources	Partial

ID	Requirement	Compliance
103	Specifying access mode of resources and encoding it in the metadata descriptions	No
104	Encoding funding information in the metadata descriptions of all resources	No
106	Encoding licensing terms in the metadata description of the resource	No
107	Encoding metadata on domain/subject/ classification for all resources when applicable	Unknown
108	Encoding language information in the metadata of content resources	No
109	Encoding statistical information in the content resources	No
110	Assigning a unique persistent identifier for all resources	Full

Ontolex (8)

Compliance	#	%
Full	7	88
No	1	13

ID	Requirement	Compliance
67	Knowledge Resource Element Id	Full
68	Data Category Linking Vocabulary	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	No
71	The KR should be ingestible through a URI	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
93	Provide identifiers for knowledge resource elements	Full
94	Data Category Linking Vocabulary	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full

OpenAIRE (19)

Compliance	#	%
Full	1	5
No	1	5
Partial	6	32
Unknown	11	58

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Partial

ID	Requirement	Compliance
33	Licensing information must be included in the metadata	Partial
36	Classification metadata should be included, where applicable, in the metadata record of the resource	Partial
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	Partial
38	Access mode of resources must be included in the metadata	Partial
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	No
41	Content resources must include metadata on their language(s)	Partial
47	Information on funding of resources may be included in the metadata	Full
99	Encoding in the metadata a direct access link for content resources	Unknown
100	Providing access to content resources (sharing/exposing and transferring)	Unknown
102	Adding version information in the metadata descriptions of all resources	Unknown
103	Specifying access mode of resources and encoding it in the metadata descriptions	Unknown
104	Encoding funding information in the metadata descriptions of all resources	Unknown
105	Encoding of format in the metadata description of content resources	Unknown
106	Encoding licensing terms in the metadata description of the resource	Unknown
107	Encoding metadata on domain/subject/ classification for all resources when applicable	Unknown
108	Encoding language information in the metadata of content resources	Unknown
109	Encoding statistical information in the content resources	Unknown
110	Assigning a unique persistent identifier for all resources	Unknown

TheSoz (27)

Compliance	#	%
Full	20	74
No	5	19
Partial	2	7

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
33	Licensing information must be included in the metadata	Full
36	Classification metadata should be included, where applicable, in the metadata record of the resource	Full
38	Access mode of resources must be included in the metadata	Full
41	Content resources must include metadata on their language(s)	Full

ID	Requirement	Compliance
44	Statistical metadata that allow monitoring of resource versions may accompany resources	Partial
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	No
67	Knowledge Resource Element Id	Full
68	Data Category Linking Vocabulary	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	Full
71	The KR should be ingestible through a URI	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
89	Version documentation in parallel with component/resource	No
91	Encoding citable publications (for scholarly attribution) in resource metadata records	No
93	Provide identifiers for knowledge resource elements	Full
94	Data Category Linking Vocabulary	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	Partial
102	Adding version information in the metadata descriptions of all resources	Full
103	Specifying access mode of resources and encoding it in the metadata descriptions	Full
104	Encoding funding information in the metadata descriptions of all resources	No
106	Encoding licensing terms in the metadata description of the resource	Full
107	Encoding metadata on domain/subject/ classification for all resources when applicable	Full
108	Encoding language information in the metadata of content resources	Full
109	Encoding statistical information in the content resources	Full
110	Assigning a unique persistent identifier for all resources	Full

schema.org (8)

Compliance	#	%
Full	7	88
No	1	13

ID	Requirement	Compliance
67	Knowledge Resource Element Id	Full

ID	Requirement	Compliance
68	Data Category Linking Vocabulary	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	No
71	The KR should be ingestible through a URI	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full
93	Provide identifiers for knowledge resource elements	Full
94	Data Category Linking Vocabulary	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Full

Without justification

ID	Requirement	Product	Compliance
33	Licensing information must be included in the metadata	LAPPS	No
38	Access mode of resources must be included in the metadata	JATS	No
38	Access mode of resources must be included in the metadata	OLiA	No
38	Access mode of resources must be included in the metadata	LAPPS	No
41	Content resources must include metadata on their language(s)	Agrovoc	No
44	Statistical metadata that allow monitoring of resource versions may accompany resources	JATS	No
44	Statistical metadata that allow monitoring of resource versions may accompany resources	OLiA	No
44	Statistical metadata that allow monitoring of resource versions may accompany resources	LAPPS	No
45	S/W (tools, web services, workflows) must indicate format of their output	GATE	No
47	Information on funding of resources may be included in the metadata	TheSoz	No
47	Information on funding of resources may be included in the metadata	JATS	No
47	Information on funding of resources may be included in the metadata	LAPPS	No
47	Information on funding of resources may be included in the metadata	Licences	No
47	Information on funding of resources may be included in the metadata	GATE	No
50	Documentation references should be versioned	Agrovoc	No
50	Documentation references should be versioned	JATS	No
50	Documentation references should be versioned	OLiA	No
50	Documentation references should be versioned	LAPPS	No
50	Documentation references should be versioned	GATE	No
67	Knowledge Resource Element Id	TheSoz	Full
67	Knowledge Resource Element Id	Agrovoc	Full
67	Knowledge Resource Element Id	JATS	Full
67	Knowledge Resource Element Id	OLiA	Full
67	Knowledge Resource Element Id	LAPPS	Full
67	Knowledge Resource Element Id	CLARIN CCR	Full
67	Knowledge Resource Element Id	schema.org	Full
68	Data Category Linking Vocabulary	TheSoz	Full

ID	Requirement	Product	Compliance
68	Data Category Linking Vocabulary	Agrovoc	Full
68	Data Category Linking Vocabulary	JATS	No
68	Data Category Linking Vocabulary	OLiA	Full
68	Data Category Linking Vocabulary	LAPPS	Full
68	Data Category Linking Vocabulary	CLARIN CCR	No
68	Data Category Linking Vocabulary	schema.org	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	TheSoz	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	Agrovoc	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	OLiA	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	LAPPS	Full
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	Ontolex	No
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	CLARIN CCR	No
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	schema.org	No
71	The KR should be ingestible through a URI	Agrovoc	Full
71	The KR should be ingestible through a URI	OLiA	Full
71	The KR should be ingestible through a URI	LAPPS	Full
71	The KR should be ingestible through a URI	Ontolex	Full
71	The KR should be ingestible through a URI	CLARIN CCR	No
71	The KR should be ingestible through a URI	schema.org	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Agrovoc	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	OLiA	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	LAPPS	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Ontolex	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	CLARIN CCR	Full
72	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	schema.org	Full
88	Embedding output format in UIMA component metadata	ARGO	No
89	Version documentation in parallel with component/resource	Agrovoc	No

ID	Requirement	Product	Compliance
89	Version documentation in parallel with component/resource	OLiA	No
89	Version documentation in parallel with component/resource	LAPPS	No
89	Version documentation in parallel with component/resource	Licences	Unknown
89	Version documentation in parallel with component/resource	ARGO	No
89	Version documentation in parallel with component/resource	ILSP	Unknown
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	Alvis	Unknown
90	Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types	GATE	Unknown
91	Encoding citable publications (for scholarly attribution) in resource metadata records	GATE	Unknown
91	Encoding citable publications (for scholarly attribution) in resource metadata records	ILSP	Unknown
91	Encoding citable publications (for scholarly attribution) in resource metadata records	TheSoz	No
91	Encoding citable publications (for scholarly attribution) in resource metadata records	Agrovoc	No
91	Encoding citable publications (for scholarly attribution) in resource metadata records	OLiA	No
92	Including license text in resource packages	GATE	Unknown
93	Provide identifiers for knowledge resource elements	TheSoz	Full
93	Provide identifiers for knowledge resource elements	Agrovoc	Full
93	Provide identifiers for knowledge resource elements	JATS	Full
93	Provide identifiers for knowledge resource elements	OLiA	Full
93	Provide identifiers for knowledge resource elements	LAPPS	Full
93	Provide identifiers for knowledge resource elements	Ontolex	Full
93	Provide identifiers for knowledge resource elements	CLARIN CCR	Full
93	Provide identifiers for knowledge resource elements	schema.org	Full
94	Data Category Linking Vocabulary	TheSoz	Full
94	Data Category Linking Vocabulary	Agrovoc	Full
94	Data Category Linking Vocabulary	JATS	No
94	Data Category Linking Vocabulary	OLiA	Full
94	Data Category Linking Vocabulary	LAPPS	Full
94	Data Category Linking Vocabulary	Ontolex	Full
94	Data Category Linking Vocabulary	CLARIN CCR	No
94	Data Category Linking Vocabulary	schema.org	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Agrovoc	Full

ID	Requirement	Product	Compliance
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	OLiA	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	LAPPS	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	Ontolex	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	CLARIN CCR	Full
95	The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.	schema.org	Full
96	Unique identifiers and versions for components using Maven	ILSP	Unknown
97	Declaring scaleout capability in UIMA	ILSP	Unknown
98	Publishing components via software repositories (Maven, Docker)	ILSP	Unknown
99	Encoding in the metadata a direct access link for content resources	CORE	Unknown
99	Encoding in the metadata a direct access link for content resources	OpenAIRE	Unknown
100	Providing access to content resources (sharing/exposing and transferring)	CORE	Unknown
100	Providing access to content resources (sharing/exposing and transferring)	OpenAIRE	Unknown
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	ARGO	No
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	ILSP	Unknown
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	Agrovoc	No
101	Making models and annotation resources accessible as entities distinct from the components they are compatible with	OLiA	No
102	Adding version information in the metadata descriptions of all resources	OpenAIRE	Unknown
102	Adding version information in the metadata descriptions of all resources	Frontiers	Unknown
102	Adding version information in the metadata descriptions of all resources	ILSP	Unknown
103	Specifying access mode of resources and encoding it in the metadata descriptions	CORE	Unknown
103	Specifying access mode of resources and encoding it in the metadata descriptions	OpenAIRE	Unknown
103	Specifying access mode of resources and encoding it in the metadata descriptions	Frontiers	Unknown

ID	Requirement	Product	Compliance
103	Specifying access mode of resources and encoding it in the metadata descriptions	GATE	Unknown
103	Specifying access mode of resources and encoding it in the metadata descriptions	ILSP	Unknown
103	Specifying access mode of resources and encoding it in the metadata descriptions	OLiA	No
104	Encoding funding information in the metadata descriptions of all resources	CORE	Unknown
104	Encoding funding information in the metadata descriptions of all resources	OpenAIRE	Unknown
104	Encoding funding information in the metadata descriptions of all resources	Frontiers	Unknown
104	Encoding funding information in the metadata descriptions of all resources	ARGO	No
104	Encoding funding information in the metadata descriptions of all resources	GATE	No
104	Encoding funding information in the metadata descriptions of all resources	ILSP	Unknown
105	Encoding of format in the metadata description of content resources	CORE	Unknown
105	Encoding of format in the metadata description of content resources	OpenAIRE	Unknown
105	Encoding of format in the metadata description of content resources	Frontiers	Unknown
106	Encoding licensing terms in the metadata description of the resource	CORE	Unknown
106	Encoding licensing terms in the metadata description of the resource	OpenAIRE	Unknown
106	Encoding licensing terms in the metadata description of the resource	Frontiers	Unknown
106	Encoding licensing terms in the metadata description of the resource	ILSP	Unknown
106	Encoding licensing terms in the metadata description of the resource	OLiA	No
107	Encoding metadata on domain/subject/ classification for all resources when applicable	CORE	Unknown
107	Encoding metadata on domain/subject/ classification for all resources when applicable	OpenAIRE	Unknown
107	Encoding metadata on domain/subject/ classification for all resources when applicable	Frontiers	Unknown
107	Encoding metadata on domain/subject/ classification for all resources when applicable	ARGO	No

ID	Requirement	Product	Compliance
107	Encoding metadata on domain/subject/ classification for all resources when applicable	ILSP	Unknown
107	Encoding metadata on domain/subject/ classification for all resources when applicable	OLiA	Unknown
108	Encoding language information in the metadata of content resources	CORE	Unknown
108	Encoding language information in the metadata of content resources	OpenAIRE	Unknown
108	Encoding language information in the metadata of content resources	Frontiers	Unknown
108	Encoding language information in the metadata of content resources	Agrovoc	No
109	Encoding statistical information in the content resources	CORE	Unknown
109	Encoding statistical information in the content resources	OpenAIRE	Unknown
109	Encoding statistical information in the content resources	Frontiers	Unknown
109	Encoding statistical information in the content resources	OLiA	No
110	Assigning a unique persistent identifier for all resources	CORE	Unknown
110	Assigning a unique persistent identifier for all resources	OpenAIRE	Unknown
110	Assigning a unique persistent identifier for all resources	Frontiers	Unknown
110	Assigning a unique persistent identifier for all resources	ILSP	Unknown
110	Assigning a unique persistent identifier for all resources	Agrovoc	Full
110	Assigning a unique persistent identifier for all resources	OLiA	Full

Requirements

1. Components must be described by machine-readable metadata

Concreteness: *abstract*
Strength: *mandatory*
Status: *final*
Category: *WG4*

In order to incorporate components into the platform in an efficient manner, the platform must be able to automatically obtain information about them. Hence, components must provide machine-readable metadata by which they describe themselves.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Full	Alvis component descriptors	Draft
ARGO	0.5	Full	UIMA component descriptors	Draft
DKPro Core	1.8.0	Full	UIMA component descriptors, Maven project descriptors	Draft
GATE	8.2	Full	CREOLE descriptors	Draft
ILSP	1.2.1	Full	UIMA component descriptors	Draft

2. Component metadata have to be embedded into the component source code

Concreteness: *abstract*
Strength: *mandatory*
Status: *final*
Category: *WG4*

To avoid implementation and metadata getting out-of-sync, the metadata have to be closely integrated with the component source code such that e.g. parameter names and types are obtained directly from the implementation (cf. [uimafit]).

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Alvis XML descriptors manually maintained	Draft
ARGO	0.5	No	(I believe) - UIMA XML descriptors manually maintained	Draft
DKPro Core	1.8.0	Full	using uimaFIT Java annotations to automatically generate UIMA XML descriptors	Draft
GATE	8.2	Partial	using CREOLE Java annotations, but not yet in all components	Draft
ILSP	1.2.1	Partial	Both UIMA XML descriptors necessary for UIMA-AS integration, but also uimaFIT Java annotations to integrate components in command line pipelines	Draft

3. Component metadata must be separable from the component

Concreteness: *abstract*
Strength: *mandatory*
Status: *final*
Category: *WG4*

The component metadata should be provided in such a way that it is separable from the component. I.e. despite the canonical source of much of the metadata being the source code (and in a second instance, the compiled code), it should not be necessary to inspect the source code (or compiled code) to access the metadata or to actually invoke the component. Instead, the component should be accompanied by a metadata file that was automatically generated during build time. Component repositories for example should use this file when looking for component metadata. The file should be at a well-known location within the component artifact.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Full	Alvis XML exist separately from the code. Each XML descriptor is next to the component class it describes. By convention, the name of the XML file is created by appending the keyword "Doc" to the end the component name, for example GeniaTaggerDoc.xml for the component named GeniaTagger.	Draft
ARGO	0.5	Partial	UIMA XML descriptors manually maintained	Draft
DKPro Core	1.8.0	Full	using uimaFIT to automatically generate UIMA XML descriptors. XML descriptors as always next to the respective component classes. Pointers to all XML descriptors as stored in at `META-INF/org.apache.uima.fit/components.txt` within the respective JAR files. Maven POM files are also embedded in the JARs under `META-INF/maven/⋯/pom.xml`.	Draft
GATE	8.2	Partial	only for those components that still use CREOLE descriptors and do **not** use CREOLE Java annotations. It is, however, trivial to produce separate versions as GATE contains an ANT task to write CREOLE descriptors to disk from the Java annotations and it would be trivial to add this to the build process of each plugin.	Draft
ILSP	1.2.1	full (I think)	Maven POM files embedded in JARs under `META-INF/maven/⋯/pom.xml`. UIMA XML descriptors are included at the top level of the generated jars.	Draft

4. URL to actual content must be discoverable

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG2*,*WG3*
Status: *final*

The URL of the actual content, e.g. a text file, PDF file, or other common data format, must be derivable from readily discoverable data. It must either be included in the metadata or it must be reliably constructable from the metadata URL, e.g. by appending or changing a suffix. Derivation rules must be clearly documented.

NOTE This requirement refers only to content resources that can be used as input for a TDM process (i.e. corpora of publications etc.), or to knowledge resources (e.g. annotation schemas, typesystems, grammars etc.) that are used as ancillary resources for the operation of TDM components. The requirement is set for the proper operation of the TDM components in so far as they need to easily access the resources they operate on or with.

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Actual content (either pdf, text, or other) is available via constructing URL based on identifiers (internal CORE id, oai identifier, doi) provided in the metadata record	Final
OpenAIRE	Jun-16	Partial	There's a link to the doi but not always; and usually the doi sends to a landing page, but there are some rules that can be used for "guessing" the download link	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Full	From XMLS, the URL can be reliably constructed using the DOI.	Final
TheSoz	Jun-16	Full	Url to the resource and the SPARQL endpoint is provided in the void.ttl file (Should be located at: http://lod.gesis.org/thesoz/void.ttl however it is not at the moment available online due to technical issues).	Final
Agrovoc	21/01/2016	Full	Url to the two versions of the resource and the SPARQL endpoint is provided in the void.ttl file (http://aims.fao.org/aos/agrovoc/void.ttl)	Final
JATS	1.1	Partial	although the URL to the JATS schemas (as DTDS, Relax NG & XSD) and elements is available at stable URIs, each at a separate folder (info is provided at https://jats.nlm.nih.gov/files.html), there is no formal metadata to indicate this	Final
OLiA	Jun-16	Full	all resources are at the landing page: http://acoli.cs.uni-frankfurt.de/resources/olia/	Final
LAPPS	Jun-16	Full	in various formats, available from http://vocab.lappsgrid.org/	Final

5. Components must detail all their environmental requirements for execution

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG4*
Status: *draft*

Text and Data Mining (TDM) components are not always self-contained entities; for example, a component maybe a wrapper to another library developed in a language not supported by the TDM framework. This library may have additional system dependencies that must be available at runtime - think of a Java TDM component wrapping a python library; the python library will require Python and potentially other Python libraries. Knowing the environmental requirements allows a workflow execution system to ensure that they are met before the component is initialised.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Some components may need external libraries or resources. Alvis works better in a Linux environment.	Draft
ARGO	0.5	Partial	All components are distributed as UIMA PEAR files written in Java, with some components requiring platform-specific binaries. Platform-specific binaries are provided for OSX, Windows and Linux. This appears to be the same behaviour as DK Pro. When running a workflow on a cluster/cloud, Argo requires an expected maximum memory limit per component instance in a workflow, and this is not included in the component's metadata - it has to be entered manually by users.	Draft
DKPro Core	1.8.0	Partial	All components are in Java, some require platform-specific binaries which are provided usually for Linux, Windows, and OS X in a pre-compiled form. There is no metadata explaining which components rely on binaries or about the supported platforms and OS versions. E.g. some binaries don't work on modern Linuxes (hunpos) others do not work on old versions. The required Java version is available through a property defined in the POM.	Draft
GATE	8.2	Partial	All components are written in Java although some may make use of external platform-specific binaries. Where this is the case either the binaries are provided as part of the plugin, or documentation is provided explaining how to obtain and install the appropriate components. There is no metadata to highlight the need for external components, although the metadata can (and usually does) provide a URL to the documentation where it would be discussed in detail.	Draft

Product	Version	Compliant	Justification	Status
ILSP	1.2.1	Full	All components are in Java. The required Java version is available through a property defined in the POM.	Draft

6. Components should have a unique identifier and a version number

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG4*
Status: *draft*

Components should contain an identifier by which they can be distinguished from each other, in addition to their version. A component registry would then be able to use the combination of the identifier and version number to produce resolvable URLs from which components can be retrieved.

See also

- 7. Components must have a fully qualified name that follows the Java class naming conventions

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Components (instances of components) are well identified in a workflow but component versioning is not maintained in Alvis.	Draft
ARGO	0.5	Partial	Each component has a unique id under the `uk.ac.nactem.uima` namespace (derived from the path of the UIMA descriptor file within the component's UIMA PEAR file) and a version number specified within the component's UIMA descriptor file. Argo doesn't currently support the concept of component versions; it only uses whichever version of the component that was installed last.	Draft
DKPro Core	1.8.0	Partial	Each component has a unique name under the `de.tudarmstadt.ukp.dkpro` namespace (basically a Java class name) and a version. The version of a component corresponds to the version of the JAR that contains the version. Each component is contained in a JAR which has a unique Maven GAVC coordinate (e.g. `de.tudarmstadt.ukp.dkpro.core:de.tudarmstadt.ukp.dkpro.core.opennlp-asl:1.8.0:jar`).	Draft

Product	Version	Compliant	Justification	Status
GATE	8.2	Partial	Each component is a Java class and as such has a unique name. Currently, however, there is no requirement for a GATE plugin or component to have a version number. In most cases we distribute GATE with a large set of plugins which are assumed to share the same version number as the framework itself. Plugins (which may contain multiple components) distributed via an update site must include a version number and all components inside that plugin share that. Experience has shown that this is far from ideal and our future plans for GATE include moving to using Maven to distribute plugins in which case each plugin will be uniquely identified by a set of Maven coordinates, although support for loading legacy plugins without a version number will remain for the foreseeable future.	Draft
ILSP	1.2.1	Partial	Each component has a unique name under the `gr.ilsp.nlp` namespace (basically a Java class name) and a version. The version of a component corresponds to the version of the JAR that contains the version. Each component is contained in a JAR which has a unique Maven GAVC coordinate (e.g. `gr.ilsp.nlp:ilsp-nlp-lemmatizer:1.2.1-SNAPSHOT.jar`). We try to use some type of semantic versioning (http://semver.org/) for the components, but for some of them this is not so strictly followed.	Draft

7. Components must have a fully qualified name that follows the Java class naming conventions

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG4*
Status: *final*

Using such a widely adopted naming convention assists component developers through familiarity and helps avoid naming collisions due to the use of domain names, under the control of the developers, as part of the identifier. It is also the recommended method of identifying components in existing text mining systems such as Argo and GATE.

NOTE

This requirement was previously titled "Components should be uniquely identified using a string that follows the Java fully-qualified class naming convention". In particular the word "uniquely" has been removed because a Java fully-qualified name does not include version information and in principle there is no safe-guard against having multiple Java classes by the same name on the classpath at the same time. The main point here is that users name their components within a namespace that can be considered under their control. The Java conventions then provide a best-practice how to choose and structure such a namespace. Additional best-practices and techniques, e.g. the Maven best-practices and version-resolving techniques then should ensure that only a single version of a given component is available at runtime. If a system needs to use multiple versions of a component, additional precautions may need to be necessary.

See also

- 6. Components should have a unique identifier and a version number

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Full	Components are Java classes organised through packages.	Draft
ARGO	0.5	Partial	All of our current components are identified by strings that follow the Java fully-qualified class naming convention, however as the component identifier within Argo is derived from a filepath it is entirely possible for this requirement to be broken by new components. For example, the UIMA descriptor for the Argo component Chemical Entity Recogniser has the path `/desc/uk/ac/nactem/uima/ChemicalEntityRecogniser.xml` which translates into the identifier `uk.ac.nactem.uima.ChemicalEntityRecogniser`.	Draft
DKPro Core	1.8.0	Full	Components are identified by a Java class name. Per Java/Maven conventions, this classname is unique within DKPro Core. There is also a version associated with each class corresponding to the version of the enclosing Maven project/JAR.	Draft
GATE	8.2	Full	All GATE components are implemented as Java Beans and as such fulfill this requirement.	Draft
ILSP	1.2.1	Full	Components are identified by a Java class name. They are by convention unique within nlp.ilsp.gr.	Draft

8. Components must associate themselves with categories defined by the OpenMinTeD project

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG4*
Status: *final*

With a large number of components, it can be difficult for workflow creators to find the most appropriate components if they can only search for components by name or by reading each component's documentation. Having components associated with categories would enable workflow creators to easily search for a particular type of component (e.g. part-of-speech tagger) as well as allowing associated UI tooling an additional way of ordering lists of components on-screen.

To be moved to FR in WP6

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Alvis does not categorize the components but can follow a categorization schema.	Draft
ARGO	0.5	Partial	Currently Argo puts components into 4 categories (reader, writer, analytics, web service), however the bulk of the components reside within the analytics category, and the only method for users to find a relevant component is to search through the entire list within the UI. The reader, writer and analytics components are identified by examining their UIMA descriptor files (looking for either the collectionReaderDescription, casConsumerDescription and analysisEngineDescription elements). Interactive components are identified by the `interactive` flag in their Argo descriptor file (An Argo descriptor is a complimentary file to the UIMA descriptor, providing additional information. For example, it can help Argo select the most appropriate UI widget for configuration parameters, such as strings that represent types or files.)	Draft
DKPro Core	1.8.0	No	DKPro Core components try to follow a naming scheme, e.g. `XxxReader`, `XxxWriter`, `XxxSegmenter`, `XxxPosTagger`, etc. Categories can be derived off these names in many cases. In some cases, components fulfill multiple purposes, e.g. TreeTagger does lemmatization and POS tagging. A second source off which category information could be derived are the input/output types declared by the components, e.g. `Sentence`, `Token`, etc. There is no direct associated with any (external) categorization system.	Draft

Product	Version	Compliant	Justification	Status
GATE	8.2	no	GATE does not currently enforce such a categorization (or have any way internally of doing so). For the core plugins (i.e. those developed in Sheffield) we try and stick to a naming convention that gives some idea of the nature of the plugin; Lang_X, Parser_X, Tagger_X, etc. but this is purely a convention and is in no way fine grained enough to be used for indexing or other forms of automatic discovery.	Draft
ILSP	1.2.1	no	By convention we package all readers, exporters, and instantiations of uimafit pipelines as three different projects. A package is also used for uima service clients. The rest of the packages concern analyzers and follow an ilsp-nlp-XXX naming scheme, with XXX being a string like lemmatizer or depparser.	Draft

9. Components must declare their annotation schema dependencies

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG4*
Status: *final*

A workflow execution system will need to know which annotation schemas to load when running a component within a workflow. Not having this information would result in the system having to load all of the annotation schemas it knows about at runtime for each component or force the workflow creators to manually specify them, both highly inefficient options. This information also forms part of the components documentation, assisting workflow creators.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Work in progress.	Draft
ARGO	0.5	Full	Argo components must declare the type systems they depend upon within their UIMA descriptor file.	Draft
DKPro Core	1.8.0	Partial	Components live within Maven artifacts. These declare dependencies on other Maven artifacts which contain the type system definitions. Most component declare input and output types. Thus, the XML type system descriptors for a component can be obtained by scanning the Maven dependencies for XML type descriptors that declare types which are used as input/output types by a given component.	Draft
GATE	8.2	No	GATE does not require components to specify annotation schemas, and has no way (currently) of recording that information even if it were available. This is because GATE does not enforce a type system. This does mean that no annotation schemas have to be loaded for a GATE component to execute. This lack of an enforced type system is a feature of GATE and not a bug and so is unlikely to change, meaning any requirement to specify schemas per component is likely to only ever be optional within GATE.	Draft
ILSP	1.2.1	Partial	UIMA components declare the type system they depend upon within their descriptor file. The type system is a Maven dependency of all readers, analyzers and exporters. Components declare input and output types.	Draft

10. Components should specify the types of the annotations that they input and output

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG4*,*WG2*
Status: *draft*

To assist in the creation of workflows, it would be helpful to know what types of annotations are required and produced by components so that UI tooling can display appropriate warnings when a component is added in an invalid position within a workflow or to help diagnose runtime exceptions whilst running a workflow.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	The type system is shared by all the components.	Draft
ARGO	0.5	Partial	Some components declare the types that they input and output, within the component's UIMA descriptor, however this is not always the case and is not required by Argo.	Draft
DKPro Core	1.8.0	Full	Most components declare the input and output types using UIMA **capabilities**. For some components, the types they operate on are configured through parameters.	Draft
GATE	8.2	Partial	GATE does not enforce any type system and as such does not require components to declare in any way their input/output types. That being said it does support XML schemas for defining annotations (and their features) and so it would be possible to bundle a set of schemas with a component (i.e. within the JAR file) that defined the expected input/output annotations. For example, the ANNIE plugin already comes with schemas which define the main NE annotations produced by the application (GATE's terminology for a workflow). We also have a component, the Schema Enforcer, that can be used to ensure that the output of an application strictly conforms to a given set of schemas removing any annotations or features not defined by the schemas. The manual annotation aspects of GATE Developer can also be configured to only allow annotations/features defined by XML schemas to be created.	Draft
ILSP	1.2.1	Partial	Components declare the input and output types using UIMA capabilities.	Draft

11. Components must declare whether they can be scaled within a workflow

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG4*
Status: *draft*

To accommodate scaling of a workflow, it is imperative to understand which components can be deployed multiple times. This is something that the developer will have to specify manually and components that cannot be scaled should be exception rather than the rule.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Do not deal with this aspect.	Draft
ARGO	0.5	Full	Components use the UIMA flag `multipleDeploymentAllowed` to indicate if they can be scaled out or not. Argo also currently doesn't scale reader, writer, remote (e.g. component development sdk) and interactive (e.g. annotation editor) components.	Draft
DKPro Core	1.8.0	Full	Components use the UIMA flag `multipleDeploymentAllowed` to indicate if they can be scaled out or not. This flag is used mostly by writer components.	Draft
GATE	8.2	Full	Being Java Beans a GATE component is inherently single threaded, although there is no reason why a component cannot be instantiated multiple times within or across JVM's. It is, however, worth pointing out that within a JVM naively creating multiple instances of a component is probably not a good idea as this is likely to be wasteful in terms of memory as multiple instances of large data structures might be created. GATE contains methods to duplicate components and entire workflows in a more sensible fashion to share internal structures where safe to do so.	Draft
ILSP	1.2.1	Full	Most of the analyzers have been deployed as UIMA-Asynchronous Scaleout services and as parts of UIMA-AS aggregate services without any obvious issues. The scaleout capabilities of the UIMA-AS framework and of DUCC have not been fully explored.	Draft

12. Components should provide documentation describing their functionality

Concreteness: *abstract*
Strength: *recommended*
Category: *WG4*
Status: *final*

Having some form of free-text description assists text and data miners in choosing the most appropriate components for their workflows and potentially boosts productivity, as it possibly reduces the need for experimentation if a component has configuration parameters.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Components have descriptions as html streams containing formatting tags.	Draft
ARGO	0.5	Partial	Components can declare a textual description using the `description` element within its associated UIMA descriptor file, however Argo doesn't enforce its existence.	Draft
DKPro Core	1.8.0	Full	Most components have some kind of JavaDoc-based description. This is carried over into UIMA descriptors and Java annotations using the uimafit-maven-plugin during build time. This makes the descriptions available through Java reflection as well as in the UIMA XML descriptors. We use these descriptors as part of auto-generating the DKPro Core reference documentation. The quality of the documentation varies.	Draft
GATE	8.2	Full	The CREOLE metadata that describes a component includes the name and a brief description as well as a URL to the full documentation of the component. None of these are, however, mandatory and documentation varies across components (i.e. we have no control of the documentation of components developed by 3rd parties)	Draft
ILSP	1.2.1	Partial	Most components have a minimal textual description in the UIMA descriptor file, which is often replicated in the Maven POM. If applicable, a reference to a scientific article is also included.	Draft

13. Citation information for component should be included in the metadata

Concreteness: *abstract*
Strength: *recommended*
Category: *WG1*,*WG4*
Status: *draft*

It is important that citable publications for each component can be obtained from the component metadata.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	No citation information in general, the documentation of the modules sometimes contains citation information.	Draft
ARGO	0.5	No	No citable publication information available.	Draft
DKPro Core	1.8.0	No	A few components contain references to relevant publications in their documentation, but it is not machine readable.	Draft
GATE	8.2	No	GATE does not support this, although it's likely that the URL to the documentation may well lead to a publication or documentation that references one.	Draft
ILSP	1.2.1	Unknown	Available for some components but not always applicable or used.	Draft

14. Components must maintain License information

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. Its is covered by REQ-33, REQ-34 and expected functionality of the OpenMinTeD platform.

See also

- 33. Licensing information must be included in the metadata
- 34. Licensing information should be expressed in a machine-readable form

It is important that not only is a license strongly assigned to components and resources but that this information is passed upwards through workflows etc. so that a license can be assigned to an aggregate or to newly created resources that result. To this end storing license information within component metadata seems the most sensible way forward.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	The licence information are currently provided as free texts in the tool documentation.	Draft
ARGO	0.5	Partial	Components can declare licensing information with the Argo Descriptor File, however the vast majority of components do not make use of this facility.	Draft
DKPro Core	1.8.0	Partial	DKPro Core components have license metadata about their own code in their POM hierarchy. It is also available for those dependencies that provide license information in their respective POMs (not all dependencies declare such license information in their POMs). For most models/resources, we currently have no license information.	Draft
GATE	8.2	No	Currently GATE components do not have a license assigned as part of the metadata. The next version of GATE will use maven for plugin/component distribution and will have access to license information documented in the pom.xml for a plugin.	Draft
ILSP	1.2.1	Partial	ILSP components do not have a license assigned as part of the metadata. ILSP components are provided free for research purposes via UIMA, SOAP and/or rest services. The license information for these services is declared via appropriate metadata as for example in https://goo.gl/yDynbu	Draft

15. Human readable information should be provided by each resource

Concreteness: *abstract*
Strength: *recommended*
Category: *WG1*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. It is overlapping with REQ-12, WG1-5 and is not sufficiently clear.

Capturing, computing and presenting simplified information (e.g., availability, reliability, QoS, provider, class) about resources bring decisional information to best choose a specific resource compared to others. For example, based on aspects such as availability or QoS, one may create a specific view to filter out some specific services.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Available for the common features (e.g., name, description, provider,...)	Draft
ARGO	0.5	Full	Argo components have a human readable description.	Draft
DKPro Core	1.8.0	Full	Components should come with a description (primarily obtained from the JavaDoc of the component). That information may not be particularly comprehensive though. Availability, reliability, and the other example information given above are not relevant for DKPro Core as it is not service-based.	Draft
GATE	8.2	Full	Each component has a human readable name and description as part of the metadata	Draft
ILSP	1.2.1	No	No availability or QoS information is provided for components available as web services. Readable names, descriptions, Terms Of Service, provider etc. are provided as metadata for these services.	Draft

16. Models/resources should be useable across different component collections/platforms

Concreteness: *abstract*
Strength: *recommended*
Category: *WG4*
Status: *final*

Different platforms/component collections that wrap the same tools should be able to be configured in similar or uniform ways to use models/resources. E.g. all wrappers should permit loading models from an arbitrary location (on disk, JAR, URL), and not require that they be packaged with the component. Support for a common repository infrastructure from which to obtain models/resources automatically would be beneficial.

NOTE	Maven is sometimes used today to store models/resources. Although Maven is largely popular in the Java world, there is e.g. also a client implementation for Python which would allow e.g. NLTK to use models stored on Maven repositories (jip).

Source: WG 4 Scenario 1 — Transferability of components between ecosystems

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Alvis has a stream manager that enables to characterize models/resources to load or to store, it supports file system, web access, zip/jar,... and is extensible.	Draft
ARGO	0.5	Partial	Argo components that offer configurable models/resources make use of standard UIMA configuration parameters. The majority of these configuration parameters accept a local file path, which relates to a user's file store within the Argo ecosystem.	Draft
DKPro Core	1.8.0	Full	DKPro Core components that use resources can typically be configured through two mechanisms: 1) via the **language/variant** coordinates which internally translate to a classpath lookup, and may make use of the built-in auto-download mechanism for models. 2) via the **PARAM_MODEL_LOCATION** which supports locally available resources (file system, classpath, ZIP/JAR files, but not arbitrary remote URLs).	Draft
GATE	8.2	Full	GATE components that are configured through models/resources etc. simply require a URL to the resources. This could be a file or jar URL as well as anormal http URL. Resources may be bundled inside jar files for distribution but these are simply the default files and can always be replaced.	Draft
ILSP	1.2.1	Partial	ILSP UIMA-based components are typically configured via uimafit ConfigurationParameter and/or ExternalResource annotations. As services, users can only configure parameters things like exporting format.	Draft

17. Components should be stateless

Concreteness: *concrete*
Strength: *recommended*
Category: *WG4*
Status: *final*

To enable scaling of components, using networked machines, it should be recommended that a component avoids any form of local state with regards to the processing of a document.

NOTE	If a component is required to be stateful, it would make sense for the platform to offer a solution to allow scaling of stateful components.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	In a general way, there is no local state for a component and determinism is guaranteed. However, aspects such as resource availability and resource access may imply state issues that Alvis core engine has to manage.	Draft
ARGO	0.5	Partial	Most Argo components are stateless. It is possible to indicate that a component is stateful (which will result in only a single instance of that component being permitted within a distributed execution) by setting the multipleDeploymentAllowed value to false in the UIMA descriptor file. Argo also automatically treats a subset of components as being stateful, which includes Reader, Writers and Interactive Components.	Draft
DKPro Core	1.8.0	Partial	Most DKPro Core components are stateless in the described sense: no information needs to be shared between multiple instances of the same component. There is the case of sharing models between multiple instances and this is experimentally supported by DKPro Core but can only be used if the models themselves are threadsafe/stateless. Note that components are **not** threadsafe! A single component instance cannot be safely invoked from multiple threads! A notable exception to statelessness are writer components which are able to aggregate information, e.g. writing all pipeline output to a single file. Such components cannot be sensibly scaled out. They are marked with the UIMA operational property **multipleDeploymentAllowed** = **false**.	Draft

Product	Version	Compliant	Justification	Status
GATE	8.2	No	**never** - GATE components are all Java beans. This means that the document to process is provided to the component via a set method making it part of the state of the component. Multiple instances of a component can of course be created to enable processing of multiple documents in parallel. Naively creating multiple instances should, however, be avoided as this is wasteful in terms of memory as it requires multiple copies of internal state that can be shared. The framework supports intelligent duplication of components and applications to share appropriate state across threads.	Draft
ILSP	1.2.1	Full	ILSP components deployed as services avoid any form of local state. Multiple instances of a service deployed on the same machine can share lexical resources and models.	Draft

18. Workflows should be described using an uniform language

Concreteness: *abstract*
Strength: *recommended*
Category: *WG4*
Status: *deprecated*

IMPORTANT This requirement has been deprecated. It should be covered by the functionality of the OpenMinTeD platform, namely Galaxy workflow editor.

A workflow represents an experiment that users describe, visualize, execute, modify and shared. To facilitate the lifecycle, the workflows should be described in a uniform way so that the instantiated components, parameters and data flow to be understandable by users as well as by the engines.

Note - we may wish to be more specific with this requirement

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Alvis has a workflow representation that only accepts sequence of modules and parameter configuration on the individual modules. The workflows are expressed using XML constructs that the Alvis engine can interpret. The workflow expression can be modified, transfered and executed in different Alvis instances. The workflow representation is however only used within Alvis.	Draft
ARGO	0.5	Partial	Possibly misinterpreting this requirement, but Argo contains a visual web-based workflow editor which allows a user to easily compose a workflow using components, which is then translated into a format understood by the execution engine within Argo. Argo is only partially compliant as the workflows it produces cannot currently be shared outside of the Argo platform.	Draft
DKPro Core	1.8.0	Unknown	DKPro Core is build on UIMA and can be invoked through UIMA by any workflow manager or execution environment that supports UIMA. We do not really care whatever language that environment is using to describe their workflows. Many users like using DKPro Core components with uimaFIT (often either using SimplePipeline or CpeBuilder) - DKPro Core provides examples for this in various languages (Java, Groovy, Jython). LAPPS Grid integrates services based on DKPro Core components with their Galaxy workflow engine.	Draft
GATE	8.2	Unknown	GATE has a standard description for workflows (as will any existing framework) but I really don't understand what this requirement is asking.	Draft

Product	Version	Compliant	Justification	Status
ILSP	1.2.1	Unknown	ILSP components are combined in workflows using UIMA (AS) aggregate descriptors and/or using the workflow functionalities of processing infrastructures like CLARIN and METASHARE/QT21.	Draft

19. Components that use external knowledge resources should delegate access to a resource adapter instead of handling it themselves

Concreteness: *abstract*
Strength: *optional*
Category: *WG2,WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated following a discussion in WG4 that came to the conclusion the proposed approach would be over-engineering. Taking a gazetteer as an example, it was considered the main functionality of a specific gazetteer implementation to perform lookups. So there may be a DatabaseGazeteer, a FileGazeteer, or a SPARQLGazeteer. Adding another layer of abstraction to create a GenericGazeteer to be configured with a DatabaseLookup or FileLookup was not seen as important.

A component that uses an external knowledge resources, e.g. a dictionary, should access that resource through an adapter instead of accessing it directly. E.g. a Gazeteer component could be configured with a WordListFileResourceAdapter that looks up words from a simple word list file or using some SparqlQueryAdapter that would perform the lookup via Sparql. Thus, providing access to different knowledge sources should not require changes to the component implementation itself, but rather require only the implementation of lightweight adapters. It is not important that all implementations use the same adapters, but rather that this abstraction exists in the first place. E.g. a GATE component may use a GATE-specific adapter implementation and a UIMA component may use an UIMA-specific implementation.

NOTE	UIMA External Resources could be used for this and I am pretty sure GATE also has one or more abstraction for such things.

Source: WG 2 Scenario 3 — The relation between documents and knowledge bases through keywords

Required actions: While UIMA has the general abstraction, there are no common implementations for different types of knowledge resources. I think GATE has some typical/common abstractions like a gazetteer and probably others. These should be inspected and additional recommendations for specific resource types may be given in future requirements. Mind that even if GATE/UIMA have such abstractions, it does not mean that they are consistently used or used at all. This should be brought to the attention of the component collection providers.

NOTE	This requirement may not be necessary in small-scale resource lookup.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Unknown	Unknown	Draft

Product	Version	Compliant	Justification	Status
ARGO	0.5	Unknown	Components developed for Argo can make use of UIMA's external resources feature, however I don't believe there are many (if any) Argo components that do. On the Argo development roadmap, we plan on using this approach with Argo reader and writer components. Currently for every serialisation format (e.g. CAS XMI, RDF/XML) and storage platform (e.g. Argo File Store, SFTP) combination we require a new component to be created. Instead, we plan to allow the addition of serialisation formats and storage platforms independently.	Draft
DKPro Core	1.8.0	Partial	We experimented with such an approach e.g. in the SemanticFieldAnnotator where a resource abstract for semantic tagging resources exists. The idea was e.g. to read semantic tags either from a plain text file or alternatively from a Uby database. The ResourceObjectProvider abstraction internally used by DKPro Core to resolve and load models is also kind of related to this requirement. E.g. there is an experimental function to enable a behind-the-scene sharing of loaded models between multiple component resources. This works provided the models themselves are thread-safe.	Draft
GATE	8.2	Unknown	Unknown	Draft
ILSP	1.2.1	Partial	Some of the classes that provide access to resource data for ILSP components implement the org.apache.uima.resource.SharedResourceObject interface.	Draft

20. Workflow engines should not require to see data

Concreteness: *concrete*
Strength: *recommended*
Category: *WG2,WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. It continues to exist as the functional requirement FS/WFEX/07.

When assembling a workflow and running it, the workflow engine should not require that all data be transferred between the components by passing it through the engine. In the best case, this would only incur a performance overhead. In the worst case, it would require potentially sensitive or proprietary data to leave its source infrastructure, even if the actual analysis components deployed would be deployed on the same infrastructure (that is assuming that the workflow engine does not also run on the same infrastructure). This could e.g. be solved by processing components talking directly to each other.

Source: WG1 Scenario 2 — SME running research analytics for funders within the European Research Area

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Alvis uses a shared data structure, the workflow engine is required to manage the data and the modules during the execution steps. However the Alvis modules could be encapsulated with the Alvis engine in a OpenMinTeD workflow.	Draft
ARGO	0.5	Partial	On certain infrastructures with shared storage, it is possible for data to bypass the workflow engine. The Argo development roadmap does contain a plan to allow components to be speak directly to one another.	Draft
DKPro Core	1.8.0	N/A	DKPro Core itself does not provide workflow functionality. It's components can in principle be used in a workflow setup where data is passed directly between components, i.e. without going through a central workflow manager. But providing such a manager is beyond the scope of DKPro Core.	Draft
GATE	8.2	Partial	GATE components don't talk to each other, passing data is the job of a pipeline (which is akin to a workflow). However, if within an OpenMinTeD workflow there were a sequence of GATE components then these could be aggregated into a GATE pipeline and deployed as a single unit meaning that data would be kept within the pipeline.	Draft
ILSP	1.2.1	Partial	For ILSP components deployed as UIMA AS services, data to be transferred from, say, component A to component B that are remotely deployed on node X are not first passed through a message broker running on node Y	Draft

21. Configuration and parametrizable options of the components should be identified and documented

Concreteness: *abstract*
Strength: *recommended*
Category: *WG4*
Status: *final*

Components in systems such as Alvis offer several parameters used to configure the behavior of the component execution. In order to help users understand a specific parameter and its role, the parameters should be well identified and described. The descriptions should contain information to help users understand how and in what context a parameter is used, and also provide the data types (e.g., boolean, integer, string) accepted by a parameter.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Full	The parameters are well-typed in Alvis, they may be simple (e.g., integer, String, boolean) or complex (e.g., list, map). A description is planned for each parameter even if some parameters currently lack descriptions.	Draft
ARGO	0.5	Full	As Argo components are also UIMA components containing UIMA XML descriptor files, all configuration parameters have a name, a type (String, Integer, Float, or Boolean), an optional description, can be multivalued and also made to be mandatory. Argo also allows an additional layer of information on top of the UIMA configuration parameters, such as indicating whether a string represents a file, a folder, or an enumeration. This is intended to assist in both validation of configuration values and in providing the most appropriate graphical widgets within the Argo UI.	Draft
DKPro Core	1.8.0	Full	Parameters are implemented in DKPro Core through the uimaFIT `@ConfigurationParameter` annotation on class fields. These fields are typed and usually have documentation. They can have a default value and they can be marked as optional/mandatory. The annotation is retained for runtime such that parameter information can be dynamically obtained via uimaFIT. Additionally, UIMA XML descriptors are automatically generated for each component which also include the parameter declarations and default values. The parameter types supported are the standard UIMA-supported types (boolean, int, float, String). Additional types are supported via uimaFIT' automatic type-coercion mechanism. In this way, fields that are annotated as `@ConfigurationParameter` can e.g. be of the type `File` or `Pattern` or be different Java collection types or arrays.	Draft

Product	Version	Compliant	Justification	Status
GATE	8.2	Full	Each parameter is typed and has a name and an optional description (most have a description but it is currently optional)	Draft
ILSP	1.2.1	Unknown	Unknown	Draft

22. The Workflow Engine Should Permit Saving Experimental Conditions in a Workflow

Concreteness: *abstract*
Strength: *recommended*
Category: *WG1*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement is deprecated. It should be added back as a functional requirement for the workflow execution service (WFEX).

For reproducibility reasons extract and save/export (?) the exact experimental conditions of a workflow, such as which input files used (knowledge base/corpus), which components are used and in what order, which are the values of the parameters required per component, etc. (Requirement extracted from Interoperability Scenario WG1-4)

See also

- 46. Output resources of web services/workflows must be accompanied by provenance metadata

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Alvis allows this through its workflow expression.	Draft
ARGO	0.5	Partial	Argo stores the configuration parameters, order of components etc. alongside each workflow, so it can be ran again with the same values at a later date. However, this may not always allow the exact same conditions to be reproduced, due to the nature of the components in the workflow, for example, a reader component that returns documents based on a web service search may provide different documents on a subsequent run, if the index behind the web service has been updated in the meantime. Argo also lacks the ability to store sets of configuration parameter values against workflows - instead, if a user wants to change the configuration values of a workflow whilst keeping the previous set, they must first duplicate the workflow and make changes to the copy.	Draft

Product	Version	Compliant	Justification	Status
DKPro Core	1.8.0	Partial	DKPro Core readers adds a DocumentMetaData annotation to each document which contains basic information from where the document was obtained and how it can be identified (usually a file URL). Minimal information about the components/models that were used is added as annotations to the documents as well. This is mainly used as supplementary information to the tagset descriptions that DKPro Core usually adds to the document (i.e. which component/model was used to generate the annotations in a given layer and which tagset was used). A comprehensive recording of components and parameters is left to the workflow engine (i.e. UIMA) and/or to the executing environment. E.g. for building experiments, we sometimes use DKPro Lab, which also keeps track of configuration information of individual components.	Draft
GATE	8.2	Partial	A GATE pipeline contains all the information required to recreate it from scratch (which components, in which order, with which parameter settings). Multiple different configurations are not supported though so changing a parameter to carry out an experiment would require saving a new copy of the pipeline.	Draft
ILSP	1.2.1	No	Components only add annotations regarding original source (typically a text file on disk).	Draft

23. The Workflow Engine should permit Licence Aggregation in Workflows

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3,WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated because it is considered to be a functional requirement of the workflow manager rather than an interoperability requirement.

A workflow must aggregate all licences either from the components which is composed of, or from the language resources/knowledge bases that are used withing it and inform the user about them. (Requirement extracted from Interoperability scenario WG3-2)

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	No licence information handled in workflows	Draft
ARGO	0.5	Full	A user will be shown a full list of licenses for a workflow when attempting to run it for the first time after any component changes are made to the workflow, to which they must agree before the workflow can be executed.	Draft
DKPro Core	1.8.0	No	Presently, DKPro Core does not record license information in processed documents.	Draft
GATE	8.2	No	GATE currently doesn't record any license information at all	Draft
ILSP	1.2.1	No	Components do not add license information to processed documents.	Draft

24. Using/treating workflows as components

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. It is considered a platform functionality requirement (WP6).

In order to incorporate an entire workflow (not only a single component) which may potentially be using a specific workflow engine/implementation or flow configuration that cannot be imported into my workflow editor of choice, it must be possible to treat the whole workflow as a component applying encapsulation and information hiding. However, it includes the ability to configure individual components in the workflow, e.g. by exposing these parameters as workflow parameters.

NOTE	Relevant approaches are e.g. the UIMA Aggregate Analysis Engines or GATE Applications.

Source: WG 4 Scenario 1 — Transferability of components between ecosystems

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Full	The workflows are special components, they can be used in other workflows and Alvis offers means to handle the internal parameters.	Draft
ARGO	0.5	No	At present Argo doesn't allow this. There is some support for this within the execution engine, however it hasn't been fully tested and changes within the UI will be required. Current support within the execution engine would limit distribution of an embedded workflow to a single node, as there is no ability to distribute its components.	Draft
UIMA	2.8.1	Full	UIMA supports the concept of **aggregate analysis engines** whose delegates can be either primitive analysis engines or further aggregate engines. Aggregate engines may declare parameters whose values can be forwarded to delegates.	Draft
DKPro Core	1.8.0	N/A	This is out of the scope of DKPro Core. See UIMA.	Draft
GATE	8.2	Full	In GATE every application is also a component allowing arbitrary nesting of applications.	Draft
ILSP	1.2.1	Partial	See UIMA for UIMA-based ILSP components.	Draft

25. Incorporation of multiple resources in parallel

Concreteness: *abstract*
Strength: *recommended*
Category: *WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated per discussion in WG4. If multiple resources of a type need to be added to a workflow, simply add the same component multiple times, once configured for each resource in question. Engineering components in general such that they support using multiple resources will impose an unnecessary complexity on them and was considered over-engineering.

Whenever possible, components should allow to incorporate multiple resources in parallel, e.g. looking up a word from multiple sources or using multiple classification models in parallel. Examples:

- Stanford CoreNLP named entity recognizer can load multiple NER models in parallel
- a Gazetteer should be able to access not only a single knowledge resources, but be able to access e.g. multiple SparQL endpoints

Components may choose or have to disambiguate results if there are multiple hits, e.g. using a relevance or confidence score, in particular if the annotation scheme being used does not allow for multiple categories to be assigned to a single element (e.g. a Named Entity) or does not allow multiple elements to exist at the same location.

Another alternative would be to add the same component multiple times to a workflow, each time using a different resources - this would require that at least temporarily multiple categories/annotation elements would be allowed. A subsequent component could be used to select the most relevant one(s).

Caveat: Having relevance/confidence scores from different sources bears a great risk of the scores being on different scales and not comparable.

NOTE	parallel access to resources could be externalized into a resource adapter, such that the component itself would not have to deal explicitly with handling multiple adapters in parallel.

Source: WG 2 Scenario 3 — The relation between documents and knowledge bases through keywords

See also

- 19. Components that use external knowledge resources should delegate access to a resource adapter instead of handling it themselves

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Not supported	Draft
ARGO	0.5	No	It is possible to model the alternative suggestion using Argo (i.e. using multiple instances of the same component but applying different resources to each one) however there is no subsequent component to determine the most relevant output. Not sure how this would be implemented at a framework level; seems very use-case specific.	Draft

Product	Version	Compliant	Justification	Status
DKPro Core	1.8.0	No	The most prevasively used mechanism to load models in DKPro Core is the ResourceObjectProvider controlled through the parameters **PARAM_MODEL_LOCATION**, **PARAM_LANGAUGE**, and **PARAM_VARIANT**. This is presently set up in a way that only a single value is permitted for each of these parameters. Hence, e.g. the CoreNlpNamedEntityRecognizer would have to be added multiple times to a pipeline in order to be used with multiple models. The preference resolution mechanism that is normally part of the CoreNLP NER component would have to be modelled as a separate pipeline component (which is currently not offered by DKPro Core). In a few cases, components require multiple models (e.g. OpenNlpSegmenter) in which case we currently use multiple parameters, e.g. **PARAM_TOKENIZATION_MODEL_LOCATION** and **PARAM_SEGMENTATION_MODEL_LOCATION**. A few older components not using the ResourceObjectProvider may support multiple resources to be loaded, e.g. the StopWordRemover component.	Draft
GATE	8.2	Unknown	you could write a GATE component in this way, but it isn't supported directly by the framework and I know of none that do this	Draft
ILSP	1.2.1	No	Not supported in a generic way.	Draft

26. It should be possible to determine the source of an annotation/assigned category

Concreteness: *abstract*
Strength: *recommended*
Category: *WG4*
Status: *final*

It should be possible to trace back which component produced an annotation element or assigned a category and which knowledge resources may have been involved in the process. This does not entail that every annotation element must have explicit provenance metadata. The requirement could also be solved by assuming that all annotations of a certain type or within a certain view were created by one component and providing just provenance at the level of the type or view.

The ability to trace back could e.g. be relevant in order to perform a post-hoc linking/disambiguation of categories obtained from different knowledge resources or to adjust/harmonize relevance/confidence scores based on different scales.

NOTE	DKPro Core includes tagset information into the annotated document (in most cases even for tags that were not seen, but present in the model). U-Compare had strong support for provenance. Some annotation schemes have support for relevance/confidence scores, but others do not.

Source: WG 2 Scenario 3 — The relation between documents and knowledge bases through keywords

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	The creator of an annotation is identified but no trace about the involved resources or annotation categories.	Draft
ARGO	0.5	No	There is no information output from an execution that identifies which components produced individual annotations.	Draft
DKPro Core	1.8.0	Partial	Most DKPro Core components add tagset information. This mechanism currently assumes that only a single component in a workflow creates annotations of a given layer (i.e., the information is maintained at the level of layers, not at the level of individual annotations). Further, the mechanism requires that the tagset information can actually be extracted from the model being used.	Draft
GATE	8.2	Partial	Some existing components (specifically the JAPE transducer) add information to annotations about what created them, but currently this isn't standard across all components, although I think I can see a way of doing so.	Draft
ILSP	1.2.1	No	No relevant annotation added.	Draft

27. Components should handle failures gracefully

Concreteness: *abstract*
Strength: *recommended*
Category: *WG4*
Status: *final*

In the event that a component fails (e.g. due to a loss in network connectivity when writing to a database), then the component should ensure that there are no side-effects (e.g. database in an inconsistent state) which prevent the component from being re-ran at a later date, from the same point as which it failed. This would be required if the execution platform were to offer checkpointing.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Alvis however can re-run a component from a data dump.	Draft
ARGO	0.5	Partial	There is no framework support for this (e.g. when an exception occurs in a component, a rollback/onError callback could be made on the component to handle any issues), however when developing components for Argo we always try to ensure that this requirement is implemented. There aren't many components in Argo, if any at all, that will produce unwanted side-effects on the occurrence of an exception.	Draft
DKPro Core	1.8.0	N/A	DKPro Core does not provide any workflow functionality. It relies on such functionality to be provided by a workflow engine. Components are usually implemented fail-fast: as soon as a problem occurs, an exception is thrown. How the workflow engine (or application embedding DKPro Core components) handles these is up to them.	Draft
GATE	8.2	No	currently if a component fails (in a way not expected by the developer) than the exception will propagate back up to the workflow and halt execution.	Draft
ILSP	1.2.1	No	Not supported.	Draft

28. Processing components should be downloadable

Concreteness: *abstract*
Strength: *recommended*
Category: *WG4*
Status: *final*

Processing components should be downloadable. The user should be able to choose on which infrastructure to run them, e.g. to ensure that the processed data does not leave a particular institution. Components that are not downloadable should clearly indicate that (e.g. using a "I am a service" flag).

Source: WG1 Scenario 2 — SME running research analytics for funders within the European Research Area

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Full	AlvisEngine and its components can be downloaded. Docker images for Alvis in progress.	Draft
ARGO	0.5	No	Currently workflows and components only exist within the Argo platform.	Draft
DKPro Core	1.8.0	Full	All released DKPro Core components come as downloadable Maven artifacts and are distributed via Maven Central.	Draft
GATE	8.2	Full	components are created by loading a plugin from a given URL. This results in the jar files being downloaded to the local machine in some way (dependent on a number of factors) and then added to the JVM. This means all processing occurs where the framework is being run and data does not have to leave the local machine.	Draft
ILSP	1.2.1	No	Processing components are available only as services.	Draft

29. The actual content of all content resources must be discoverable

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG2*,*WG3*
Status: *deprecated*

CAUTION	This requirement has been deprecated in favor of 4. URL to actual content must be discoverable.

The URL of the actual content of a resource, e.g. a text file, PDF file, or other common data format, must be derivable from readily discoverable data. It must either be included in the metadata or it must be reliably constructable from the metadata URL, e.g. by appending or changing a suffix; derivation rules must be clearly documented.

NOTE	This requirement refers only to content resources that can be used as input for a TDM process (i.e. corpora of publications etc.), or to knowledge resources (e.g. annotation schemas, typesystems, grammars etc.) that are used as ancillary resources for the operation of TDM components. The requirement is set for the proper operation of the TDM components in so far as they need to easily access the resources they operate on or with.

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Actual content (either pdf, text, or other) is available via constructing URL based on identifiers (internal CORE id, oai identifier, doi) provided in the metadata record	Final
OpenAIRE	Jun-16	Partial	There's a link to the doi but not always; and usually the doi sends to a landing page, but there are some rules that can be used for "guessing" the download link	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Full	From XMLS, the URL can be reliably constructed using the DOI.	Final
TheSoz	Jun-16	Full	Url to the resource and the SPARQL endpoint is provided in the void.ttl file (however file is not at the moment available online due to technical issues).	Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	Partial	although the URL to the JATS schemas (as DTDS, Relax NG & XSD) and elements is available at stable URIs, each at a separate folder (info is provided at https://jats.nlm.nih.gov/files.html), there is no formal metadata to indicate this	Final
OLiA	Jun-16	Full		Final
LAPPS	Jun-16	Full		Final

30. Metrics for the confidence level of the TDM operation should be included in the metadata

Concreteness: *abstract*
Strength: *optional*
Category: *WG1*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated, as it is premature and requires defining evaluation framework first.

It is important to include metadata stating the confidence level of the TDM operation in order to help users select the appropriate components.

NOTE	This requirement is considered premature; although it is important, we need first to define the appropriate metrics, methods and tools and perform evaluation on components in an objective way and then add it to their metadata.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Alvis does not provide this aspect. Alvis does not yet deal with metadata.	Final
ARGO	0.5	No	Argo components don't have any specific metadata relating to confidence levels that will help users selecting components when developing a workflow. We would expect this information to reside within the main textual description of a component, however none of the existing Argo components provide this. There are a number of Argo components that produce confidence level values against annotations they produce during a workflow execution, however this wouldn't appear to directly relate to this requirement.	Final
DKPro Core	1.8.0	No	The confidence/quality of models (NOT necessarily tools) is typically very much depending on the processed data. It would be a lot of effort to provide such information. The main problem, however, is that there is not a wide range of free suitable gold standard corpora that could be used to generate such information.	Final
GATE	8.2	Partial	Confidence level is available for tools where this is appropriate. In some cases, e.g. a tool for converting data from one format to another, it is not appropriate	Final
ILSP	1.2.1	Partial	Some ILSP tools provide confidence level attributes for annotations, e.g. named entities.	Final

31. Metrics for the performance of the TDM operation should be included in the metadata

Concreteness: *abstract*
Strength: *optional*
Category: *WG1*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated, as it is premature and requires defining evaluation framework first.

It is important to include metadata related to the performance of TDM operation the appropriate components.

NOTE	This requirement is considered premature; although it is important, we need first to define the appropriate metrics, methods and tools and perform evaluation on components in an objective way and then add it to their metadata.

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Alvis does not provide this aspect.	Final
ARGO	0.5	No	No performance information given by existing components	Final
DKPro Core	1.8.0	No	See above	Final
GATE	8.2	Partial	Performance is sometimes available for specific tools over specific collections, but not generally. Quoting performance would also need us to quote the conditions under which it was obtained	Final
ILSP	1.2.1	No	Not supported	Final

32. Version must be included in the metadata description for all resources

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG2*,*WG3*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. It is covered by 6. Components should have a unique identifier and a version number and 102. Adding version information in the metadata descriptions of all resources.

Version information must be included in the metadata of all resources.

See also

- 6. Components should have a unique identifier and a version number
- 102. Adding version information in the metadata descriptions of all resources

NOTE	At this point, we only wish to check whether versioning is represented in the metadata, and the recommended action can be to at least add the last update date; however, to ensure interoperability, we need to create guidelines for defining the notion of "version" for the same resource type and promote standardisation of its representation, at least per resource type; we also need to distinguish between versions of the metadata records and versions of the resources themselves

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Metadata format harvested at the moment (oai_dc) does not contain such information. However future integrations of other md formats (such RIOXX / OpenAIRE) contain this information and if provided (from the source repository) then it shall be captured by CORE (and therefore propagated from there to OMTD)	Final
OpenAIRE	Jun-16	No	not in the metadata of the original data provider	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Partial	Licensing and version is included in the body of the article. For supplementary materials, most recent version is used. Frontiers has not published an article using tools.	Final
TheSoz	Jun-16	Full	owl:versionInfo in void.ttl file	Final
Agrovoc	21/01/2016	Partial	modification date (dcterms:modified) is available	Final
JATS	1.1	Full	Provided in the schema	Final
OLiA	Jun-16	Partial	no version id, but a textual description of the resource's evolution.	Final
LAPPS	Jun-16	No		Final

Product	Version	Compliant	Justification	Status
Licences	Jun-16	Full	Reasons of (legal) clarity and certainty require the most clear and reliable information to be attached to the resource/s	Final
Alvis	0.5rc	No	Alvis does not provide this aspect.	Final
ARGO	0.5	No	Argo components have versions specified in both their POM and UIMA XML files, however these are usually not updated by developers, are not guaranteed to be synchronised and are not exploited by the Argo system. For example, an Argo instance will only ever use the latest version of a component when executing a workflow.	Final
DKPro Core	1.8.0	Full	Version is included in the POM file and in the UIMA XML descriptors for components and in the POM file and properties file for each model.	Final
GATE	8.2	No	Planned	Final
ILSP	1.2.1	Partial	Only for components integrated in infrastructures like CLARIN and METASHARE. For example https://goo.gl/yDynbu	Final

33. Licensing information must be included in the metadata

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG3*
Status: *final*

All resources must be accompanied with a clear indication of their licensing terms; this can be indicated by the licence name, reference to a url or document that is accessible to the user and, if possible, a user-friendly summary.

NOTE	For this version, we will allow also "open access" or "other" etc. as statements in the metadata; in the next versions, this needs to be standardised, promoting licence values. It would also be preferable to attach the full document of the licence for (legal) certainty reasons

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Corpus coming from Open Access repositories, explicit licensing information included where available	Final
OpenAIRE	Jun-16	Partial	All publications are harvested from open access sources, but licence values are not always present in the metadata of the original data provider	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Full	CCBy Licensing and version is included in the body of the article.	Final
TheSoz	Jun-16	Full	`dc:license` in `void.ttl` file	Final
Agrovoc	21/01/2016	Full	http://aims.fao.org/aos/agrovoc/void.ttl	Final
JATS	1.1	Partial	although not in the metadata, the documentation includes a faq stating "The Standard (both PDF and HTML versions) is copyrighted by NISO, but all of the non-normative information found on this site is in the public domain. That includes all of the schemas and the Tag Libraries. The Tag Sets may be used freely and without permission from either the NLM or NISO."	Final
OLiA	Jun-16	No	Not yet; plan to be released under a Creative Commons Attribution Sharealike licence as soon as a reference publication has appeared	Final
LAPPS	Jun-16	No		Final
Licences	Jun-16	Full	Only for standard public licences	Final
Alvis	0.5rc	No	Alvis does not provide this aspect but some licence information are provided as free texts in the tool descriptions or in README of some tools.	Final

Product	Version	Compliant	Justification	Status
ARGO	0.5	Partial	Licensing information regarding components can be included within the Argo Descriptor File, however the majority of existing components within Argo do not have this information. When available, it's in free text.	Final
DKPro Core	1.8.0	Partial	Licensing information for our code is included in the POM file. Licenses for transitive dependencies can be reached (where available) by traversing the dependency hierarchy and inspecting the POM files of the dependencies. For models, we have no license information most of the time.	Final
GATE	8.2	Partial	The tool metadata does not provide licensing information, but license information is distributed with the tool	Final
ILSP	1.2.1	Partial	Only for components integrated in infrastructures like CLARIN and METASHARE. For example https://goo.gl/yDynbu	Final

34. Licensing information should be expressed in a machine-readable form

Concreteness: *abstract*
Strength: *recommended*
Category: *WG1*,*WG3*
Status: *final*

Licensing terms should be expressed in a machine-readable form so that the s/w can automatically compute permitted uses (cf. REL & CC-like licensing elements).

Product	Version	Compliant	Justification	Status
Licences	Jun-16	Partial	A few standard licences are available in machine-readable form, in RDF (CC-REL/ODRL) but they are not all officially recognised; additionally, for licences of DK-PRO Core, there is usually an URL reference to the license text in the POM, which is used by the Maven SPDX plugin to generate an RDF description of the licenses.	Final
DKPro Core	1.8.0	Partial	Licensing information for our code is included in the POM file. Licenses for transitive dependencies can be reached (where available) by traversing the dependency hierarchy and inspecting the POM files of the dependencies. For models, we have no license information most of the time.	Final

35. All resources must include a unique persistent identifier

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG2*,*WG3*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. It is covered by REQ-6 and WG1-8.

All resources must include an identifier (e.g. PID, DOI etc.) that allows them to be uniquely, non-ambiguously and persistently identified.

See also

- 6. Components should have a unique identifier and a version number
- [WG1-8]

NOTE	for this version, we only need a unique identifier, of whichever scheme the provider already uses; for next versions, we should recommend a scheme

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Full	Offer a list of identifiers (internal CORE id, source's OAI identifier and where available or resolved via external service a DOI)	Final
OpenAIRE	Jun-16	Partial	We keep the original OAI identifier and all identifiers harvested from the metadata of the original data provider; but the identifiers may not always be obligatory	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Full	DOI has an identifier. All resources follow PMC naming convention.	Final
TheSoz	Jun-16	No		Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	Full	see point 1	Final
OLiA	Jun-16	Full		Final
LAPPS	Jun-16	Full		Final
Licences	Jun-16	Full	For standard licences, the url can also be used; This would meet the demand for clear and stable locus of the resource/s	Final
Alvis	0.5rc	Partial	A local identification of the resources is done.	Final
ARGO	0.5	Full	All components and type systems have unique identifiers.	Final

Product	Version	Compliant	Justification	Status
DKPro Core	1.8.0	Full	Components can be uniquely identified using their corresponding (<groupId>, <artifactId>, <versionId>) tuple	Final
GATE	8.2	Full	Fully qualified class name of processing resources is unique.	Final
ILSP	1.2.1	Partial	Only for components integrated in infrastructures like CLARIN and METASHARE. For example https://goo.gl/yDynbu	Final

36. Classification metadata should be included, where applicable, in the metadata record of the resource

Concreteness: *abstract*
Strength: *recommended*
Category: *WG1*,*WG2*
Status: *final*

It is highly recommended to include classification, as applicable, to resources (e.g. domain, text type, genre etc.); the preferred form should be in accordance to one of the recommended controlled vocabularies/authority lists; if not, a link to a URL that contains the list of values used for the specific classification should be included; in all cases, the source of value for the classification must be properly indicated (through the name or link to the vocabulary) and available to the user; mappings between the controlled vocabularies are recommended.

NOTE	For this version, we include in the metadata what exists in the resource metadata; for next versions, we promote standardisation through at least the marking of the classification scheme. It is also important to distinguish between the various classification information, e.g. subtypes of documents or tools vs. domain of documents and domain covered by a tool.

See also

- 40. Component metadata must include standardised categories/tags that make them easy to discover

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Very limited classification metadata: only what the original metadata record offers (keywords, topics, or others that are usually not part of a defined vocabulary)	Final
OpenAIRE	Jun-16	Partial	Mainly keywords from the original metadata record; but this is not always present; in some subsets we have also used topic classification algorithms	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Partial	Frontiers is in the middle of deploying an XML based on JATS. i.e. Article types are mapped to JATS. Domain, field, specialty, and taxonomy used is Frontiers defined.	Final
TheSoz	Jun-16	Full	via DbPedia URIs. The following values are present in the current `void.ttl`: `dc:subject` http://dbpedia.org/resource/Social_sciences ; `dc:subject` http://dbpedia.org/resource/Thesaurus ;	Final
Agrovoc	21/01/2016	Full	DbPedia URIs	Final
Alvis	0.5rc	No	Not yet	Final
ARGO	0.5	No	No metadata regarding component classification, however the name of a component is a generally a good indicator	Final

Product	Version	Compliant	Justification	Status
DKPro Core	1.8.0	Partial	Tool classification is recorded in the documentation (e.g. at https://dkpro.github.io/dkpro-core/releases/1.7.0/components/) but not in the metadata. However, the documentation is automatically created (since 1.8.0) and a classification can be automatically derived from the component names.	Final
GATE	8.2	Partial	Tool classification is partially described by naming conventions	Final
ILSP	1.2.1	No	Not supported	Final

37. Information on the structural annotation (layout) of resources should be included in the metadata of the resource

Concreteness: *abstract*
Strength: *recommended*
Category: *WG1*
Status: *final*

Resources, such as publications & text files from corpora, that are annotated at the structural level (i.e. are tagged as to elements of their internal structure) should include in their metadata a reference to the annotation scheme used (e.g. TEI, JATS etc.), through a name or link to a url

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Structure of resources not included as part of metadata record but described separately	Final
OpenAIRE	Jun-16	Partial	If the resource is included, it may be PDF, HTML or Word, but without any structural encoding	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Full	Article metadata include references to the annotations use. Frontiers uses NLM DTD.	Final

38. Access mode of resources must be included in the metadata

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG2*,*WG4*
Status: *final*

Access mode of all resources (e.g. for s/w, whether they are downloadable or accessible as web services/workflows & for L/KRs through i/f, SPARQL endpoints etc.), must be included in the metadata

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Full	Resources are available, accessible and downloadable by default. If resources are not available is explicitly stated in a separate field (marked as deleted/disabled)	Final
OpenAIRE	Jun-16	Partial	If the resource is available, it can only be downloadable	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Full	For researchers, the page includes access to the resources (PDF, ePUB, XML, readcube). For crawlers, the FTP site to access resources can be provided upon request.	Final
TheSoz	Jun-16	Full	SPARQL endpoint and URI lookup endpoint in `void.ttl` file	Final
Agrovoc	21/01/2016	Full	void:sparqlEndpoint, void:dataDump	Final
JATS	1.1	No		Final
OLiA	Jun-16	No		Final
LAPPS	Jun-16	No		Final
Alvis	0.5rc	No	No explicit and formal specification of access mode	Final
ARGO	0.5	Partial	Components can be declared as native to Argo (allowing them to be executed within the Argo execution framework) or as web services, however there are also existing native components that simply wrap web services, so these types are not always mutually exclusive.	Final
DKPro Core	1.8.0	Full	All of the components are portable software packages available via maven artifact repository	Final
GATE	8.2	Full	All components are downloadable packages	Final
ILSP	1.2.1	Full	Only for components integrated in infrastructures like CLARIN and METASHARE. For example https://goo.gl/yDynbu	Final

39. Content resources must include metadata on their format (e.g. XML, DOCX etc.)

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*
Status: *final*

Content resources must include metadata on their format (e.g. XML, DOCX etc.), preferrably in accordance to the IANA mimetype; where applicable, further information on the specific type of the format (e.g. XML compatible with a specific schema XSD) should be included; when the resource is the output of an OMTD operation, the metadata should automatically receive the appropriate format values

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Resources contain their format type in their response Content-type header. Not included in the metadata record	Final
OpenAIRE	Jun-16	No	Various formats (doc, pdf, html etc.) but not specified in the metadata	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Full	Available in XML in NLM 3.0, available as JATS. Also available in ePUB	Final

40. Component metadata must include standardised categories/tags that make them easy to discover

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. It is covered by REQ-8.

Component (web services & workflows but also of tools) classification & description must include metadata that describe the task they perform (e.g. annotation, named entity recognition), preferably in the form of a standard(ised) vocabulary

See also

- 36. Classification metadata should be included, where applicable, in the metadata record of the resource
- 8. Components must associate themselves with categories defined by the OpenMinTeD project

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	No metadata	Final
ARGO	0.5	Partial	Components can be grouped by type (readers,, writers, analytics, web services) and by the type systems they use.	Final
DKPro Core	1.8.0	Partial	Tool classification is recorded in the documentation (e.g. at https://dkpro.github.io/dkpro-core/releases/1.7.0/components/) but not in the metadata. However, the documentation is automatically created (since 1.8.0) and a classification can be automatically derived from the component names.	Final
GATE	8.2	No		Final
ILSP	1.2.1	Partial	Only for components integrated in infrastructures like CLARIN and METASHARE. For example https://goo.gl/yDynbu	Final

41. Content resources must include metadata on their language(s)

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG2*
Status: *final*

Content resources including language, knowledge and text files must include metadata on their language(s)

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Described only the metadata record, not for each content resource separately	Final
OpenAIRE	Jun-16	Partial	If in the metadata of the original data provider, the document language is specified	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Full	Meta-text available for language. Other fields defined.	Final
TheSoz	Jun-16	Full	via language attribute for each entity in the thesaurus	Final
Agrovoc	21/01/2016	No		Final
JATS	1.1	No	Only in English as far as I have checked	Final
OLiA	Jun-16	No	Language only indirectly deducible from the description of the resource in the data type property 'comment' value.	Final
LAPPS	Jun-16	No	No	Final

42. The metadata can include the information on which projects/workflows involve the resource

Concreteness: *abstract*
Strength: *optional*
Category: *WG1*,*WG2*,*WG3*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement is deprecated. It should be added back as a functional requirement for the workflow execution service (WFEX). The functional requirement should state that the registry should be able to collect and display all resources (e.g. workflows, components, whatever) that use a given resources (e.g. a component, model, KR, etc.). This information must be dynamically collected and displayed in the registry through a query. It cannot be part of static metadata.

The metadata can include the information on which projects/workflows involve the resource, thus allowing to state resource "popularity", increase reliability, bring TDM outputs to a better interoperability.

NOTE	For use inside OMTD, this can be moved to the platform functions; for use out of OMTD, this can be left to the providers/users, so we can at best recommend it. This information can be used for evaluation purposes.

See also

- 49. Metadata of tools should contain information about the models available for them

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Only if source's includes this information - core covers RIOXX's format (which offers this information - OpenAIRE equivalent for UK) that only a small subset of UK repositories is compliant to (and UK repos is only a small subset of the global coverage so essentially a very minor portion of the whole corpus)	Final
OpenAIRE	Jun-16	Partial	Only as regards funding	Final
TheSoz	Jun-16	No	Unknown	Final
Agrovoc	21/01/2016	No	Unknown	Final
JATS	1.1	No	Unknown	Final
OLiA	Jun-16	No	Unknown	Final
LAPPS	Jun-16	No	Unknown	Final
Licences	Jun-16	Unknown	Unknown	Final
Alvis	0.5rc	No	No metadata	Final
ARGO	0.5	No	This would be fairly simple to implement on a per-instance basis of Argo; all of the data required to conform to this requirement is held within an SQL database used by Argo.	Final

Product	Version	Compliant	Justification	Status
DKPro Core	1.8.0	No	It would be unfeasible to maintain a list of which project/workflows make use of our components. We do not know who is using the software and for which purposes.	Final
GATE	8.2	No	Unknown	Final
ILSP	1.2.1	No	Unknown	Final

43. S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG4*
Status: *final*

S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Not supported	Final
ARGO	0.5	No	UIMA descriptors allow the definition of languages supported by a component, however Argo doesn't make use of this information at present and currently no Argo components declare this metadata.	Final
DKPro Core	1.8.0	Partial	Most DKPro Core components take a model as a parameter. The model then usually determines the language. Also for components that are locked to a specific language, this information is currently not included in the component metadata.	Final
GATE	8.2	No	Not currently provided	Final
ILSP	1.2.1	Partial	Only for components integrated in infrastructures like CLARIN and METASHARE. For example https://goo.gl/yDynbu	Final

44. Statistical metadata that allow monitoring of resource versions may accompany resources

Concreteness: *abstract*
Strength: *optional*
Category: *WG1*,*WG2*
Status: *final*

Resources must include statistical data (e.g. size of resource, total number of concepts, etc.) which allow monitoring of changes across versions; this will help measuring performance of components with different versions of the same resource.

NOTE	for version 1 this is set to optional, but for version 2 we should increase the strength for specific metadata elements

Product	Version	Compliant	Justification	Status
TheSoz	Jun-16	Partial	number of triples contained inside the resource and number of triples aligned with other resources (e.g. dbpedia)	Final
Agrovoc	21/01/2016	Full	number of concepts, labels	Final
JATS	1.1	No		Final
OLiA	Jun-16	No		Final
LAPPS	Jun-16	No		Final

45. S/W (tools, web services, workflows) must indicate format of their output

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG4*
Status: *final*

s/w must indicate the specific formats of their output (e.g. statistical data, annotated corpora etc.)

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Information is limited	Final
ARGO	0.5	Partial	Components can optionally declare what types they will output.	Final
DKPro Core	1.8.0	Partial	via @TypeCapability tags and also sometimes declared as javadoc. Mind that some components cannot clearly state the supported inputs/outputs until they actually process a document and have loaded the necessary models/rules. It may also be the case that despite declaring a certain input, the component may work without it or that despite declaring a certain output, the component may not actually produce it (e.g. because the user disabled the creation of the output or because a document simply contains no instance of the specific type)	Final
GATE	8.2	No		Final
ILSP	1.2.1	Partial	Only for components integrated in infrastructures like CLARIN and METASHARE. For example https://goo.gl/yDynbu	Final

46. Output resources of web services/workflows must be accompanied by provenance metadata

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement is deprecated. It should be added back as a functional requirement for the workflow execution service (WFEX).

Output resources of web services/workflows must automatically be accompanied by appropriate metadata that can be derived from the usage of the service/workflow (e.g. format, language etc.) & indicating the processing thereof (provenance).

NOTE	The mechanism of adding this information is a function of the OMTD platform, but the actual values of the elements that need to be enriched or added are dependent on the s/w, so this information must be present in the s/w metadata that describes the output

See also

- 22. The Workflow Engine Should Permit Saving Experimental Conditions in a Workflow

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Alvis can provide some information, for example data formats and logs	Final
ARGO	0.5	No	Not supported	Final
DKPro Core	1.8.0	Partial	Some information is recoded in the TagSetDescription, TagDescription and DocumentMetaData.	Final
GATE	8.2	No	Not supported	Final
ILSP	1.2.1	Partial	Only for components integrated in infrastructures like CLARIN and METASHARE. For example https://goo.gl/yDynbu	Final

47. Information on funding of resources may be included in the metadata

Concreteness: *abstract*
Strength: *optional*
Category: *WG1*,*WG2*,*WG3*,*WG4*
Status: *final*

Information on funding sources (e.g. projects, funders data etc.) may be included in the metadata of the resources

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	same as point 17	Final
OpenAIRE	Jun-16	Full	If the original provider includes it, we harvest it	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Partial	New articles have the source of funding, if authors declare (i.e. connection to fundref)	Final
TheSoz	Jun-16	No		Final
Agrovoc	21/01/2016	No	only its creator (FAO)	Final
JATS	1.1	No		Final
OLiA	Jun-16	No	Some ontologies reference the project in which the resource was created.	Final
LAPPS	Jun-16	No		Final
Licences	Jun-16	No		Final
Alvis	0.5rc	No	No funding information included in metadata	Final
ARGO	0.5	No	Not implemented as we have not foreseen a need for this. We would expect this type of information to exist in the textual description of a component. No plans to implement.	Final
DKPro Core	1.8.0	No	DKPro Core is a community project.	Final
GATE	8.2	No		Final
ILSP	1.2.1	No	Not a standard practice	Final

48. All resource metadata records must include a reference to the metadata schema used for their description

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG1*,*WG2*,*WG3*,*WG4*
Status: *deprecated*

NOTE	Deprecated because this is a requirement on each metadata schema rather than the actual resources

All resource metadata records must include a reference to the metadata schema used for their description.

NOTE	for provenance information, we need to know the metadata schema from which we are extracting/converting information.

Product	Version	Compliant	Justification	Status
CORE	Jun-16	Partial	Schema not described formally in .xsd format but as part of API documentation	Final
OpenAIRE	Jun-16	Full	OpenAIRE schema	Final
Frontiers	NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.1	Partial	XML provides the reference to the metadata schema. In a multiple metadata scenario, Frontiers provides the DOI of the related articles.	Final
TheSoz	Jun-16	Full	defined in the header	Final
Agrovoc	21/01/2016	Full	see rdf prefixes	Final
JATS	1.1	No	no metadata	Final
OLiA	Jun-16	Full	List of namespaces	Final
LAPPS	Jun-16	No		Final
Alvis	0.5rc	No	No metadata support	Final
ARGO	0.5	Partial	Argo components are defined using UIMA XML descriptors, which do have a schema reference, and with an optional Argo XML descriptor, which don't have a schema reference.	Final
DKPro Core	1.8.0	Partial	UIMA XML descriptors and POMs contain a schema reference. Ad-hoc DKPro Core-specific metadata is currently not governed by a formal schema.	Final
GATE	8.2	No		Final
ILSP	1.2.1	No	Not supported	Final

49. Metadata of tools should contain information about the models available for them

Concreteness: *abstract*
Strength: *recommended*
Category: *WG1*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement is deprecated. Neither the model creators nor the component creators may know of all models/components they are compatible with. We discussed to keep component metadata, the model metadata, and the linking between components and models separately. A suggestion is to rephrase that requirement as "resource providers (e.g. components) should provide a list of other resources (e.g. models) they are compatible with". The concrete format of such a compatibility list likely needs to be defined. Additionally, the requirement should be turned into a functional requirement stating that the registry should be able to collect and display all resources (e.g. workflows, components, whatever) that are **compatible with a given resources** (e.g. a component, model, KR, etc.). This information must be dynamically collected and displayed in the registry through a query. It cannot be part of static metadata. This is different from REQ-42. REQ-42 is about actually depending/making use of other resources. REQ-49 is about being compatible (i.e. potentially making use of) other resources.

Information about the models available for the tools (e.g. location, language, domain, etc) must be included in the metadata (c.f. https://dkpro.github.io/dkpro-core/releases/1.7.0/models/ as an example).

NOTE	This is important for the operation of the components; so, we need to standardize naming/reference to the models and add it to the metadata.

See also

- 42. The metadata can include the information on which projects/workflows involve the resource

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	No metadata support	Final
ARGO	0.5	No	Not supported and not planned. Would require a mechanism adding to Argo which allows users to register models against components.	Final
DKPro Core	1.8.0	Full	The POM of a tool contains references to the known available models. This information is can be used to automatically derive which models are available for which tools. We do this as part of our automatically generated documentation.	Final
GATE	8.2	No		Final
ILSP	1.2.1	No	Not supported	Final

50. Documentation references should be versioned

Concreteness: *abstract*
Strength: *recommended*
Category: *WG1*,*WG2*,*WG3*,*WG4*
Status: *final*

When there is a link to documentation, that documentation should be versioned and should be about the version of a component/resource which carries the reference metadatum. The link should not simply point at the latest version of the documentation which may or may not be relevant for the given component/resource version in question.

Product	Version	Compliant	Justification	Status
TheSoz	Jun-16	No	Documentation is not versioned (http://lod.gesis.org/thesoz/de/help.html)	Final
Agrovoc	21/01/2016	No		Final
JATS	1.1	No		Final
OLiA	Jun-16	No		Final
LAPPS	Jun-16	No		Final
Licences	Jun-16	Partial	Identifying the version of the associated documentation is optimal, but it still could be linked either to "any later version" (as long as it is specified that the version number in question applies to it and to any late version)	Final
Alvis	0.5rc	No	No version	Final
ARGO	0.5	No	There is currently no mechanism in place for linking changes in a version of a component with any links to its documentation.	Final
DKPro Core	1.8.0	Full	Documentation is versioned in parallel with the source code	Final
GATE	8.2	No		Final
ILSP	1.2.1	No	Not supported	Final

51. License should be attached

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

The licence (full legal document) must be attached to the resource, or be publicly available (e.g. URI to permanent web page).

The licence should be attached to the resource, since there is no guarantee that the permanent URI will not change in the future. Public projects such as CC, GNU, Apache, provide with reasonable certainty that the licence will remain publicly available in a permanent location. This has developed a standard, but for smaller projects and/or individual providers may not be a sufficient guarantee.

52. License information must be in metadata

Concreteness: *abstract*
Strength: *recommended*
Category: *WG1*,*WG3*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated as a duplicate of 33. Licensing information must be included in the metadata.

All resources must have a licence clearly indicated in the metadata description of the resource.

53. Licensor must be entitled to grant license

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

```
The licence must be applied by the right holder or an authorised party
```

54. Licensees should remain with a copy of the license

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

Researchers, data providers and repositories should retain a copy of the text of the licence.

55. Standard licenses should be used

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

Licences should be expressed in standard form (Public Licences such as CCPL, GNU GPL, etc). Specifically devised licences in fact create additional transactive costs in terms of compatibility.

56. License should be machine readable

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

The licence should be expressed not only in the full standard legal document, but also in a machine readable form, preferably using one of the standard Rights Expression Languages (RELs) and relevant metadata schemas and vocabularies.".

57. License should be understandable by non-lawyers

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

The licence should also be expressed in a statement that makes it easy for non lawyers understand the main rights and obligations (similar to CC “commons deed”).

58. TDM must be explicitly allowed

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

To be TDMable a licence must specifically allow Text and Data Mining for (commercial; non commercial; research only; any;) purposes.

Alternatively, [not sure how to express this condition] to be TDMable a licence must specifically identify the rights included in its scope and the uses permitted.

59. Right for (temporary) reproduction must be granted

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

In order to allow TDM a licence must permit the reproduction, at least temporary, of the content protected by (copyright, SGDR, related rights).

60. Boundary for derivative work must be clearly defined

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

The licence must clarify what activities can be done with the results of TDM which can qualify as a derivative work of the original protected work.

61. No restrictions on TDM results which are not derived works

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

The licence should not unduly restrict what can be done with results that are not derivative works of the original protected works.

62. World-wide and irrevocable license grant

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

To be TDMable a licence should not include restriction as to the geographical or temporal dimension of its validity.

The licence must be irrevocable in order to grant the maximum level of reusability.

63. License must qualify for Open Access rights

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

Open Access compliant licences must include in their scope all rights covered by copyright and related rights, such as reproduction, redistribution, communication to the public, making available to the public, public performance, translation, adaptation, etc.

64. License must qualify for Open Access uses

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

Open Access compliant licences must cover all subject matter related to copyright and related rights such as: performers' performances, broadcasts of broadcasting organisations, sound recordings (Rome), first fixation of a film (EU), SGDR (EU), and employ an open ended wording to include also additional (usually national or regional) related rights (e.g.: non original photographs, scientific editions of public domain works, typographical arrangements, etc).

65. License must qualify for Open Access must not restrict use in any way

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

Open Access compliant licences must must permit the exercise of the aforementioned rights by anyone for any purpose, without limitations regarding temporal or geographical dimensions, nor should the purpose of the use be limited to non commercial, research only, academic uses, and similar unduly restrictive wording.

66. License must qualify for Open Access may include attribution requirements

Concreteness: *abstract*
Strength: *recommended*
Category: *WG3*
Status: *draft*

Open Access compliant licences must can only features limitations connected with the requirement to acknowledge authorship (paternity) and to guarantee the integrity of the work.

67. Knowledge Resource Element Id

Concreteness: *abstract*
Strength: *recommended*
Category: *WG2*
Status: *final*

For the purposes of resource access and resource schema harmonization, schema elements in Knowledge Resources should be discoverable, identifiable and dereferencable/resolvable through a URI. This is in line with the Linked Data paradigm. If deemed necessary, multiple ids are allowed for the same knowledge resource element if these are linked by means of an equivalence relation in the registry.

Product	Version	Compliant	Justification	Status
TheSoz	June 2016	Full		Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	Full		Final
OLiA	June 2016	Full		Final
LAPPS	June 2016	Full		Final
Ontolex	June 2016	Full	Documentation is versioned in parallel with the source code	Final
CLARIN CCR	June 2016	Full		Final
schema.org	June 2016	Full		Final

68. Data Category Linking Vocabulary

Concreteness: *abstract*
Strength: *recommended*
Category: *WG2*
Status: *final*

Linking elements from different schemas can be modelled in various ways. The most straightforward is the set of coarse grained lightweight thesaural mapping relations expressed by SKOS (https://www.w3.org/2004/02/skos/). This is our preferred linking approach, in combination with the following Owl and RDF object properties: owl:sameAs, rdfs:subClassOf, and rdfs:subPropertyOf, owl:equivalentClass, and owl:equivalentProperty. The compliance assessment checks the use of mentioned linking vocabularies within the KRs themselves, requesting an answer to the question: does the resource contain any of these links?

Product	Version	Compliant	Justification	Status
TheSoz	June 2016	Full		Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	No		Final
OLiA	June 2016	Full		Final
LAPPS	June 2016	Full		Final
Ontolex	June 2016	Full	Documentation is versioned in parallel with the source code	Final
CLARIN CCR	June 2016	No		Final
schema.org	June 2016	Full		Final

69. Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.

Concreteness: *abstract*
Strength: *recommended*
Category: *WG2*
Status: *final*

Description

Product	Version	Compliant	Justification	Status
TheSoz	June 2016	Full		Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	Full	JATS is in XSD format, but each element can be accessed via a URI	Final
OLiA	June 2016	Full		Final
LAPPS	June 2016	Full		Final
Ontolex	June 2016	No		Final
CLARIN CCR	June-2016	No		Final
schema.org	1.2.1	No		Final

70. All KR content elements need to be added as text annotations within a TDM workflow.

Concreteness: *abstract*
Strength: *mandatory*
Category: *WG2*
Status: *final*

When working with a knowledge resource (KR), its relevant elements need to be added as text metadata. In other words, any text annotations created from information in the KR should be linked to the URI of the relevant KR element. This enables the TDM modules to work in a uniform manner with the informational content the KRs provide. This is in line with the Linked Data paradigm.

This is a methodological requirement an as such not suited for the compliance check.

71. The KR should be ingestible through a URI

Concreteness: *abstract*
Strength: *recommended*
Category: *WG2*
Status: *final*

The KR should be downloadable through a URI for ingest (possibly after adaptation) into the TDM workflow.

Product	Version	Compliant	Justification	Status
TheSoz	June 2016	Full	SPARQL endpoint	Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	Full	In XSD (and RELAX-NG & DTD formats) and accessible as three different schemas but the user has to select one of these.	Final
OLiA	June 2016	Full		Final
LAPPS	June 2016	Full		Final
Ontolex	June 2016	Full		Final
CLARIN CCR	June-2016	No		Final
schema.org	1.2.1	Full		Final

72. The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.

Concreteness: *abstract*
Strength: *recommended*
Category: *WG2*
Status: *final*

Standard formats enable standard ingestion into the TDM workflow.

Product	Version	Compliant	Justification	Status
TheSoz	June 2016	Full	RDF	Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	Partial	XSD/RELAX-NG/DTD	Final
OLiA	June 2016	Full		Final
LAPPS	June 2016	Full		Final
Ontolex	June 2016	Full		Final
CLARIN CCR	June-2016	Full		Final
schema.org	1.2.1	Full		Final

73. Stick to widely used data compression formats

Concreteness: *concrete*
Strength: *best practice*
Category: *WG4*
Status: *draft*

Compression algorithms are used to reduce the size of data. To avoid problems in working with such data (creating or consuming), only algorithms/formats should be used for which there is broad support across platforms and programming languages. There are:

- gzip (also known as gz)
- zip
- xz
- tar (including tar.gz)

Specifically to be avoided are commercial algorithms and their implementations, e.g.

- rar

74. Machine-readable metadata for UIMA components

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 1. Components must be described by machine-readable metadata

The UIMA framework defines a series of so-called **resource specifiers** contain metadata about components. The following types of components and associate resource specifiers are supported by OpenMinTeD:

Component	Resource specifier
Reader component	CollectionReaderDescription
Analysis engine / Writer component	AnalysisEngineDescription

It is expected that these descriptors can be located within the same JAR file that also contains the component they describe.

The descriptor file should be located next to the Java class file that implements the described component, as it is indicated by the `implementationName` element within the descriptor.

The JAR file must contain a locator file called `META-INF/org.apache.uima.fit/components.txt` which contains pointers to all component descriptor files within a JAR.

```
classpath*:my/organizations/domain/name/package/Component.xml
```

The locator file should be automatically generated using the `generate` goal of the uimaFIT Maven Plugin as described in the uimaFIT documentation.

There must be at most **one** resource specifier per component class. I.e. no two resource specifiers should use the same implementation class.

Table 1. Compliance assessment

Product	Version	Compliant	Justification	Status
ARGO	0.5	Partial	Argo components are described using standard UIMA XML Descriptor files, however the descriptor files are not required to be located next to the implementation class. Argo components don't contain the `META-INF/org.apache.uima.fit/components.txt` locator file as they are distributed as UIMA PEAR packages, which already contain this information.	Draft
DKPro Core	1.8.0	Full	Resource specifier files automatically generated using uimaFIT Maven plugin.	Draft
ILSP		Unknown	Unknown	Draft

75. Embedding UIMA component metadata into the source code

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 2. Component metadata have to be embedded into the component source code

The uimaFIT library provides Java annotations to embed most types of UIMA metadata as Java annotations directly into the source code of UIMA components. Particularly relevant for OpenMinTeD are the following uimaFIT annotations:

uimaFIT annotation	Description
`@MimeTypeCapability`	For reader components: defines the supported input media types; for writer components: defines the supported output media types. The media type should be an official IANA media type.
`@TypeCapability`	Defines the UIMA types that a component can consume or produce.
`@LanguageCapability`	Defines the languages a UIMA component supports. The language should be specified according to IETF BCP47.
`@ConfigurationParameter`	Identifies a component parameter, whether the parameter is mandatory/optional, and may define a default parameter value.

Component metadata other than the ones specified are automatically derived from the source code and generated into a UIMA resource specifier by using the uimaFIT Maven plugin goal `generate`, e.g. the parameter types, implementation name, etc.

The uimaFIT `@ResourceMetaData` annotation **must not** be used as it would interfere with the need of the Maven POM and the UIMA resource specifier metadata being synchronized.

Table 2. Compliance assessment

Product	Version	Compliant	Justification	Status
ARGO	0.5	Partial	Some Argo components make use of the uimaFIT annotations, although it is not a requirement of Argo for components to do so. It is planned that, at some point, we will convert all of our existing components to use uimaFIT annotations.	Draft
DKPro Core	1.8.0	Partial	uimaFIT annotations are used. Language codes are expected to be ISO 639-1. Many components use private media types because many supported formats do not have a corresponding media type.	Draft
ILSP		Unknown	Unknown	Draft

76. Separating UIMA metadata from the component

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 3. Component metadata must be separable from the component

Storing the component metadata as a XML resource specifier with in the same JAR file that also hosts the described component facilitates separating the metadata from the component.

Specifically, the JAR file can be downloaded and the resource specifier can be located within the JAR using `META-INF/org.apache.uima.fit/components.txt` without having to download any (transitive) dependencies. The resource specifier can be immediately used, e.g. for ingestion into the OpenMinTeD registry.

Mind that the uimaFIT annotations embedded in the component's class file cannot be used as easily. Extracting the metadata from the class file requires resolving the class file, which in turn requires having access to the transitive dependencies. This would entail a much larger download and would be impracticable for the OpenMinTeD registry crawler.

Table 3. Compliance assessment

Product	Version	Compliant	Justification	Status
ARGO	0.5	Full	UIMA resource specifiers are manually maintained by the resource developers.	Draft
DKPro Core	1.8.0	Full	UIMA resource specifiers are auto-generated using the uimaFIT Maven plugin.	Draft
ILSP		Unknown	Unknown	Draft

77. Unique identifiers and a versions for UIMA components

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. It has been merged with REQ-82 into REQ-96.

See also

- 6. Components should have a unique identifier and a version number

The unique identifier for a UIMA component is constructed from the following coordinates that are either obtained from the Maven POM or from the UIMA resource specifier.

Coordinate	Description	Source
groupId	Group ID of the Maven artifact containing the component	Maven POM
artifactId	Artifact ID of the Maven artifact containing the component	Maven POM
version	Version of the Maven artifact containing the component	Maven POM
packaging	**optional** Packaging of the Maven artifact containing the component	Maven POM
classifier	**optional** Classifier of the Maven artifact containing the component	Maven POM
class	Name of the class implementing the component	UIMA resource specifier

The identifier has the following forms depending on the presence of the optional coordinates:

UIMA component identifiers

```
uima-mvn:{groupId}:{artifactId}:{version}#{class}
uima-mvn:{groupId}:{artifactId}:{classifier}:{version}#{class}
uima-mvn:{groupId}:{artifactId}:{packaging}:{classifier}:{version}#{class}
```

Example

```
uima-mvn:my.organization.tdm:dictionary-tools:1.0.1#my.organization.tdm.DictionaryLookup
```

NOTE

The version is also available from the UIMA resource specifier. It must be in sync with the version specified in the Maven POM. The uimaFIT Maven plugin goal `generate` ensures this by determining the version in the generated UIMA resource specifier from the version specified in the Maven POM. The `overrideComponentVersion` and `componentVersion` settings in the uimaFIT MAven plugin must **not** be used. Neither must the uimaFIT `@ResourceMetaData` annotation be used to manually provide a version.

Table 4. Compliance assessment

Product	Version	Compliant	Justification	Status
ARGO	0.5	Unknown	Unknown	Draft
DKPro Core	1.8.0	Full	UIMA components are provided as Maven artifacts.	Draft
ILSP		Unknown	Unknown	Draft

78. Specifying input and output types of UIMA components

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 10. Components should specify the types of the annotations that they input and output

The input and output of UIMA components should be declared in the component's resource specified using type capabilities. Capabilities should be declared in the component's source code using the uimaFIT `@TypeCapability` annotation.

Example of input/output capability declaration using uimaFIT annotations

```
@TypeCapability(
  inputs = {
    "de.tudarmstadt.ukp.dkpro.core.api.segmentation.type.Token",
    "de.tudarmstadt.ukp.dkpro.core.api.segmentation.type.Sentence" },
  outputs = {
    "de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos.POS" })
```

It may not always be known what types a component produces or consumes. For example, the production/consumption of some types might be optional. Optional types should always be declared, even if they are not always produced/consumed.

The types being produced or consumed by truly generic components may depend on their specific parameter settings or on they models they are configured with (e.g. a rule-based transducer). Such components should not be expected to declare input/output types.

UIMA type capabilities only specify the names of input and output types, but not where to locate the type system information or which version of the type system is used.

OpenMinTeD may therefore check the dependencies of the Maven artifact containing the component for type system declarations using the `META-INF/org.apache.uima.fit/types.txt` locator mechanism.

NOTE	Searching for types in the transitive dependencies of a component requires downloading all the transitive dependencies. This is something we explicitly wanted to avoid in 3. Component metadata must be separable from the component.
NOTE	Annotation types for input and output resources can be mapped to the OMTD-SHARE element annotationLevel embedded in the inputContentResourceInfo and the outputResourceInfo respectively.

Table 5. Compliance assessment

Product	Version	Compliant	Justification	Status
ARGO	0.5	Partial	All components have the ability (within their UIMA XML descriptor) to declare their input/output types, however existing Argo components generally lack this information. This is something we will correct.	Draft
DKPro Core	1.8.0	Full	Input/output capabilities declared where possible.	Draft
ILSP		Unknown	Unknown	Draft

79. Documentation of UIMA components

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 12. Components should provide documentation describing their functionality
- 21. Configuration and parametrizable options of the components should be identified and documented

The documentation of UIMA components should be maintained as JavaDoc in the component source code. The uimaFIT Maven plugin goal `enhance` picks the documentation of components and parameters up from there and embeds it into the compiled class. The goal `generate` finally transfers the documentation into the automatically generated UIMA resource specifiers.

The **description** properties of the uimaFIT `@ConfigurationParameter` and of the `@ResourceMetaData` annotations should **not** be used.

Component documentation should be provided as JavaDoc on the component class itself.

Documentation and parameter constant for UIMA component parameters

```
/**
 * Part-of-Speech annotator using OpenNLP.
 */
public class OpenNlpPosTagger
    extends JCasAnnotator_ImplBase
{
```

Parameter documentation should be placed before a constant declaration which defines the parameters name as shown in the example below. This ensures that IDEs show the parameters and associated documentation when auto-completion is invoked on a UIMA component. It also allows the uimaFIT Maven plugin goal `enhance` to pick up the documentation and to automatically transfer it into the **description** property.

Documentation and parameter constant for UIMA component parameters

```
/**
 * Use this language instead of the document language to resolve the model.
 */
public static final String PARAM_LANGUAGE = "language";
@ConfigurationParameter(name = PARAM_LANGUAGE, mandatory = false)
protected String language;
```

Table 6. Compliance assessment

Product	Version	Compliant	Justification	Status
ARGO	0.5	Partial	Documentation is manually maintained within the UIMA resource specifiers.	Draft
DKPro Core	1.8.0	Full	Documentation is maintained as JavaDoc and transferred into UIMA resource specifiers using the uimaFIT Maven plugin.	Draft

Product	Version	Compliant	Justification	Status
ILSP		Unknown	Unknown	Draft

80. Common elements to represent/describe an executable workflow

Concreteness: *abstract|concrete*
Strength: *recommended*
Category: *WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. Its parent abstract requirement (REQ-18) should be covered by the functionality of the OpenMinTeD platform, namely Galaxy workflow editor.

A uniform workflow language allows to use the same elements of language to represent/describe a workflow (See * 18. Workflows should be described using an uniform language. For example, Common Workflow Language (CWL) is a workflow language used in many systems, for example Galaxy. Common parts to represent an executable workflow are the following:

Parts	Description
Main_desc	To provide information about the workflow.
Main_inputs	To define the inputs parameters of the workflow.
Main_outputs	To define the output parameters of the workflow.
Steps	To define the steps of the workflows.

`Main_desc` is the part of the workflow representation that gives information about the workflow. It contains the following fields:

Field	Description
name	Name of the workflow.
command	Define the command to use for invoking the workflow.
version	Version of the workflow.
label	A human readable description of the workflow.

`Main_inputs` and `Main_outputs` are parts of the workflow representation that respectively define the inputs and outputs `parameters` of the workflow. Each `parameter` contains at least the following fields:

Field	Description
id	to identify this parameter.
type	type of values this parameter accepts e.g. float, string, File ...
value	to specify a default value for this parameter, if exists.

`Steps` is the part of the workflow representation that defines the list of `modules` and the list of `dependencies` between modules of the workflow. Each `module` is defined with the following fields :

Field	Description
name	name of the module.
command	define the command to use for invoking this module.
inputs	define the inputs parameters of this module. See the `parameter` table for the fields.
outputs	define the outputs parameters of this module. See the `parameter` table for the fields.

The dependencies between the modules define the dataflow of the workflow. They tells what outputs are fed by inputs for execution. Each `dependency` is defined with the following fields:

Field	Description
module.output.id	Is the `id` of a `output` of a specific `module` in this workflow.
module.input.id	Is the `id` of a `input` of a specific `module` in this workflow.

NOTE All elements described are supported by CWL that is built on technologies such as JSON-LD for data modeling and Docker for portable runtime environments. Another part, partially supported by CWL, can be considered to add `controls` like loops, conditions to the workflow.

81. Embedding GATE component metadata into the source code

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 2. Component metadata have to be embedded into the component source code

GATE provides a number of Javaannotations that can be used to embed CREOLE metadata into the source of GATE components. The annotations relevant to OpenMinTeD are as follows:

Annotation	Description
`@CreoleResource`	used to annotate a class to show that it is a GATE resource. The type of resource (processing resource, language resource, etc.) depends upon the class hierarchy.
`@CreoleParameter`	used to annotate a bean setter method to indicate that the property defines a CREOLE parameter. A parameter can be provided with a default value using the `defaultValue` parameter of the annotation.

All other metadata is determined at runtime via inspection of the class.

Table 7. Compliance assessment

Product	Version	Compliant	Justification	Status
GATE	8.2	Full	Components can be fully specified through Java annotations	draft

82. Unique identifiers and a versions for GATE components

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *deprecated*

IMPORTANT This requirement has been deprecated. It has been merged with REQ-77 into REQ-96.

See also

- 6. Components should have a unique identifier and a version number

Each GATE component is a Java class and as such can be uniquely identified by it's classname. Components are to be distributed via Maven and versioning information will be recorded as part of the POM. Together these two sources provide the following information

Coordinate	Description	Source
groupId	Group ID of the Maven artifact containing the component	Maven POM
artifactId	Artifact ID of the Maven artifact containing the component	Maven POM
version	Version of the Maven artifact containing the component	Maven POM
class	Name of the class implementing the component	GATE CREOLE Metadata (derived from classname)

Table 8. Compliance assessment

Product	Version	Compliant	Justification	Status
GATE	8.2	Partial	Components are not yet distributed by Maven and as such versioning info is not available	draft

83. Documentation of GATE components

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*,*WG4*
Status: *draft*

See also

- 12. Components should provide documentation describing their functionality
- 15. Human readable information should be provided by each resource
- 21. Configuration and parametrizable options of the components should be identified and documented

GATE components and their parameters are documented in two ways; CREOLE metadata and JavaDoc comments.

It is possible to give each component and parameter a name and a human readable comment (used as a tooltip in the GUI) via the CREOLE metadata. Components can also be given a URL of more extensive documentation that the user can be redirected to. Combined with the way the GUI restricts configuration fields based upon automatically extracted metadata this documentation should be enough to aid regular users in configuring and using a component.

JavaDoc comments are used to provide more API style documentation of each component and parameter and it is envisaged that this information is useful to those trying to embed GATE within another application rather than using the GUI.

Table 9. Compliance assessment

Product	Version	Compliant	Justification	Status
GATE	8.2	Partial	While GATE provides ways of documenting components it is difficult to enforce this on all components especially those developed by 3rd parties	draft

84. Separating GATE metadata from the component

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 3. Component metadata must be separable from the component

CREOLE metadata can be completely defined using an XML file (called `creole.xml` located at the root of a GATE plugin), however, we don't recommend this as it is too easy for embedded metadata and the XML version to go out of sync. It is useful though to have a standalone copy of the metadata as extracting it from the class file requires resolving the class, which in turn requires having access to the transitive dependencies which would be inefficient in situations such as the OpenMinTeD registry crawler.

We recommend that during the build process a fully expanded copy of the metadata is produced and stored within the resulting JAR as `META-INF/gate/creole.xml` so that other systems, such as the OpenMinTeD registry crawler, can read the metadata directly without reference to any of the class files of the components or their transitive dependencies.

Table 10. Compliance assessment

Product	Version	Compliant	Justification	Status
GATE	8.2	Partial	While we provide the tools needed to produce the standalone metadata it is not currently part of the standard build process and so rarely happens. This will change with the next version of GATE.	draft

85. Unique identifier and version for components in the OMTD-SHARE schema

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *deprecated*

IMPORTANT This requirement has been deprecated. It has been replaced with REQ-96.

The OMTD-SHARE schema includes specific distinct metadata elements for identifier and version of all resources. More specifically:

Name	Element/Attribute	Type	Strength	Repeatability	Description
identifier	element	xs:string	mandatory	yes	Reference to a PID, DOI or any kind of identifier used by the resource provider for the resource; must obligatorily be used with the attribute *resourceIdentifierSchemeName*
resourceIdentifierSchemeName	attribute	open controlled vocabulary	mandatory	no	Reference to the resource scheme that has been used for the identifier (e.g. DOI, handle etc.); the list of values includes the most widespread schemes used for resources
schemeURI	attribute	URI	mandatory when applicable	no	Reference to a URI of the resource scheme that has been used for the identifier, providing more information about the resource scheme
version	attribute	xs:string	recommended	no	Any string, usually a number, that identifies the version of a resource

Components that already include an identifier and version from previous metadata records can map them to these elements.

NOTE	For components collected from Maven, the Maven id (groupId, artifactId, versionId) can be used with a reference to the Maven scheme (i.e. use the attribute *schemeURI* with the value"https://maven.apache.org/pom.html#Maven_Coordinates").

NOTE	There is an ongoing discussion about a unique Persistent Identifier (PID) and the procedure for assigning it which may lead to further specifications.

See also

- 6. Components should have a unique identifier and a version number
- 32. Version must be included in the metadata description for all resources
- 35. All resources must include a unique persistent identifier
- 77. Unique identifiers and a versions for UIMA components
- 82. Unique identifiers and a versions for GATE components

Product	Version	Compliant	Justification	Status
Alvis		Unknown	Unknown	Draft
ARGO	0.5	Unknown	Unknown	Draft
DKPro Core	1.8.0	Unknown	Unknown	Draft
GATE	8.2	Unknown	Unknown	Draft
ILSP		Unknown	Unknown	Draft

86. Attaching format properties to the description of the inputs and outputs that use file

Concreteness: *concrete*
Strength: *recommended*
Category: *WG1*,*WG4*
Status: *deprecated*

IMPORTANT	This requirement has been deprecated. It is not necessary, as the associated abstract REQ-45 is implemented by REQ-88.

See also

- 45. S/W (tools, web services, workflows) must indicate format of their output
- 88. Embedding output format in UIMA component metadata
- 72. The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.

Some components (tools) will produce and consume files. To reduce mismatches and help to properly handle the files, the following properties/features about the `file formats` are relevant (to attach to the description of the inputs/outputs that use `file`) :

Feature	Strength	Description
id	required	identify the format
label	optional	label or name to display
mimeType	optional	mimeType as described in the metadata (e.g., ms:mimeType).
encoding	optional	identify the character encoding as described in the metadata (e.g., ms:characterEncoding)
parents	optional	formats from which this format is derived from, that allows to maintain a hierarchy of the formats. (Galaxy administrators do manage data types based on a hierarchy of formats)
doc_url	optional	link to the complete documentation provided in metadata (e.g., ms:documentationURL)

The features are normally present in the metadata (e.g., ms:dataFormat) and in the embedded annotations (e.g., java annotations). To simplify, we can suppose that `id` is the only required element, even if all may be important for the users (e.g., to identify a specific file), the developers (e.g., to understand a file for implementation) or tools (e.g., to automatically process a file).

87. Embedding language capability in UIMA component metadata

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 43. S/W (tools, web services, workflows) must indicate whether they are language-independent or the language(s) of the resources they take as input and output
- 75. Embedding UIMA component metadata into the source code

This requirement can be fulfilled with implementation of requirement 75. However, it is common that components take a model as input parameter (for instance in the case of DKPro Core) and hence the languages supported by the component are determined by the available models. In such cases language information can not be encoded in the component metadata. One option is to implement a functionality in the registry to look up available models for each component and update the component language capabilities based on the model's language. This function can be triggered every time a new model is added to the registry.

Table 11. Compliance assessment

Product	Version	Compliant	Justification	Status
ARGO	0.5	Partial	Argo components can embed language metadata in their UIMA XML descriptors, however most components do not declare this information.	Draft
DKPro Core	1.8.0	Partial	Most DKPro Core components use language-specific models. It is impossible to know the full set up models for a component as new models can be created by users at any time. For those components where the language support is hard-coded (i.e. not determined by the model used), the component ususally bears language metadata in the form of a uimaFIT LanguageCapability annotation which translates into a language capability declaration in the component's UIMA descriptor.	Draft
ILSP	1.2.1	Unknown	Unknown	Draft

88. Embedding output format in UIMA component metadata

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 45. S/W (tools, web services, workflows) must indicate format of their output
- 75. Embedding UIMA component metadata into the source code
- 86. Attaching format properties to the description of the inputs and outputs that use file

This requirement can be fulfilled with implementation of requirement 75. The output format of UIMA components can be defined using @MimeTypeCapability annotation.

NOTE	This can be mapped to the OMTD-SHARE schema elements *mimeType*, which should take values from the IANA mimetype controlled vocabulary, and, in case this is not sufficient, *dataFormatSpecific* (a free text field for specifying a name of a data format) and *documentationURL* (link to a URL with more infomation about the format), all embedded in *outputResourceInfo*.

Table 12. Compliance assessment

Product	Version	Compliant	Justification	Status
ARGO	0.5	No		Draft
DKPro Core	1.8.0	Full	Components define their output format using @MimeTypeCapability annotation.	Draft
ILSP	1.2.1	Unknown	Unknown	Draft

89. Version documentation in parallel with component/resource

Concreteness: *concrete*
Strength: *recommended*
Category: *WG1*,*WG2*,*WG3*,*WG4*
Status: *draft*

See also

- 50. Documentation references should be versioned

To keep the source material (i.e. component/resource) and documentation in sync, it is a good practice to version documentation along with the source material with identical versions. This means that documentation should reside in a subdirectory inside the source directory. It is also recommended that the source directory is stored under a version control repository. Every time a release is made, contents of the release should be identical to a specific time-state of the repository. This guarantees that each release of the source material contains its corresponding documentation and they are always kept in sync.

For software (i.e. components) the documentation should be stored in a module dedicated for this purpose (e.g. for softwareX there could be softwareX-documentation module).

In some cases a certain version of documentation needs to be updated; for instance, to fix errors or provide more examples (not typical for software). In this scenario, it is recommended to publish the updated documentation with the same version identifier but with an updated revision number. For example, if the first documentation is released with version **v2.0 rev.1** then the updated documentation should be published with version **v2.0 rev.2**. As an alternative, update date of the document can be attached instead of the revision identifier.

Product	Version	Compliant	Justification	Status
TheSoz	Jun-16	No	Documentation is not versioned (http://lod.gesis.org/thesoz/de/help.html)	Final
Agrovoc	21/01/2016	No		final
OLiA	Jun-16	No		final
LAPPS	Jun-16	No		final
Licences	Jun-16	Unknown		final
Alvis	0.5rc	No	No version	final
ARGO	0.5	No		draft
DKPro Core	1.8.0	Full	Documentation is versioned in parallel with the source code	final
GATE	8.2	Unknown	Possible for all GATE components to be written this way	final
ILSP	1.2.1	Unknown		draft

90. Components must be assigned at least one category from the OMTD-SHARE controlled vocabulary for component types

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*,*WG4*
Status: *draft*

The OMTD-SHARE metadata schema includes the element *componentType* which can be used for classifying components with values taken from a controlled vocabulary created in the OpenMinTeD project (implemented in SKOS: *pending link*). Each component must be assigned at least one value that best describes its function/task; the assignment of multiple values is possible, allowing for multiple classification. The current set of values focuses on terms of functions/tasks used by NLP and TDM experts, but it will be extended to include terms for non-expert users, catering for the description of integrated complex tasks.

- 8. Components must associate themselves with categories defined by the OpenMinTeD project

Product	Version	Compliant	Justification	Status
Alvis		Unknown	Unknown	Draft
ARGO	0.5	No	This requirement could be fulfilled through a change to the specification of the Argo Descriptor file.	Draft
DKPro Core	1.8.0	No	DKPro Core uses naming conventions for components and DKPro Meta implements a set of rules to derive component types from these naming conventions. However, the component types used by DKPro Meta (and consequently the DKPro Core reference documentation) do not fully correspond to the OMTD-SHARE component types. Also, the component type is not explicitly included with the respective component metadata. It can only be found in the reference documentation.	Draft
GATE	8.2	Unknown	Unknown	Draft
ILSP		Unknown	Unknown	Draft

91. Encoding citable publications (for scholarly attribution) in resource metadata records

Concreteness: *concrete*
Strength: *recommended*
Category: *WG1*,*WG2*,*WG4*
Status: *draft*

- 13. Citation information for component should be included in the metadata

It is customary for resource creators/owners to ask for scholarly attribution through the citation of publications; these are usually publications that describe the resource, its functionalities, design principles etc. The OMTD-SHARE metadata schema includes a specific metadata element for this, namely 'mustBeCitedWith', which is recommended and can be filled in with

- the identifier of a publication (‘publicationIdentifier’) or
- a free text field (‘documentUnstructured’), which can be used for recording a whole bibliographic record.

The preferred option is adding the identifier of the publication (with the appropriate attributes).

Name	Element/Attribute	Type	Strength	Repeatability	Description
publicationIdentifier	element	xs:string	mandatory (choice with documentUnstructured)	yes	Reference to a PID, DOI or any kind of identifier used for the publication; must obligatorily be used with the attribute *publicationIdentifierSchemeName*
publicationIdentifierSchemeName	attribute	open controlled vocabulary	mandatory	no	Reference to the publication scheme that has been used for the identifier (e.g. DOI, handle, arXiv, bibcode etc.); the list of values includes the most widespread schemes used for publications

Name	Element/Attribute	Type	Strength	Repeatability	Description
schemeURI	attribute	URI	mandatory when applicable	no	Reference to a URI of the resource scheme that has been used for the identifier, providing more information about the resource scheme
documentUnstructured	element	xs:string	mandatory (choice with publicationIdentifier)	no	Used either as a free text description or as an unstructured (free form) presentation of a bibliographic record; alternative to the structured presentation of a document

Further specifications according to this requirement:

- for components, in accordance with 2. Component metadata have to be embedded into the component source code, this information should be provided in the component metadata. The recommended way to do this is by adding Java annotations in the GATE and UIMA descriptors specific to this element. We recommend the usage of the @citation element. Within this element the author should place one or more bibliographic records, preferably in bibtex format.

NOTE For clarification reasons, it is noted that this relation is different from the 'datacite:relationType' *isCitedBy*, which records the actual citation of the resource in various publications; in fact, it is closer to the *isDocumentedIn* and the *dc:bibliographicCitation* but this relation type doesn't highlight the preferential nature of the citable publication.

Table 13. Compliance assessment

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	No	Not supported for the moment.	Draft
ARGO	0.5	No	No plans to support this.	Draft
DKPro Core	1.8.0	No	Occasionally, pointers to relevant papers are included in the documentation or metadata of components, but this information is only indicative. The DKPro Core website explicitly asks users to look up relevant papers for the components they use - or more specifically - the tools they wrap and the models that are used with them.	Draft
GATE	8.2	Unknown	Unknown	Draft

Product	Version	Compliant	Justification	Status
ILSP		Unknown	Unknown	Draft
TheSoz	June-16	No		Final
Agrovoc	21/01/2016	No		Draft
OLiA	June-16	No		Draft
LAPPS	June-16	No		Draft

92. Including license text in resource packages

Concreteness: *concrete*
Strength: *recommended*
Category: *WG1*,*WG3*,*WG4*
Status: *draft*

- 51. License should be attached

The license text of a resources should be included with a packaged resource. The exact way this is done depends on the type of packaging.

ZIP files

In ZIP files, the license should be located at the root of the file in a plain text file called `LICENSE`. The file may optionally have the suffix `.txt`. If multiple licenses apply, all of them should be concatenated in that file.

Java JAR files

In Java JAR files, the license should be located in a plain text file `META-INF/LICENSE`. The file may optionally have the suffix `.txt`. If multiple licenses apply, all of them should be concatenated in that file.

Docker

FIXME

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	partial	Exist for the corpus	Draft
ARGO	0.5	Partial	Some existing Argo components contain a LICENSE.txt file, however this is not always the case and these license files don't get included with the packaged components.	Draft
DKPro Core	1.8.0	Partial	JAR files for DKPro Core components contain a LICENSE.txt file. JAR files for models so far usually do not contain a LICENSE.txt file.	Draft
GATE	8.2	Unknown	Unknown	Draft
ILSP		Unknown	Unknown	Draft

93. Provide identifiers for knowledge resource elements

Concreteness: *concrete*
Strength: *recommended*
Category: *WG2*
Status: *draft*

See also

- 67. Knowledge Resource Element Id

When making use of multiple knowledge resources, TDM workflows may need to identify individual knowledge resource elements and knowledge resource schema elements. For example,

- A workflow may need to access information about an element.
- Multiple resources may be harmonised, requiring the relationships between schema elements to be specified.

For this identification purpose, knowledge resource schema elements should be identified with a URI. For Linked Data resources, the following identifiers should be used:

- JSON-LD - the @id keyword
- RDF/XML - the attributes xml:base, rdf:ID and rdf:about
- XML - the xml:id attribute

Product	Version	Compliant	Justification	Status
TheSoz	June 2016	Full		Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	Full		Final
OLiA	June 2016	Full		Final
LAPPS	June 2016	Full		Final
Ontolex	June 2016	Full		Final
CLARIN CCR	June 2016	Full		Final
schema.org	June 2016	Full		Final

94. Data Category Linking Vocabulary

Concreteness: *concrete*
Strength: *recommended*
Category: *WG2*
Status: *draft*

See also

- 68. Data Category Linking Vocabulary
- 69. Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.

Knowledge resource authors should provide linkage between their own resource, other resources in their domain, and generic resources such as citation standards. This enables TDM workflows that make use of multiple resources to resolve relationships between those resources.

Where linking between knowledge resources is modeled by a Linked Data resource, mapping should be expressed through RDF statements, using relations from SKOS, together with the following OWL and RDF object properties:

- owl:sameAs
- owl:equivalentClass
- owl:equivalentProperty
- rdfs:subClassOf
- rdfs:subPropertyOf

Product	Version	Compliant	Justification	Status
TheSoz	June 2016	Full		Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	No		Final
OLiA	June 2016	Full		Final
LAPPS	June 2016	Full		Final
Ontolex	June 2016	Full		Final
CLARIN CCR	June 2016	No		Final
schema.org	June 2016	Full		Final

95. The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.

Concreteness: *concrete*
Strength: *recommended*
Category: *WG2*
Status: *draft*

See also

- 72. The KR format should be in a standard format such as XML, JSON-LD or RDF/XML.

Standard formats for knowledge resources enables TDM to make use of common software infrastructure for ingesting and using knowledge resources, and can ease knowledge resource interoperability. Knowledge resources should use XML or JSON based syntax. For Linked Data resources, any RDF format may be used, for example JSON-LD or RDF/XML. Other formats are acceptable for resources that are not based on Linked Data.

Product	Version	Compliant	Justification	Status
TheSoz	June 2016	Full	RDF	Final
Agrovoc	21/01/2016	Full		Final
JATS	1.1	Partial	XSD/RELAX-NG/DTD	Final
OLiA	June 2016	Full		Final
LAPPS	June 2016	Full		Final
Ontolex	June 2016	Full		Final
CLARIN CCR	June-2016	Full		Final
schema.org	1.2.1	Full		Final

96. Unique identifiers and versions for components using Maven

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*,*WG4*
Status: *draft*

See also

- 6. Components should have a unique identifier and a version number
- 77. Unique identifiers and a versions for UIMA components
- 82. Unique identifiers and a versions for GATE components

Each GATE or UIMA component is a Java class and as such can be uniquely identified by its classname. Components are to be distributed via Maven and versioning information will be recorded as part of the POM. Together these two sources provide the following information

Coordinate	Description	Source
groupId	Group ID of the Maven artifact containing the component	Maven POM
artifactId	Artifact ID of the Maven artifact containing the component	Maven POM
version	Version of the Maven artifact containing the component	Maven POM
packaging	**optional** Packaging of the Maven artifact containing the component	Maven POM
classifier	**optional** Classifier of the Maven artifact containing the component	Maven POM
class	Name of the class implementing the component	UIMA resource specifier or GATE CREOLE Metadata (derived from classname)

In UIMA, the identifier has the following forms depending on the presence of the optional coordinates:

Component identifiers

```
uima-mvn:{groupId}:{artifactId}:{version}#{class}
uima-mvn:{groupId}:{artifactId}:{classifier}:{version}#{class}
uima-mvn:{groupId}:{artifactId}:{packaging}:{classifier}:{version}#{class}
```

Example

```
uima-mvn:my.organization.tdm:dictionary-tools:1.0.1#my.organization.tdm.DictionaryLookup
```

NOTE

The version is also available from the UIMA resource specifier. It must be in sync with the version specified in the Maven POM. The uimaFIT Maven plugin goal `generate` ensures this by determining the version in the generated UIMA resource specifier from the version specified in the Maven POM. The `overrideComponentVersion` and `componentVersion` settings in the uimaFIT MAven plugin must **not** be used. Neither must the uimaFIT `@ResourceMetaData` annotation be used to manually provide a version.

Table 14. Compliance assessment

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Alvis is provided as Maven artifacts. Decomposing each component as a maven sub-artfifact is not done.	Draft
ARGO	0.5	Full	All Argo components are available as Maven artifacts, however the Argo platform itself only supports UIMA PEAR files , which are produced from these Maven artifacts.	Draft
DKPro Core	1.8.0	Full	UIMA components are provided as Maven artifacts.	Draft
ILSP		Unknown	Unknown	Draft
GATE	8.2	Partial	Components are not yet distributed by Maven and as such versioning info is not available	Draft

97. Declaring scaleout capability in UIMA

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG4*
Status: *draft*

See also

- 11. Components must declare whether they can be scaled within a workflow

Whether or not a component maybe replicated/scaled up within a workflow by the workflow engine can be declared in UIMA using the `multipleDeploymentAllowed` flag. If this flag is set to `true`, then the workflow engine may create multiple instances of the component and run them in parallel with the implicit effect that each component may only see a subset of the documents to be processed. Components that write/export data should set this flag to `false`.

The flag can be set using a uimaFIT annotation on the component class.

Setting the `multipleDeploymentAllowed` using uimaFIT

```
@OperationalProperties(multipleDeploymentAllowed=false)
class MyComponentClass
```

Table 15. Compliance assessment

Product	Version	Compliant	Justification	Status
ARGO	0.5	Full	UIMA Components can be deployed multiple times within a workflow execution if this flag is set and it is not an instance of CollectionReader, CasConsumer, WebServiceReader, WebServiceConsumer, InteractiveAnalysisEngine	Draft
DKPro Core	1.8.0	Full	Flag is (un)set depending on the type of component. Processing components can usually be scaled while writers cannot.	Draft
ILSP		Unknown	Unknown	Draft
GATE	8.2	Unknown	Unknown	Draft

98. Publishing components via software repositories (Maven, Docker)

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG4*
Status: *draft*

See also

- 28. Processing components should be downloadable

In order to make components downloadable, they must be published either via a Maven repository or via a Docker repository. The respective repositories used should be Maven Central and DockerHub as these are by default supported by related tooling.

The process of publishing software via Maven Central is detailed in the OSSRH Guide.

The process of publishing software via DockerHub is detailed here.

Table 16. Compliance assessment

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	The Alvis engine is publish via a maven repository, Dockering Alvis is in progress. Note that, into Alvis, an executable component is a couple of <Alvis engine + the component>.	Draft
ARGO	0.5	No	Components are installed into an Argo instance using a script with UIMA PEAR files; Argo does not directly interact with a Maven/Docker repository.	Draft
DKPro Core	1.8.0	Full	DKPro Core is fully available from Maven Central.	Draft
ILSP		Unknown	Unknown	Draft
GATE	8.2	Unknown	Unknown	Draft

99. Encoding in the metadata a direct access link for content resources

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*
Status: *draft*

See also

- 4. URL to actual content must be discoverable

The contents of a resource that is going to be processed for TDM purposes (scholarly publications in OpenMinTeD) must be directly accessible by components in a single step. In order to avoid confusion and misconceptions, the access location of the resource must be provided in a metadata element conceived for this purpose and not through the identifier(s).

The following recommendations, therefore, are in order:

- aggregators (OpenAIRE and CORE) and other content resource providers that register their resources directly to the OpenMinTeD registry must use the element 'downloadURL' for encoding only direct URL's, i.e. leading to the file(s) of the resource, and not for landing pages
- publishers, journals, literature repositories, archives, etc. that wish to provide access to their publications for TDM purposes can do so through OpenAIRE and CORE, following their respective guidelines.

NOTE	The element 'downloadURL' can be repeated for multiple exact copies of the same publication (e.g. PDF versions of the publication distributed at various websites). For different variants of the same publication, e.g. in PDF and HTML format, the whole 'documentDistributionInfo' module can be repeated, with the relevant information encoded in the 'downloadURL' and 'mimetype' elements.

Table 17. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft

100. Providing access to content resources (sharing/exposing and transferring)

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*
Status: *draft*

See also

- 4. URL to actual content must be discoverable
- 99. Encoding in the metadata a direct access link for content resources

The contents of a resource that is going to be processed for TDM purposes (scholarly publications in OpenMinTeD) must be directly accessible by components in a single step. Publications are registered to OpenMinTeD via OpenAIRE and CORE. End-users … (user corpus building process via RESTful APIs) OpenAIRE and CORE: harvesting process (OAI-PMH) content connector from publishers: ResourceSync

Table 18. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft

101. Making models and annotation resources accessible as entities distinct from the components they are compatible with

Concreteness: *concrete*
Strength: *recommended*
Category: *WG1*,*WG2*,*WG4*
Status: *draft*

See also

- 16. Models/resources should be useable across different component collections/platforms

To ensure that this requirement is fully specified, it is recommended that models and resources that can be used for annotating content resources (e.g. tagsets, lexica, ontologies etc.) are made accessible and described as separate entities from the components they can be deployed with. This means that for models and other resources embedded in Maven artifacts together with components, they should be packaged as separate entities but can still be stored in Maven, where they will be collected from. Further to this, a set of metadata elements that can be used for the automatic identification of compatible ancillary resources and s/w components must be filled in their descriptions; these include mainly:

- 'language'(for all resources)
- 'variant' (for models)
- format, i.e. 'mimetype' and, optionally, 'dataFormatSpecific' & 'documentationURL' (for all resources)
- 'relationType' with value 'isCompatibleWith': a relation that can be used to encode compatible resources, e.g. a model that can potentially be used with two or more taggers, or a tagger that can use all models in a specific format; it is encoded as an external relation linking two resources, through the resource identifier (preferably) or the resource name.

Table 19. Compliance assessment

Product	Version	Compliant	Justification	Status
Alvis		Partial	Possible but not done yet	Draft
ARGO	0.5	No		Draft
DKPro Core	1.8.0	Partial	Models are shipped as separate maven artifacts (there may be very few exceptions). Compatibility between components and models is recorded in the POM of the component in DKPro Core via the Maven Managed Dependencies mechanism. Mimetype, dataFormatSpecific are not recorded. Some models include links to the relevant websites. This is planned to be extended in future.	Draft

Product	Version	Compliant	Justification	Status
GATE	8.2	No	Currently (default) resources are stored in the same artifact as the components. Also while GATE9 can load components from maven artifacts that are not CREOLE plugins cannot be loaded and models/resources are not treated as plugins. A workflow system that uses the GATE API could however pull in resources via maven which are not GATE plugins.	Draft
ILSP		Unknown	Unknown	Draft
TheSoz	June-16	Partial	Language attributes are stored per concept and term, not on the whole resource. Vocabulary and schema of the resource are declared in the `void.ttl` file.	Final
Agrovoc	21/01/2016	No		Draft
OLiA	June-16	No		Draft
LAPPS	June-16	No		Draft

102. Adding version information in the metadata descriptions of all resources

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*,*WG2*,*WG3*,*WG4*
Status: *draft*

See also

- 32. Version must be included in the metadata description for all resources
- 96. Unique identifiers and versions for components using Maven

Although versioning is considered important for all resource types, the practices used for its encoding differ across them. Therefore, in OpenMinTeD the general rule is that the version of the resource must be clearly indicated in its metadata description. This is done with the element 'version' (of type *xs:string*) and, if the providers wish, they can also provide more information about the types of changes between the versions in the element 'revision' and the date of the last update ('lastDateUpdated').

However, the values that the 'version' element takes can differ depending on the source and resource type; given that the OMTD-SHARE element 'version' is of type *xs:string*, this can be accommodated without problems in the mapping process:

- For publications: following the OpenAIRE guidelines (new version to be released), the values of the OpenAIRE element 'version' must be taken from a controlled vocabulary (*accepted, published, draft, submitted, updated*)
- For all other resources, the most popular practice is to provide a number (e.g. 1.0, 3.0.1 etc.) and this is endorsed by OpenMinTeD; semantic versioning principles are recommended where possible for its assignment; the element, though, where this information is encoded in existing metadata descriptions may differ:
  - For components collected from Maven repositories: according to 96. Unique identifiers and versions for components using Maven, this information must be added in the 'version' element in the pom xml file
  - For resources described with metadata in RDF, the 'owl:versionInfo' is the equivalent to the OMTD-SHARE 'version' element
  - If there is no information on the metadata record of a resource harvested into OMTD, the version element is to be filled in with the creation date of the resource and, if this is again missing, the date of the registration of the resource in OpenMinTeD.

NOTE In the next version of the schema, the "version" element will be restricted to the numeric type of encoding versions, following the semantic versioning principles, and another element will be added that will be of type "controlled vocabulary" that will take values such as *accepted, published, submitted* etc. for publications, but also *alpha, beta, testing* etc. for s/w resources.

Table 20. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Partial	Versions exist for the components	Draft
OpenAIRE		Unknown	Unknown	Draft

Product	Version	Compliant	Justification	Status
Frontiers		Unknown	Unknown	Draft
Alvis	0.5rc	Partial	Versions exist for the components. In general, version for resources has to be adapted to follow the semantic versioning principles.	Draft
ARGO	0.5	Partial	Type Systems and Components have versioning information, however other resources do not.	Draft
DKPro Core	1.8.0	Full	Version information is contained in the pom file	Draft
GATE	9	Full	Components distributed via maven include version information within the pom.xml file.	Draft
ILSP		Unknown	Unknown	Draft
TheSoz	June-16	Full	owl:versionInfo in void.ttl file	Final
Agrovoc	21/01/2016	Partial	modification date (dcterms:modified) is available	Final
OLiA	June-16	Partial	no version id, but a textual description of the resource's evolution.	Final
LAPPS	June-16	No		Final

103. Specifying access mode of resources and encoding it in the metadata descriptions

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*,*WG2*,*WG4*
Status: *draft*

See also

- 38. Access mode of resources must be included in the metadata

All content resources to be mined for OpenMinTeD purposes as well as ancillary resources (tagsets, lexica, models etc.) must be downloadable and accessible in a single step (cf. 4. URL to actual content must be discoverable, 99. Encoding in the metadata a direct access link for content resources). The elements 'distributionMedium', with values *downloadable*, *accessibleThroughInterfacace*, etc. (cf. link to the controlled vocabulary to be provided) is reserved for the access mode.

Components can be accessed as web services and/or as downloadable tools, and this must be indicated in the 'componentDistributionMedium' element with values *webService*, *sourceCode*, *executableCode* or *sourceAndExecutableCode*; if a component is available in more than one modes, it is recommended that this is described as two distinct distributions (i.e. with two 'distributionInfo' elements) that allow better encoding of all relevant details (e.g. potentially different licensing terms, different access locations etc.). See also 98. Publishing components via software repositories (Maven, Docker) for making components downloadable.

Table 21. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft
Alvis	0.5rc	Full	The Alvis engine, its components and the available resources are downlodable from repositories and web sites.	Draft
ARGO	0.5	Partial	Source code for our open source components is available on GitHub.	Draft
DKPro Core	1.8.0	Full	DKPro Core components are maven modules which can always be accessed by downloading from maven repositories.	Draft
GATE	8.2	Unknown	Unknown	Draft
ILSP		Unknown	Unknown	Draft
TheSoz	June-16	Full	SPARQL endpoint and URI lookup endpoint in void.ttl file	Final
Agrovoc	21/01/2016	Full	void:sparqlEndpoint, void:dataDump	Final
OLiA	June-16	No		Final
LAPPS	June-16	No		Final

104. Encoding funding information in the metadata descriptions of all resources

Concreteness: *concrete*
Strength: *recommended*
Category: *WG1*,*WG2*,*WG4*
Status: *draft*

See also

- 47. Information on funding of resources may be included in the metadata

Funding information in metadata descriptions of all resources is optional; we recommend adopting the DataCite recommendations for all resources, which are also endorsed by OpenAIRE:

- 'fundingReference': 'funderName', and optional 'funderIdentifier' (with attribute 'type': *ISNI, GRID, Crossref funder Id, other*), 'awardNumber' (with attribute 'awardURI') and 'awardTitle'.

NOTE	in the current version of OMTD-SHARE (v1.1), funding information is encoded in the 'fundingProject' element, which can be filled in with a 'projectIdentifier' or a 'projectName'. Projects can also be uploaded to the OpenMinTeD registry with a set of specific metadata elements which include information on the funder(s), types of awards etc. The next version will be aligned with this requirement.

To fulfill this requirement, further actions are in order: * addition in the component metadata of the required information; this can be done through the following recommendations: * Usage of the following java annotations (in line with OpenAire's standards) * @funderName * @funderIdentifier (optional) * @funderIdentifierType (mandatory if funderIdentifier is present) * @awardNumber * @awardURI (optional) * @awardTitle

- resources coming from the Linked Data community:
  - the 'foaf:fundedBy' property can be used to record the funding organization; if possible, adding an identifier (e.g. ISNI) for the organization would be an asset.

Table 22. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft
Alvis	0.5rc	No	The information does not exist or is not formalized.	Draft
ARGO	0.5	No		Draft

Product	Version	Compliant	Justification	Status
DKPro Core	1.8.0	N/A	DKPro Core is now a community effort based on volunteer contributions. There is no dedicated funding. Also, DKPro Core mainly wraps third-party tools in UIMA. We do not keep track of third-party funding information and are not aware of third-parties providing such information in an automatically processable way.	Draft
GATE	8.2	No		Draft
ILSP		Unknown	Unknown	Draft
TheSoz	June-16	No	Only resource publisher is declared	Final
Agrovoc	21/01/2016	No	only its creator (FAO)	Final
OLiA	June-16	No	Some ontologies reference the project in which the resource was created.	Final
LAPPS	June-16	No		Final

105. Encoding of format in the metadata description of content resources

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*
Status: *draft*

See also

- 37. Information on the structural annotation (layout) of resources should be included in the metadata of the resource
- 39. Content resources must include metadata on their format (e.g. XML, DOCX etc.)

All content resources (i.e. publications, but also ancillary resources, such as lexica, tagsets etc.) that are involved in the TDM process must include in the metadata description information on the format with which they are distributed (e.g. docx, xml, rdf/xml etc.). The OMTD-SHARE schema includes the following metadata elements:

- 'mimeType': mandatory; must be filled in with one of the values of the IANA (Internet Assigned Numbers Authority, http://www.iana.org/assignments/media-types/media-types.xhtml)
- 'dataFormatSpecific': this element is optional and can be used to provide further information about the format of the resource, when the 'mimetype' does not suffice, e.g. for xml files structured following a specific schema (e.g. structural annotation schema, such as TEI or JATS, or a simple CSV file with columns ordered in a specific way); to help the resource provider, the element takes a list of values with the data formats currently covered by partners of the OpenMinTeD consortium (link to the controlled vocabulary to be provided).
- 'documentationURL': for data formats that are not included in the previous list, the resource providers are recommended to include a link to a URL where the format is documented (e.g. with an XSD or other formal explanation of the format, examples etc.).

In addition, standard file extensions are highly recommended when publishing a content resource, as it helps in discovering the correct mimetype.

NOTE	If there are multiple distributions of the same resource with different formats from different locations, they should be described as separate distributions with the 'mimetype' repeated inside the 'distributionInfo' wrapper element. If the same location is used for all distributions (i.e. the access location is a landing page), providers can simply repeat the 'mimetype' inside the 'textFormatInfo' element at the resource level. For corpora composed of resources with various formats, again the 'mimetype' element inside the 'textFormatInfo' can be repeated.
NOTE	As a general rule, standard text formats should be preferred, e.g. for structural annotation, the JATS article tag suite (https://jats.nlm.nih.gov/) or TEI (http://www.tei-c.org/index.xml).
NOTE	Under consideration: discuss how the page with the controlled vocabulary of dataFormatSpecific can include more information on each format so that it's directly usable for components.

Table 23. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft

106. Encoding licensing terms in the metadata description of the resource

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG3*
Status: *draft*

See also

- 33. Licensing information must be included in the metadata

It is mandatory to include an indication of the licensing terms under which a resource is made accessible. In OpenMinTeD this can be implemented in one of the following ways, in preferential order:

- Selection from a list of values of the most popular standard licences in the 'licence' element (link to the controlled vocabulary to be provided) and indication of the licence version in the 'version' element;
- For licences not included in the list, selection of the 'other' value in the 'licence' element and usage of the elements 'nonStandardLicenceTermsName' to provide a name for the licence and 'nonStandardLicenceTermsURL' to provide a link to the licence text or 'nonStandardLicenceTermsText' to add the licence text.
- For legacy resources that come without a clear indication of licence but which can be classified into one of the Rights Statements discussed in OpenMinTeD, the last option is to use the element 'rightsStmtName' and 'rightsStmtURL' (with a link to the definition of the rights statement)

NOTE	For resources that are produced through OpenMinTeD operations (e.g. user queries for creating corpora, resources annotated by workflows in the platform etc.), the user will be shown an aggregation of all licence values of the resources.

To fulfill the above requirements, further recommendations:

- For all resources: the 'dc:license' (which requires a licence document) is preferable to 'dc:rights' (which can be filled in with a free text statement); it is essential to also attach the full legal document containing the licensing terms to the packaged resource and add the licenseURI in the metadata description.
- Components coming from Maven repositories: the wrapper element <license> with nested elements <name> & <url> for the full title of the licence (mentioning also version) and for the url respectively must be obligatorily filled in; if possible, use of one of the OpenMinTeD recommended licences is preferred.
- Resources from the Linked Data community can use the dc:license or dc:rights elements to provide a link to the licence text or a licence name.

Table 24. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft

Product	Version	Compliant	Justification	Status
Alvis	0.5rc	Partial	Provided in free-text mode. Identification of the licences according to the most popular standard licences in progress.	Draft
ARGO	0.5	Partial	Licenses of components maybe declared in the component metadata, in a LICENSE.txt file at the root of a component's source code project or in the Maven POM file.	Draft
DKPro Core	1.8.0	Partial	For components, license information is in the POM file and more detailed in the LICENSE.txt file included in each Maven artifact. For models, the information may presently be missing or only be available in a DKPro Core-specific metadata file associated with the model.	Draft
GATE	9	Full	License information should be includedin the pom.xml from where it can be automatically extracted	Draft
ILSP		Unknown	Unknown	Draft
TheSoz	June-16	Full	dc:license in void.ttl file	Final
Agrovoc	21/01/2016	Full	dc:license in void.ttl file	Final
OLiA	June-16	No		Final
LAPPS	June-16	No		Final

107. Encoding metadata on domain/subject/ classification for all resources when applicable

Concreteness: *concrete*
Strength: *recommended*
Category: *WG1,WG2*
Status: *draft*

See also

- 36. Classification metadata should be included, where applicable, in the metadata record of the resource

Domain/subject classification is valuable both for the discovery of resources of particular interest to the users, as well as for ensuring better results for the TDM processes. However, the vocabularies used for the classification of resources are numerous (cf. {http://id.loc.gov/vocabulary/classSchemes.html} and {https://www.loc.gov/standards/sourcelist/subject.html}), and there isn't yet any common consensus for adopting only a few of them.

Therefore, OpenMinTeD suggests that (a) relevant metadata elements are set at the recommended level, (b) the use of widespread classification vocabularies is promoted and (c) mandates that at least the classification vocabulary, if any, used for the specific resource is described with a name and URI.

The above requirements translate into the following metadata elements:

- recommended 'domain', with attributes mandatory 'classificationSchemeName' and optional 'schemeURI'
- recommended 'subject', with attributes mandatory 'classificationSchemeName' and optional 'schemeURI'
- recommended 'keyword', which can be used as a last resort to at least ingest keywords currently used for the classification of resources.

The attribute 'classificationSchemeName' uses a controlled vocabulary (link to be provided) with widespread classification schemes to help users in filling in this information.

The 'dc:subject' element can be used for encoding classification, but this doesn't provide the required information.

NOTE	Classification is also possible with the metadata elements 'textGenre', 'textType', and 'register', which also carry the same attributes as domain and subject.

Components are also included in the compliance table, although in the vast majority, they are domain-independent. The reason for their inclusion is to indicate whether they come with specific resources (e.g. models) that are trained with corpora of specific domains.

Table 25. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft
Alvis		No	Not currently	Draft

Product	Version	Compliant	Justification	Status
ARGO	0.5	No		Draft
DKPro Core	1.8.0	No	For some models, the “variant” coordinate includes relevant information.	Draft
GATE	8.2	No	No current built in support but I assume this is another OMTD-SHARE level requirement which has no bearing on the actual component implementations	Draft
ILSP		Unknown	Unknown	Draft
TheSoz	June-16	Full	via DbPedia URIs. The following values are present in the current void.ttl: dc:subject http://dbpedia.org/resource/Social_sciences ; dc:subject http://dbpedia.org/resource/Thesaurus	Final
Agrovoc	21/01/2016	Full	DbPedia URIs	Final
OLiA	June-16	Unknown	Unknown	Draft
LAPPS	June-16	Unknown	Unknown	Draft

108. Encoding language information in the metadata of content resources

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*,*WG2*
Status: *draft*

See also

- 41. Content resources must include metadata on their language(s)

Language is one of the most important features for discovering content resources and for ensuring the proper operation of processing components. Therefore, it is important not only to include it in the metadata descriptions of resources but also to normalise it.

For normalistaion purposes, OpenMinTeD recommends the use of the IETF's BCP 47 (https://tools.ietf.org/html/bcp47), which specifies syntax for language tags, so as to include language, region, script and variant tags in one code.

Language information is mandatory and can be repeated for multilingual resources.

The metadata elements to be used are:

- 'language' for the language of the contents of the resource: mandatory language tag, which is concatenated by mandatory language identifier (from the ISO 639 codes) and optional script, region and variant tags
- 'metalanguage': can be optionally used for lexical/conceptual resources and language descriptions, for the language that is used to describe a resource (e.g. a lexicon of Greek terms with definitions in English is considered as having 'language' *Greek* and 'metalanguage' *English*).

Table 26. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft
TheSoz	June-16	Full	via language attribute for each entity in the thesaurus	Final
Agrovoc	21/01/2016	No		Final
OLiA	June-16	No	Language only indirectly deducible from the description of the resource in the data type property 'comment' value.	Final
LAPPS	June-16	No		Final

109. Encoding statistical information in the content resources

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1,WG2*
Status: *draft*

See also

- 44. Statistical metadata that allow monitoring of resource versions may accompany resources

Statistical metrics are one of the ways that one can use to decide on whether a certain resource is of interest as well as to monitor versions of the same resource. Size is a feature that can be considered stable enough to use for this reason; however, size is recorded with various units of measurement, depending on the type of the resource as well as on community practices.

For this reason, OpenMinTeD uses a set of two mandatory metadata elements, 'size' and 'sizeUnit' which are combined together in the 'sizeInfo' element. 'SizeInfo' is obligatory and can be repeated to record different size dimensions, if the users wishes to. 'Size' records in the form of a number the size of a resource with regard to the 'sizeUnit', while 'sizeUnit' takes values from a controlled vocabulary (link to be provided), which includes a list of the most widespread values for resources.

Table 27. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft
TheSoz	June-16	Full	number of triples contained inside the resource and number of triples aligned with other resources (e.g. dbpedia)	Final
Agrovoc	21/01/2016	Full	number of concepts, labels	Final
OLiA	June-16	No		Final
LAPPS	June-16	No		Final

110. Assigning a unique persistent identifier for all resources

Concreteness: *concrete*
Strength: *mandatory*
Category: *WG1*,*WG2*,*WG3*
Status: *draft*

See also

- 35. All resources must include a unique persistent identifier
- 96. Unique identifiers and versions for components using Maven

All resources must be assigned an identifier (e.g.DOI, Handle PID, URI etc.) that allows them to be uniquely, non-ambiguously and persistently identified.

Resources that already have a PID, can include this in their metadata descriptions, in the element 'identifier'/'publicationIdentifier' using the 'resourceIdentifierSchemeName'/'publicationIdentifierSchemeName' attribute to specify the corresponding scheme, for instance:

- publications that already have a DOI can include this in the 'publicationIdentifier' with the value *doi* 'publicationIdentifierSchemeName'; DOIs are the recommended identifiers for publications
- datasets, lexica, etc. that already have a DOI or a Handle id, will include these in the 'resourceIdentifier' with 'resourceIdentifierSchemeName' *doi* or *hdl* respectively
- knowledge resources that come from the Linked Data community and have a URI, can encode this in the 'resourceIdentifier' with 'resourceIdentifierSchemeName' *uri*
- s/w components, in accordance to 96. Unique identifiers and versions for components using Maven, can include the component coordinates and add the value *other* for 'resourceIdentifierScheme' and *https://maven.apache.org/pom.html#Maven_Coordinates* for 'schemeURI'.

Resources without PIDs will be assigned one according to the procedure specified in the OMTD policies.

NOTE	The element 'identifier' is not reserved for PIDs; it can be used for any kind of identifier, including organization-internal identifiers.

Table 28. Compliance assessment

Product	Version	Compliant	Justification	Status
CORE		Unknown	Unknown	Draft
OpenAIRE		Unknown	Unknown	Draft
Frontiers		Unknown	Unknown	Draft
Alvis	0.5rc	Full	Components can be uniquely and persistently identified	Draft
ARGO	0.5	Full	All Argo components and type systems are identifiable using Maven coordinates.	Draft
DKPro Core	1.8.0	Full	UIMA components are provided as Maven artifacts.	Draft

Product	Version	Compliant	Justification	Status
GATE	9	Full	Components distributed via maven can be uniquely and persistently identified via their maven coordinates and class name	Draft
ILSP		Unknown	Unknown	Draft
TheSoz	June-16	Full	http://lod.gesis.org/thesoz/ (`void:uriLookupEndpoint` in `void.ttl` file)	Final
Agrovoc	21/01/2016	Full		Draft
OLiA	June-16	Full		Draft
LAPPS	June-16	Full		Draft

Bibliography

- [uimafit] Philip V. Ogren, Steven J. Bethard (2009) Building Test Suites for UIMA Components Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing (SETQA-NLP 2009). June 2009. (PDF)
- [Prokopidis2011]Prokopidis, P., Georgantopoulos, B. & Papageorgiou, H. (2011). A suite of NLP tools for Greek. in the 10th International Conference of Greek Linguistics. Komotini, Greece. (PDF)
- CORE FAQ's: https://core.ac.uk/about#faqs
- Knoth, P. (2013). From Open Access Metadata to Open Access Content: Two Principles for Increased Visibility of Open Access Content. Open Repositories 2013, Charlottetown, Prince Edward Island, Canada. (PDF)
- Príncipe, Pedro, Najla Rettberg, Eloy Rodrigues, Mikael K. Elbæk, Jochen Schirrwagen, Nikos Houssos, Lars Holm Nielsen, Brigitte Jörg (2014). OpenAIRE Guidelines: Supporting Interoperability for Literature Repositories, Data Archives and CRIS. Procedia Computer Science. Special issue on 12th International Conference on Current Research Information Systems, CRIS 2014 — Managing data intensive science – The role of Research Information Systems in realising the digital agenda. Elsevier. http://doi.org/10.1016/j.procs.2014.06.015
- OpenAIRE guidelines: https://guidelines.openaire.eu/en/latest/literature/index.html