OpenMinTeD Interoperability Specification

OpenMinTeD Interoperability Team (T5.2)

Version 1.0.0

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.

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 (21)

ID	Requirement	Concreteness	Status	Strength	WG's
4	URL to actual content must be discoverable	abstract	final	mandatory	WG1, WG2 (17), WG3 (23)
13	Citation information for component	abstract	draft	recommended	WG1, WG4 (33)
15	Human readable information should be provided by each resource	abstract	final	recommended	WG1, WG4 (33)
30	Metrics for the confidence level of the TDM operation should be included in the metadata	abstract	final	optional	WG1, WG4 (33)
31	Metrics for the performance of the TDM operation should be included in the metadata	abstract	final	optional	WG1, WG4 (33)
32	Version must be included in the metadata description for all resources	abstract	final	mandatory	WG1, WG2 (17), WG3 (23), WG4 (33)
33	Licensing information must be included in the metadata	abstract	final	mandatory	WG1, WG3 (23)
34	Licensing information should be expressed in a machine-readable form	abstract	final	recommended	WG1, WG3 (23)
35	All resources must include a unique persistent identifier	abstract	final	mandatory	WG1, WG2 (17), WG3 (23), WG4 (33)
36	Classification metadata should be included, where applicable, in the metadata record of the resource	abstract	final	recommended	WG1, WG2 (17)
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 (17), WG4 (33)
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	abstract	final	mandatory	WG1
40	Component metadata must include standardised categories/tags that make them easy to discover	abstract	final	mandatory	WG1, WG4 (33)
41	Content resources must include metadata on their language(s)	abstract	final	mandatory	WG1, WG2 (17)

ID	Requirement	Concreteness	Status	Strength	WG's
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 (33)
44	Statistical metadata that allow monitoring of resource versions may accompany resources	abstract	final	optional	WG1, WG2 (17)
45	S/W (tools, web services, workflows) must indicate format of their output	abstract	final	mandatory	WG1, WG4 (33)
47	Information on funding of resources may be included in the metadata	abstract	final	optional	WG1, WG2 (17), WG3 (23), WG4 (33)
48	All resource metadata records must include a reference to the metadata schema used for their description	abstract	final	mandatory	WG1, WG2 (17), WG3 (23), WG4 (33)
50	Documentation references should be versioned	abstract	final	recommended	WG1, WG2 (17), WG3 (23), WG4 (33)

WG2 (17)

ID	Requirement	Concreteness	Status	Strength	WG's
4	URL to actual content must be discoverable	abstract	final	mandatory	WG1 (21), WG2, WG3 (23)
10	Components should specify the types of the annotations that they input and output	abstract	draft	mandatory	WG4 (33), WG2
32	Version must be included in the metadata description for all resources	abstract	final	mandatory	WG1 (21), WG2, WG3 (23), WG4 (33)
35	All resources must include a unique persistent identifier	abstract	final	mandatory	WG1 (21), WG2, WG3 (23), WG4 (33)
36	Classification metadata should be included, where applicable, in the metadata record of the resource	abstract	final	recommended	WG1 (21), WG2

ID	Requirement	Concreteness	Status	Strength	WG's
38	Access mode of resources must be included in the metadata	abstract	final	mandatory	WG1 (21), WG2, WG 4 (33)
41	Content resources must include metadata on their language(s)	abstract	final	mandatory	WG1 (21), WG2
44	Statistical metadata that allow monitoring of resource versions may accompany resources	abstract	final	optional	WG1 (21), WG2
47	Information on funding of resources may be included in the metadata	abstract	final	optional	WG1 (21), WG2, WG 3 (23) , WG4 (33)
48	All resource metadata records must include a reference to the metadata schema used for their description	abstract	final	mandatory	WG1 (21), WG2, WG 3 (23) , WG4 (33)
50	Documentation references should be versioned	abstract	final	recommended	WG1 (21), WG2, WG 3 (23) , WG4 (33)
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 or RDF.	abstract	final	recommended	WG2

WG3 (23)

ID	Requirement	Concreteness	Status	Strength	WG's
4	URL to actual content must be discoverable	abstract	final	mandatory	WG1 (21), WG2 (17), WG3
32	Version must be included in the metadata description for all resources	abstract	final	mandatory	WG1 (21), WG2 (17), WG3, WG4 (33)
33	Licensing information must be included in the metadata	abstract	final	mandatory	WG1 (21), WG3
34	Licensing information should be expressed in a machine-readable form	abstract	final	recommended	WG1 (21), WG3
35	All resources must include a unique persistent identifier	abstract	final	mandatory	WG1 (21), WG2 (17), WG3, WG4 (33)
47	Information on funding of resources may be included in the metadata	abstract	final	optional	WG1 (21), WG2 (17), WG3, WG4 (33)
48	All resource metadata records must include a reference to the metadata schema used for their description	abstract	final	mandatory	WG1 (21), WG2 (17), WG3, WG4 (33)
50	Documentation references should be versioned	abstract	final	recommended	WG1 (21), WG2 (17), WG3, WG4 (33)
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

ID	Requirement	Concreteness	Status	Strength	WG's
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

WG4 (33)

ID	Requirement	Concreteness	Status	Strength	WG's
1	Components should be described by machine-readable metadata	abstract	final	mandatory	WG4
2	Component metadata should be embedded into the component source code	abstract	final	recommended	WG4
3	Component metadata is separable from the component	abstract	final	mandatory	WG4
5	Components should 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 should have a fully qualified name that follows the Java class naming conventions	concrete	final	mandatory	WG4
8	Components should associate themselves with categories defined by the OpenMinTeD project	abstract	final	mandatory	WG4
9	Components should declare their annotation schema dependencies	abstract	final	mandatory	WG4

ID	Requirement	Concreteness	Status	Strength	WG's
10	Components should specify the types of the annotations that they input and output	abstract	draft	mandatory	WG4, WG2 (17)
11	Components should 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	abstract	draft	recommended	WG1 (21), WG4
14	Components must maintain License information	abstract	draft	mandatory	WG4
15	Human readable information should be provided by each resource	abstract	final	recommended	WG1 (21), 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
18	Workflows should be described using an uniform language	abstract	draft	recommended	WG4
21	Configuration and parametrizable options of the components should be identified and documented	abstract	final	recommended	WG4
24	Using/treating workflows as components	abstract	final	mandatory	WG4
26	Ability to determine 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
30	Metrics for the confidence level of the TDM operation should be included in the metadata	abstract	final	optional	WG1 (21), WG4
31	Metrics for the performance of the TDM operation should be included in the metadata	abstract	final	optional	WG1 (21), WG4
32	Version must be included in the metadata description for all resources	abstract	final	mandatory	WG1 (21), WG2 (17), WG3 (23), WG4

ID	Requirement	Concreteness	Status	Strength	WG's
35	All resources must include a unique persistent identifier	abstract	final	mandatory	WG1 (21), WG2 (17), WG3 (23), WG4
38	Access mode of resources must be included in the metadata	abstract	final	mandatory	WG1 (21), WG2 (17), WG4
40	Component metadata must include standardised categories/tags that make them easy to discover	abstract	final	mandatory	WG1 (21), 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 (21), WG4
45	S/W (tools, web services, workflows) must indicate format of their output	abstract	final	mandatory	WG1 (21), WG4
47	Information on funding of resources may be included in the metadata	abstract	final	optional	WG1 (21), WG2 (17), WG3 (23), WG4
48	All resource metadata records must include a reference to the metadata schema used for their description	abstract	final	mandatory	WG1 (21), WG2 (17), WG3 (23), WG4
50	Documentation references should be versioned	abstract	final	recommended	WG1 (21), WG2 (17), WG3 (23), WG4

By Status

deprecated (10)

ID	Requirement	Concreteness	Strength	WG's
19	Components that use external knowledge resources should delegate access to a resource adapter instead of handling it themselves	abstract	optional	WG2 (17), WG4 (33)
20	Workflow engines should not require to see data	concrete	recommended	WG2 (17), WG4 (33)
22	The Workflow Engine Should Permit Saving Experimental Conditions in a Workflow	abstract	recommended	WG1 (21), WG4 (33)
23	The Workflow Engine should permit Licence Aggregation in Workflows	abstract	recommended	WG3 (23), WG4 (33)
25	Incorporation of multiple resources in parallel	abstract	recommended	WG4 (33)
29	The actual content of all content resources must be discoverable	abstract	mandatory	WG1 (21), WG2 (17), WG3 (23)
42	The metadata can include the information on which projects/workflows involve the resource	abstract	optional	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
46	Output resources of web services/workflows must be accompanied by provenance metadata	abstract	mandatory	WG1 (21), WG4 (33)
49	Metadata of tools should contain information about the models available for them	abstract	recommended	WG1 (21), WG4 (33)
52	License information must be in metadata	abstract	recommended	WG1 (21), WG3 (23)

draft (22)

ID	Requirement	Concreteness	Strength	WG's
5	Components should detail all their environmental requirements for execution	abstract	mandatory	WG4 (33)

ID	Requirement	Concreteness	Strength	WG's
6	Components should have a unique identifier and a version number	abstract	mandatory	WG4 (33)
10	Components should specify the types of the annotations that they input and output	abstract	mandatory	WG4 (33), WG2 (17)
11	Components should declare whether they can be scaled within a workflow	abstract	mandatory	WG4 (33)
13	Citation information for component	abstract	recommended	WG1 (21), WG4 (33)
14	Components must maintain License information	abstract	mandatory	WG4 (33)
18	Workflows should be described using an uniform language	abstract	recommended	WG4 (33)
51	License should be attached	abstract	recommended	WG3 (23)
53	Licensor must be entitled to grant license	abstract	recommended	WG3 (23)
54	Licensees should remain with a copy of the license	abstract	recommended	WG3 (23)
55	Standard licenses should be used	abstract	recommended	WG3 (23)
56	License should be machine readable	abstract	recommended	WG3 (23)
57	License should be understandable by non-lawyers	abstract	recommended	WG3 (23)
58	TDM must be explicitly allowed	abstract	recommended	WG3 (23)
59	Right for (temporary) reproduction must be granted	abstract	recommended	WG3 (23)
60	Boundary for derivative work must be clearly defined	abstract	recommended	WG3 (23)
61	No restrictions on TDM results which are not derived works	abstract	recommended	WG3 (23)
62	World-wide and irrevocable license grant	abstract	recommended	WG3 (23)
63	License must qualify for Open Access rights	abstract	recommended	WG3 (23)
64	License must qualify for Open Access uses	abstract	recommended	WG3 (23)
65	License must qualify for Open Access must not restrict use in any way	abstract	recommended	WG3 (23)

ID	Requirement	Concreteness	Strength	WG's
66	License must qualify for Open Access may include attribution requirements	abstract	recommended	WG3 (23)

final (40)

ID	Requirement	Concreteness	Strength	WG's
1	Components should be described by machine-readable metadata	abstract	mandatory	WG4 (33)
2	Component metadata should be embedded into the component source code	abstract	recommended	WG4 (33)
3	Component metadata is separable from the component	abstract	mandatory	WG4 (33)
4	URL to actual content must be discoverable	abstract	mandatory	WG1 (21), WG2 (17), WG3 (23)
7	Components should have a fully qualified name that follows the Java class naming conventions	concrete	mandatory	WG4 (33)
8	Components should associate themselves with categories defined by the OpenMinTeD project	abstract	mandatory	WG4 (33)
9	Components should declare their annotation schema dependencies	abstract	mandatory	WG4 (33)
12	Components should provide documentation describing their functionality	abstract	recommended	WG4 (33)
15	Human readable information should be provided by each resource	abstract	recommended	WG1 (21), WG4 (33)
16	Models/resources should be useable across different component collections/platforms	abstract	recommended	WG4 (33)
17	Components should be stateless	concrete	recommended	WG4 (33)
21	Configuration and parametrizable options of the components should be identified and documented	abstract	recommended	WG4 (33)
24	Using/treating workflows as components	abstract	mandatory	WG4 (33)
26	Ability to determine source of an annotation/assigned category	abstract	recommended	WG4 (33)
27	Components should handle failures gracefully	abstract	recommended	WG4 (33)
28	Processing components should be downloadable	abstract	recommended	WG4 (33)

ID	Requirement	Concreteness	Strength	WG's
30	Metrics for the confidence level of the TDM operation should be included in the metadata	abstract	optional	WG1 (21), WG4 (33)
31	Metrics for the performance of the TDM operation should be included in the metadata	abstract	optional	WG1 (21), WG4 (33)
32	Version must be included in the metadata description for all resources	abstract	mandatory	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
33	Licensing information must be included in the metadata	abstract	mandatory	WG1 (21), WG3 (23)
34	Licensing information should be expressed in a machine-readable form	abstract	recommended	WG1 (21), WG3 (23)
35	All resources must include a unique persistent identifier	abstract	mandatory	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
36	Classification metadata should be included, where applicable, in the metadata record of the resource	abstract	recommended	WG1 (21), WG2 (17)
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	abstract	recommended	WG1 (21)
38	Access mode of resources must be included in the metadata	abstract	mandatory	WG1 (21), WG2 (17), WG4 (33)
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	abstract	mandatory	WG1 (21)
40	Component metadata must include standardised categories/tags that make them easy to discover	abstract	mandatory	WG1 (21), WG4 (33)
41	Content resources must include metadata on their language(s)	abstract	mandatory	WG1 (21), WG2 (17)
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 (21), WG4 (33)
44	Statistical metadata that allow monitoring of resource versions may accompany resources	abstract	optional	WG1 (21), WG2 (17)

ID	Requirement	Concreteness	Strength	WG's
45	S/W (tools, web services, workflows) must indicate format of their output	abstract	mandatory	WG1 (21), WG4 (33)
47	Information on funding of resources may be included in the metadata	abstract	optional	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
48	All resource metadata records must include a reference to the metadata schema used for their description	abstract	mandatory	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
50	Documentation references should be versioned	abstract	recommended	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
67	Knowledge Resource Element Id	abstract	recommended	WG2 (17)
68	Data Category Linking Vocabulary	abstract	recommended	WG2 (17)
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	abstract	recommended	WG2 (17)
70	All KR content elements need to be added as text annotations within a TDM workflow.	abstract	mandatory	WG2 (17)
71	The KR should be ingestible through a URI	abstract	recommended	WG2 (17)
72	The KR format should be in a standard format such as XML, JSON or RDF.	abstract	recommended	WG2 (17)

By Strength

mandatory (25)

ID	Requirement	Concreteness	Status	WG's
1	Components should be described by machine-readable metadata	abstract	final	WG4 (33)
3	Component metadata is separable from the component	abstract	final	WG4 (33)
4	URL to actual content must be discoverable	abstract	final	WG1 (21), WG2 (17), WG3 (23)
5	Components should detail all their environmental requirements for execution	abstract	draft	WG4 (33)
6	Components should have a unique identifier and a version number	abstract	draft	WG4 (33)
7	Components should have a fully qualified name that follows the Java class naming conventions	concrete	final	WG4 (33)
8	Components should associate themselves with categories defined by the OpenMinTeD project	abstract	final	WG4 (33)
9	Components should declare their annotation schema dependencies	abstract	final	WG4 (33)
10	Components should specify the types of the annotations that they input and output	abstract	draft	WG4 (33), WG2 (17)
11	Components should declare whether they can be scaled within a workflow	abstract	draft	WG4 (33)
14	Components must maintain License information	abstract	draft	WG4 (33)
24	Using/treating workflows as components	abstract	final	WG4 (33)
29	The actual content of all content resources must be discoverable	abstract	deprecated	WG1 (21), WG2 (17), WG3 (23)
32	Version must be included in the metadata description for all resources	abstract	final	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
33	Licensing information must be included in the metadata	abstract	final	WG1 (21), WG3 (23)

ID	Requirement	Concreteness	Status	WG's
35	All resources must include a unique persistent identifier	abstract	final	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
38	Access mode of resources must be included in the metadata	abstract	final	WG1 (21), WG2 (17), WG4 (33)
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	abstract	final	WG1 (21)
40	Component metadata must include standardised categories/tags that make them easy to discover	abstract	final	WG1 (21), WG4 (33)
41	Content resources must include metadata on their language(s)	abstract	final	WG1 (21), WG2 (17)
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 (21), WG4 (33)
45	S/W (tools, web services, workflows) must indicate format of their output	abstract	final	WG1 (21), WG4 (33)
46	Output resources of web services/workflows must be accompanied by provenance metadata	abstract	deprecated	WG1 (21), WG4 (33)
48	All resource metadata records must include a reference to the metadata schema used for their description	abstract	final	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
70	All KR content elements need to be added as text annotations within a TDM workflow.	abstract	final	WG2 (17)

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 (17), WG4 (33)
30	Metrics for the confidence level of the TDM operation should be included in the metadata	abstract	final	WG1 (21), WG4 (33)

ID	Requirement	Concreteness	Status	WG's
31	Metrics for the performance of the TDM operation should be included in the metadata	abstract	final	WG1 (21), WG4 (33)
42	The metadata can include the information on which projects/workflows involve the resource	abstract	deprecated	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
44	Statistical metadata that allow monitoring of resource versions may accompany resources	abstract	final	WG1 (21), WG2 (17)
47	Information on funding of resources may be included in the metadata	abstract	final	WG1 (21), WG2 (17), WG3 (23), WG4 (33)

recommended (41)

ID	Requirement	Concreteness	Status	WG's
2	Component metadata should be embedded into the component source code	abstract	final	WG4 (33)
12	Components should provide documentation describing their functionality	abstract	final	WG4 (33)
13	Citation information for component	abstract	draft	WG1 (21), WG4 (33)
15	Human readable information should be provided by each resource	abstract	final	WG1 (21), WG4 (33)
16	Models/resources should be useable across different component collections/platforms	abstract	final	WG4 (33)
17	Components should be stateless	concrete	final	WG4 (33)
18	Workflows should be described using an uniform language	abstract	draft	WG4 (33)
20	Workflow engines should not require to see data	concrete	deprecated	WG2 (17), WG4 (33)
21	Configuration and parametrizable options of the components should be identified and documented	abstract	final	WG4 (33)
22	The Workflow Engine Should Permit Saving Experimental Conditions in a Workflow	abstract	deprecated	WG1 (21), WG4 (33)

ID	Requirement	Concreteness	Status	WG's
23	The Workflow Engine should permit Licence Aggregation in Workflows	abstract	deprecated	WG3 (23), WG4 (33)
25	Incorporation of multiple resources in parallel	abstract	deprecated	WG4 (33)
26	Ability to determine source of an annotation/assigned category	abstract	final	WG4 (33)
27	Components should handle failures gracefully	abstract	final	WG4 (33)
28	Processing components should be downloadable	abstract	final	WG4 (33)
34	Licensing information should be expressed in a machine-readable form	abstract	final	WG1 (21), WG3 (23)
36	Classification metadata should be included, where applicable, in the metadata record of the resource	abstract	final	WG1 (21), WG2 (17)
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	abstract	final	WG1 (21)
49	Metadata of tools should contain information about the models available for them	abstract	deprecated	WG1 (21), WG4 (33)
50	Documentation references should be versioned	abstract	final	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
51	License should be attached	abstract	draft	WG3 (23)
52	License information must be in metadata	abstract	deprecated	WG1 (21), WG3 (23)
53	Licensor must be entitled to grant license	abstract	draft	WG3 (23)
54	Licensees should remain with a copy of the license	abstract	draft	WG3 (23)
55	Standard licenses should be used	abstract	draft	WG3 (23)
56	License should be machine readable	abstract	draft	WG3 (23)
57	License should be understandable by non-lawyers	abstract	draft	WG3 (23)
58	TDM must be explicitly allowed	abstract	draft	WG3 (23)

ID	Requirement	Concreteness	Status	WG's
59	Right for (temporary) reproduction must be granted	abstract	draft	WG3 (23)
60	Boundary for derivative work must be clearly defined	abstract	draft	WG3 (23)
61	No restrictions on TDM results which are not derived works	abstract	draft	WG3 (23)
62	World-wide and irrevocable license grant	abstract	draft	WG3 (23)
63	License must qualify for Open Access rights	abstract	draft	WG3 (23)
64	License must qualify for Open Access uses	abstract	draft	WG3 (23)
65	License must qualify for Open Access must not restrict use in any way	abstract	draft	WG3 (23)
66	License must qualify for Open Access may include attribution requirements	abstract	draft	WG3 (23)
67	Knowledge Resource Element Id	abstract	final	WG2 (17)
68	Data Category Linking Vocabulary	abstract	final	WG2 (17)
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	abstract	final	WG2 (17)
71	The KR should be ingestible through a URI	abstract	final	WG2 (17)
72	The KR format should be in a standard format such as XML, JSON or RDF.	abstract	final	WG2 (17)

By Concreteness

abstract (69)

ID	Requirement	Status	Strength	WG's
1	Components should be described by machine-readable metadata	final	mandatory	WG4 (33)
2	Component metadata should be embedded into the component source code	final	recommended	WG4 (33)
3	Component metadata is separable from the component	final	mandatory	WG4 (33)
4	URL to actual content must be discoverable	final	mandatory	WG1 (21), WG2 (17), WG3 (23)
5	Components should detail all their environmental requirements for execution	draft	mandatory	WG4 (33)
6	Components should have a unique identifier and a version number	draft	mandatory	WG4 (33)
8	Components should associate themselves with categories defined by the OpenMinTeD project	final	mandatory	WG4 (33)
9	Components should declare their annotation schema dependencies	final	mandatory	WG4 (33)
10	Components should specify the types of the annotations that they input and output	draft	mandatory	WG4 (33), WG2 (17)
11	Components should declare whether they can be scaled within a workflow	draft	mandatory	WG4 (33)
12	Components should provide documentation describing their functionality	final	recommended	WG4 (33)
13	Citation information for component	draft	recommended	WG1 (21), WG4 (33)
14	Components must maintain License information	draft	mandatory	WG4 (33)
15	Human readable information should be provided by each resource	final	recommended	WG1 (21), WG4 (33)
16	Models/resources should be useable across different component collections/platforms	final	recommended	WG4 (33)
18	Workflows should be described using an uniform language	draft	recommended	WG4 (33)

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 (17), WG4 (33)
21	Configuration and parametrizable options of the components should be identified and documented	final	recommended	WG4 (33)
22	The Workflow Engine Should Permit Saving Experimental Conditions in a Workflow	deprecated	recommended	WG1 (21), WG4 (33)
23	The Workflow Engine should permit Licence Aggregation in Workflows	deprecated	recommended	WG3 (23), WG4 (33)
24	Using/treating workflows as components	final	mandatory	WG4 (33)
25	Incorporation of multiple resources in parallel	deprecated	recommended	WG4 (33)
26	Ability to determine source of an annotation/assigned category	final	recommended	WG4 (33)
27	Components should handle failures gracefully	final	recommended	WG4 (33)
28	Processing components should be downloadable	final	recommended	WG4 (33)
29	The actual content of all content resources must be discoverable	deprecated	mandatory	WG1 (21), WG2 (17), WG3 (23)
30	Metrics for the confidence level of the TDM operation should be included in the metadata	final	optional	WG1 (21), WG4 (33)
31	Metrics for the performance of the TDM operation should be included in the metadata	final	optional	WG1 (21), WG4 (33)
32	Version must be included in the metadata description for all resources	final	mandatory	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
33	Licensing information must be included in the metadata	final	mandatory	WG1 (21), WG3 (23)
34	Licensing information should be expressed in a machine-readable form	final	recommended	WG1 (21), WG3 (23)

ID	Requirement	Status	Strength	WG's
35	All resources must include a unique persistent identifier	final	mandatory	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
36	Classification metadata should be included, where applicable, in the metadata record of the resource	final	recommended	WG1 (21), WG2 (17)
37	Information on the structural annotation (layout) of resources should be included in the metadata of the resource	final	recommended	WG1 (21)
38	Access mode of resources must be included in the metadata	final	mandatory	WG1 (21), WG2 (17), WG4 (33)
39	Content resources must include metadata on their format (e.g. XML, DOCX etc.)	final	mandatory	WG1 (21)
40	Component metadata must include standardised categories/tags that make them easy to discover	final	mandatory	WG1 (21), WG4 (33)
41	Content resources must include metadata on their language(s)	final	mandatory	WG1 (21), WG2 (17)
42	The metadata can include the information on which projects/workflows involve the resource	deprecated	optional	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
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 (21), WG4 (33)
44	Statistical metadata that allow monitoring of resource versions may accompany resources	final	optional	WG1 (21), WG2 (17)
45	S/W (tools, web services, workflows) must indicate format of their output	final	mandatory	WG1 (21), WG4 (33)
46	Output resources of web services/workflows must be accompanied by provenance metadata	deprecated	mandatory	WG1 (21), WG4 (33)
47	Information on funding of resources may be included in the metadata	final	optional	WG1 (21), WG2 (17), WG3 (23), WG4 (33)

ID	Requirement	Status	Strength	WG's
48	All resource metadata records must include a reference to the metadata schema used for their description	final	mandatory	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
49	Metadata of tools should contain information about the models available for them	deprecated	recommended	WG1 (21), WG4 (33)
50	Documentation references should be versioned	final	recommended	WG1 (21), WG2 (17), WG3 (23), WG4 (33)
51	License should be attached	draft	recommended	WG3 (23)
52	License information must be in metadata	deprecated	recommended	WG1 (21), WG3 (23)
53	Licensor must be entitled to grant license	draft	recommended	WG3 (23)
54	Licensees should remain with a copy of the license	draft	recommended	WG3 (23)
55	Standard licenses should be used	draft	recommended	WG3 (23)
56	License should be machine readable	draft	recommended	WG3 (23)
57	License should be understandable by non-lawyers	draft	recommended	WG3 (23)
58	TDM must be explicitly allowed	draft	recommended	WG3 (23)
59	Right for (temporary) reproduction must be granted	draft	recommended	WG3 (23)
60	Boundary for derivative work must be clearly defined	draft	recommended	WG3 (23)
61	No restrictions on TDM results which are not derived works	draft	recommended	WG3 (23)
62	World-wide and irrevocable license grant	draft	recommended	WG3 (23)
63	License must qualify for Open Access rights	draft	recommended	WG3 (23)
64	License must qualify for Open Access uses	draft	recommended	WG3 (23)

ID	Requirement	Status	Strength	WG's
65	License must qualify for Open Access must not restrict use in any way	draft	recommended	WG3 (23)
66	License must qualify for Open Access may include attribution requirements	draft	recommended	WG3 (23)
67	Knowledge Resource Element Id	final	recommended	WG2 (17)
68	Data Category Linking Vocabulary	final	recommended	WG2 (17)
69	Interoperability between elements from different knowledge resource schemas should be expressed through RDF statements.	final	recommended	WG2 (17)
70	All KR content elements need to be added as text annotations within a TDM workflow.	final	mandatory	WG2 (17)
71	The KR should be ingestible through a URI	final	recommended	WG2 (17)
72	The KR format should be in a standard format such as XML, JSON or RDF.	final	recommended	WG2 (17)

concrete (3)

ID	Requirement	Status	Strength	WG's
7	Components should have a fully qualified name that follows the Java class naming conventions	final	mandatory	WG4 (33)
17	Components should be stateless	final	recommended	WG4 (33)
20	Workflow engines should not require to see data	deprecated	recommended	WG2 (17), WG4 (33)

Compliance

By Product

ARGO (35)

Compliance	#	%
Full	6	17
No	12	34
Partial	17	49

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

ID	Requirement	Compliance
30	Metrics for the confidence level of the TDM operation should be included in the metadata	No
31	Metrics for the performance of the TDM operation should be included in the metadata	No
32	Version must be included in the metadata description for all resources	No
33	Licensing information must be included in the metadata	Partial
35	All resources must include a unique persistent identifier	Full
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
40	Component metadata must include standardised categories/tags that make them easy to discover	Partial
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
48	All resource metadata records must include a reference to the metadata schema used for their description	Partial
50	Documentation references should be versioned	No

Agrovoc (16)

Compliance	#	%
Full	12	75
No	3	19
Partial	1	6

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
32	Version must be included in the metadata description for all resources	Partial
33	Licensing information must be included in the metadata	Full
35	All resources must include a unique persistent identifier	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

ID	Requirement	Compliance
47	Information on funding of resources may be included in the metadata	No
48	All resource metadata records must include a reference to the metadata schema used for their description	Full
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 or RDF.	Full

Alvis (35)

Compliance	#	%
Full	5	14
No	19	54
Partial	11	31

ID	Requirement	Compliance
1	Components should be described by machine-readable metadata	Full
2	Component metadata should be embedded into the component source code	No
3	Component metadata is separable from the component	Full
5	Components should detail all their environmental requirements for execution	Partial
6	Components should have a unique identifier and a version number	Partial
7	Components should have a fully qualified name that follows the Java class naming conventions	Full
8	Components should associate themselves with categories defined by the OpenMinTeD project	Partial
9	Components should declare their annotation schema dependencies	No
10	Components should specify the types of the annotations that they input and output	Partial
11	Components should declare whether they can be scaled within a workflow	No
12	Components should provide documentation describing their functionality	Partial
13	Citation information for component	No
14	Components must maintain License information	No
15	Human readable information should be provided by each resource	Partial

ID	Requirement	Compliance
16	Models/resources should be useable across different component collections/platforms	Partial
17	Components should be stateless	Partial
18	Workflows should be described using an uniform language	Partial
21	Configuration and parametrizable options of the components should be identified and documented	Full
24	Using/treating workflows as components	Full
26	Ability to determine source of an annotation/assigned category	Partial
27	Components should handle failures gracefully	No
28	Processing components should be downloadable	No
30	Metrics for the confidence level of the TDM operation should be included in the metadata	No
31	Metrics for the performance of the TDM operation should be included in the metadata	No
32	Version must be included in the metadata description for all resources	No
33	Licensing information must be included in the metadata	No
35	All resources must include a unique persistent identifier	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	No
40	Component metadata must include standardised categories/tags that make them easy to discover	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
48	All resource metadata records must include a reference to the metadata schema used for their description	No
50	Documentation references should be versioned	No

CLARIN CCR (5)

Compliance	#	%
Full	2	40
No	3	60

ID	Requirement	Compliance
67	Knowledge Resource Element Id	Full

ID	Requirement	Compliance
68	Data Category Linking Vocabulary	No
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	No
72	The KR format should be in a standard format such as XML, JSON or RDF.	Full

CORE (11)

Compliance	#	%
Full	2	18
Partial	9	82

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Partial
32	Version must be included in the metadata description for all resources	Partial
33	Licensing information must be included in the metadata	Partial
35	All resources must include a unique persistent identifier	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	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
48	All resource metadata records must include a reference to the metadata schema used for their description	Partial

DKPro Core (36)

Compliance	#	%
Full	15	42
N/A	2	6
No	5	14
Partial	13	36
Unknown	1	3

ID	Requirement	Compliance
1	Components should be described by machine-readable metadata	Full
2	Component metadata should be embedded into the component source code	Full
3	Component metadata is separable from the component	Full
5	Components should detail all their environmental requirements for execution	Partial
6	Components should have a unique identifier and a version number	Partial
7	Components should have a fully qualified name that follows the Java class naming conventions	Full
8	Components should associate themselves with categories defined by the OpenMinTeD project	No
9	Components should declare their annotation schema dependencies	Partial
10	Components should specify the types of the annotations that they input and output	Full
11	Components should declare whether they can be scaled within a workflow	Full
12	Components should provide documentation describing their functionality	Full
13	Citation information for component	No
14	Components must maintain License information	Partial
15	Human readable information should be provided by each resource	Full
16	Models/resources should be useable across different component collections/platforms	Full
17	Components should be stateless	Partial
18	Workflows should be described using an uniform language	Unknown
21	Configuration and parametrizable options of the components should be identified and documented	Full
24	Using/treating workflows as components	N/A
26	Ability to determine source of an annotation/assigned category	Partial
27	Components should handle failures gracefully	N/A
28	Processing components should be downloadable	Full
30	Metrics for the confidence level of the TDM operation should be included in the metadata	No
31	Metrics for the performance of the TDM operation should be included in the metadata	No
32	Version must be included in the metadata description for all resources	Full
33	Licensing information must be included in the metadata	Partial
34	Licensing information should be expressed in a machine-readable form	Partial
35	All resources must include a unique persistent identifier	Full
36	Classification metadata should be included, where applicable, in the metadata record of the resource	Partial

ID	Requirement	Compliance
38	Access mode of resources must be included in the metadata	Full
40	Component metadata must include standardised categories/tags that make them easy to discover	Partial
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
48	All resource metadata records must include a reference to the metadata schema used for their description	Partial
50	Documentation references should be versioned	Full

Frontiers (11)

Compliance	#	%
Full	7	64
Partial	4	36

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
32	Version must be included in the metadata description for all resources	Partial
33	Licensing information must be included in the metadata	Full
35	All resources must include a unique persistent identifier	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
48	All resource metadata records must include a reference to the metadata schema used for their description	Partial

GATE (35)

Compliance	#	%
Full	11	31

Compliance	#	%
No	12	34
Partial	10	29
Unknown	1	3
no	1	3

ID	Requirement	Compliance
1	Components should be described by machine-readable metadata	Full
2	Component metadata should be embedded into the component source code	Partial
3	Component metadata is separable from the component	Partial
5	Components should detail all their environmental requirements for execution	Partial
6	Components should have a unique identifier and a version number	Partial
7	Components should have a fully qualified name that follows the Java class naming conventions	Full
8	Components should associate themselves with categories defined by the OpenMinTeD project	no
9	Components should declare their annotation schema dependencies	No
10	Components should specify the types of the annotations that they input and output	Partial
11	Components should declare whether they can be scaled within a workflow	Full
12	Components should provide documentation describing their functionality	Full
13	Citation information for component	No
14	Components must maintain License information	No
15	Human readable information should be provided by each resource	Full
16	Models/resources should be useable across different component collections/platforms	Full
17	Components should be stateless	No
18	Workflows should be described using an uniform language	Unknown
21	Configuration and parametrizable options of the components should be identified and documented	Full
24	Using/treating workflows as components	Full
26	Ability to determine source of an annotation/assigned category	Partial
27	Components should handle failures gracefully	No
28	Processing components should be downloadable	Full
30	Metrics for the confidence level of the TDM operation should be included in the metadata	Partial
31	Metrics for the performance of the TDM operation should be included in the metadata	Partial

ID	Requirement	Compliance
32	Version must be included in the metadata description for all resources	No
33	Licensing information must be included in the metadata	Partial
35	All resources must include a unique persistent identifier	Full
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
40	Component metadata must include standardised categories/tags that make them easy to discover	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
48	All resource metadata records must include a reference to the metadata schema used for their description	No
50	Documentation references should be versioned	No

ILSP (13)

Compliance	#	%
Full	1	8
No	5	38
Partial	7	54

ID	Requirement	Compliance
30	Metrics for the confidence level of the TDM operation should be included in the metadata	Partial
31	Metrics for the performance of the TDM operation should be included in the metadata	No
32	Version must be included in the metadata description for all resources	Partial
33	Licensing information must be included in the metadata	Partial
35	All resources must include a unique persistent identifier	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
40	Component metadata must include standardised categories/tags that make them easy to discover	Partial
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

ID	Requirement	Compliance
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
48	All resource metadata records must include a reference to the metadata schema used for their description	No
50	Documentation references should be versioned	No

JATS (15)

Compliance	#	%
Full	5	33
No	7	47
Partial	3	20

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Partial
32	Version must be included in the metadata description for all resources	Full
33	Licensing information must be included in the metadata	Partial
35	All resources must include a unique persistent identifier	Full
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
48	All resource metadata records must include a reference to the metadata schema used for their description	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 or RDF.	Partial

LAPPS (15)

Compliance	#	%
Full	7	47
No	8	53

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
32	Version must be included in the metadata description for all resources	No
33	Licensing information must be included in the metadata	No
35	All resources must include a unique persistent identifier	Full
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
48	All resource metadata records must include a reference to the metadata schema used for their description	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 or RDF.	Full

Licences (6)

Compliance	#	%
Full	3	50
No	1	17
Partial	2	33

ID	Requirement	Compliance
32	Version must be included in the metadata description for all resources	Full
33	Licensing information must be included in the metadata	Full
34	Licensing information should be expressed in a machine-readable form	Partial
35	All resources must include a unique persistent identifier	Full
47	Information on funding of resources may be included in the metadata	No
50	Documentation references should be versioned	Partial

OLiA (15)

Compliance	#	%
Full	8	53
No	6	40
Partial	1	7

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
32	Version must be included in the metadata description for all resources	Partial
33	Licensing information must be included in the metadata	No
35	All resources must include a unique persistent identifier	Full
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
48	All resource metadata records must include a reference to the metadata schema used for their description	Full
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 or RDF.	Full

Ontolex (5)

Compliance	#	%
Full	4	80
No	1	20

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 or RDF.	Full

OpenAIRE (11)

Compliance	#	%
Full	2	18
No	2	18
Partial	7	64

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Partial
32	Version must be included in the metadata description for all resources	No
33	Licensing information must be included in the metadata	Partial
35	All resources must include a unique persistent identifier	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
48	All resource metadata records must include a reference to the metadata schema used for their description	Full

TheSoz (16)

Compliance	#	%
Full	12	75
No	3	19
Partial	1	6

ID	Requirement	Compliance
4	URL to actual content must be discoverable	Full
32	Version must be included in the metadata description for all resources	Full
33	Licensing information must be included in the metadata	Full
35	All resources must include a unique persistent identifier	No
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

ID	Requirement	Compliance
41	Content resources must include metadata on their language(s)	Full
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
48	All resource metadata records must include a reference to the metadata schema used for their description	Full
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 or RDF.	Full

UIMA (1)

Compliance	#	%
Full	1	100

ID	Requirement	Compliance
24	Using/treating workflows as components	Full

schema.org (5)

Compliance	#	%
Full	4	80
No	1	20

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 or RDF.	Full

Without justification

ID	Requirement	Product	Compliance
4	URL to actual content must be discoverable	Agrovoc	Full
4	URL to actual content must be discoverable	OLiA	Full
4	URL to actual content must be discoverable	LAPPS	Full
19	Components that use external knowledge resources should delegate access to a resource adapter instead of handling it themselves	Alvis	Unknown
19	Components that use external knowledge resources should delegate access to a resource adapter instead of handling it themselves	GATE	Unknown
29	The actual content of all content resources must be discoverable	Agrovoc	Full
29	The actual content of all content resources must be discoverable	OLiA	Full
29	The actual content of all content resources must be discoverable	LAPPS	Full
32	Version must be included in the metadata description for all resources	LAPPS	No
33	Licensing information must be included in the metadata	OLiA	No
33	Licensing information must be included in the metadata	LAPPS	No
35	All resources must include a unique persistent identifier	TheSoz	No
35	All resources must include a unique persistent identifier	Agrovoc	Full
35	All resources must include a unique persistent identifier	OLiA	Full
35	All resources must include a unique persistent identifier	LAPPS	Full
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
40	Component metadata must include standardised categories/tags that make them easy to discover	GATE	No
41	Content resources must include metadata on their language(s)	Agrovoc	No
42	The metadata can include the information on which projects/workflows involve the resource	TheSoz	No
42	The metadata can include the information on which projects/workflows involve the resource	Agrovoc	No
42	The metadata can include the information on which projects/workflows involve the resource	JATS	No
42	The metadata can include the information on which projects/workflows involve the resource	OLiA	No
42	The metadata can include the information on which projects/workflows involve the resource	LAPPS	No
42	The metadata can include the information on which projects/workflows involve the resource	Licences	Unknown

ID	Requirement	Product	Compliance
42	The metadata can include the information on which projects/workflows involve the resource	GATE	No
42	The metadata can include the information on which projects/workflows involve the resource	ILSP	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
48	All resource metadata records must include a reference to the metadata schema used for their description	LAPPS	No
48	All resource metadata records must include a reference to the metadata schema used for their description	GATE	No
49	Metadata of tools should contain information about the models available for them	GATE	No
50	Documentation references should be versioned	TheSoz	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

ID	Requirement	Product	Compliance
67	Knowledge Resource Element Id	schema.org	Full
68	Data Category Linking Vocabulary	TheSoz	Full
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 or RDF.	Agrovoc	Full
72	The KR format should be in a standard format such as XML, JSON or RDF.	OLiA	Full
72	The KR format should be in a standard format such as XML, JSON or RDF.	LAPPS	Full
72	The KR format should be in a standard format such as XML, JSON or RDF.	Ontolex	Full
72	The KR format should be in a standard format such as XML, JSON or RDF.	CLARIN CCR	Full
72	The KR format should be in a standard format such as XML, JSON or RDF.	schema.org	Full

Requirements

1. Components should 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 should be embedded into the component source code

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

To avoid implementation and metadata getting out-of-sync, the metadata should 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 is 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		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

5. Components should 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 lunix 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 worklow 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 althougbh 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 explainnig 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 should 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. Experiance 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 should 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 use 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 techniquees, 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 using 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

Product	Version	Compliant	Justification	Status
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 should 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 should 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 progess.	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 specifiy annotation shemas, 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 should 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	We 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 sturctures 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	The description are free texts.	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

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 may sometimes contain 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: *draft*

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 agregate 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: *final*

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 ceate 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 releavant 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 caracterize 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 guaranted. 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 reequires multiple copies of internal state that can be shared. The framework supports inteligent 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: *draft*

A workflow represents an experiment that users describe, visualize, execute, modifify 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	Possibily 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 serives 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
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 gazeteer as an example, it was considered the main functionality of a specific gazeteer 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 gazetter 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 can 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 mulitvalued 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 beeen 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 idenfified (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 experiements, 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: *final*

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. Aggegate 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 arbritary 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 Gazeteer 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. Ability to determine 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 occurrance 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 propogate 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	No	Not supported	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: *final*

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: *final*

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	Perfomance is sometimes available for specific tools over specific collections, but not generally. Quoting perfomance 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: *final*

Version information must be included in the metadata 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 udpate 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
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

Product	Version	Compliant	Justification	Status
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 avaialable	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		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 tool descriptions or in some tool readme.	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: *final*

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

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
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

Product	Version	Compliant	Justification	Status
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 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: *final*

Component (web services & workflows but also of tools) classification & description must include metadata that describe the task they perform (e.g. anntotation, named entity recogntion), 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

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 reliablility, 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 ususally determines the language. Also for components that a 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 are limited	Final
ARGO	0.5	Partial	Components can optionally declare what types it 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 resour ce was created.	Final
LAPPS	Jun-16	No		Final
Licences	Jun-16	No		Final
Alvis	0.5rc	No	No funding information included in metatdata	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: *final*

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		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 vocabulariess 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*

If you want to work with a KR its relevant elements need to be added as text metadata. This enables the TDM modules to work in a uniform manner with the informational content the KRs provide. In order to obtain annotations you need to follow KR access routines such as XPATH/ontology based and gazetteer based lookup, and existing APIs/web services.

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 or RDF.

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

Standard formats enable standard ingest 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

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)