OPEN DATA PORTAL

Deliverable D3.2 Quality Analysis Component

Autoren: Adrian Wilke
Reviewer: Matthias Wauer

Veröffentlichung	Vertraulich
Fälligkeitsdatum	31.12.2018
Fertigstellung	29.01.2019 (aktuelle erweiterte Version)
Arbeitspaket	AP 3.1 Qualitätsanalyse
Typ	Software Dokumentation
Status	Final
Version	1.1

Abstract:
This deliverable describes the current architecture and implementation of the OPAL metadata quality analysis framework "Civet". In addition to the technical specifications, relevant previous works are described and used as foundations for the development.

Schlagworte:
Metadata, Quality Analysis, Software

Das Startkapital für die Mobilität 4.0

Bundesministerium für Verkehr und digitale Infrastruktur

Inhalt

Foundations **2**
Underlying works in the OPAL project 2
Current state of the project 2
Quality dimensions and metrics 3
Vocabulary and data structures 4
Data Acquisition 5
Analysis of available data 5
RDF concepts 5
Datasets: Structure 6
Datasets: Number of records 6
Distributions: Structure 7
Distributions: Number of records 7
Catalogs: Instances 7
Publisher/Agent: Structure 8
Publisher/Agent: Number of records 8
Architecture and implementation **9**
Datasets, distributions, and quality metrics 9
Implementation of required metric properties and metric result types 10
Civet architecture 11
Usage of the component **12**
Civet Java API 12
Civet Command line interface (CLI) 12
Outlook: Future development **13**
References **13**

1 Foundations

Foundations of the **OPAL quality analysis component "Civet"** consist of related OPAL components, data structures, and available data. The foundations are described in the following sections.

1.1 Underlying works in the OPAL project

This section gives an overview of OPAL components and deliverables, which form a basis of the quality analysis component.

1.1.1 Current state of the project

The OPAL Civet component has been implemented for batch-processing of metadata records. A specification of the integration of metrics and the respective required data fields is completed. Additionally, first metrics values for approximately **880,000 metadata records** (datasets) and their related **1,330,000 distributions** (available form of datasets, e.g. CSV or RSS) are computed and integrated into the OPAL RDF database.

All previous work packages have been processed. However, there are ongoing successive improvements to individual implementations of the existing components. Related delays resulted, inter alia, from the integration of additional data sources. The integration of the additional sources was not planned at the beginning of the project, but resulted in an extended database, which is also utilized in the Civet component. However, the additional workload in the past caused a delayed start and completion of the Civet component due to missing data and specifications. All of the OPAL components are continuously extended throughout the entire project period. The following tables presents the current states of initial component and specification versions:

Deliverable	Description of initial versions	Current state
D2.1	Crawler component	Completed on time
D3.1	Specification of quality criteria	Completed on time
D4.1	Specification of RDF vocabulary	Completed
D4.2	RDF conversion component	In progress
D3.2	Quality analysis component	Completed

Table: Current state of initial components (as of 2019-01-17)

1.1.2 Quality dimensions and metrics

The OPAL quality framework Civet implements the specifications of quality criteria, which were defined in [OPAL.D3.1]. The following 13 quality dimensions and 48 metrics have been worked out:

No.	Criterion	No.	Criterion	No.	Criterion
Expressiveness		Trust		Interlinking	
1	Extend	16	Provider identity	34	Labeled data
2	Weighted extend	17	Trusted provider	35	Linked data representation
3	Categorization	18	Metadata authenticity	36	Metadata interlinking
4	Description*	19	Usage of digital signatures	Contactability	
Temporal		Community		37	Contact URL
5	Timeliness	20	Communication	38	Contact Email
6	Update rate	21	Trust transfer	39	Classical contact information*
Understandability		22	Correctness	Access	
7	Readability	23	Confirmation	40	Open metadata
8	Language errors	Versatility		41	Retrievability
9	Example applications	24	Multiple serializations*	Versioning*	
Rights		25	Multiple languages	42	Version numbering*
10	Machine readable license*	26	Multiple access methods	43	Period of time*
11	Human readable license	Representation		Data	
12	Known License	27	Open format	44	Open data format
13	Open License	28	Data format	45	Data format
14	Permission for commercial use	29	Machine processable	46	Machine processable data
15	Permissions*	30	Vocabulary	47	Unique data identifier
		31	Date Format	48	Multiple data serializations
		32	Unique identifier		
		33	Locality*		

*Extensions of the literature review by the data-driven-approach

Table: Overview of OPAL quality dimensions and metrics

1.1.3 Vocabulary and data structures

The OPAL vocabulary was defined in [OPAL.D4.1]. The following standardized (RDF) vocabularies are re-used in the OPAL Civet component to describe, access, and store metadata records:

- **Dublin Core** [VocabDC]
- **Data Catalog Vocabulary (DCAT)** [VocabDCAT]
- **Friend of a Friend (FOAF)** [VocabFOAF]
- **Data Quality Vocabulary (DQV)** [VocabDQV]
- **Quality assessment for Linked Data (LDQD)** [VocabLDQD]
- **Simple Knowledge Organization System Namespace Document (SKOS)** [VocabSKOS]

In addition to the quality dimensions and metrics describes in the previous section, the following quality categories are used to combine individual quality dimensions:

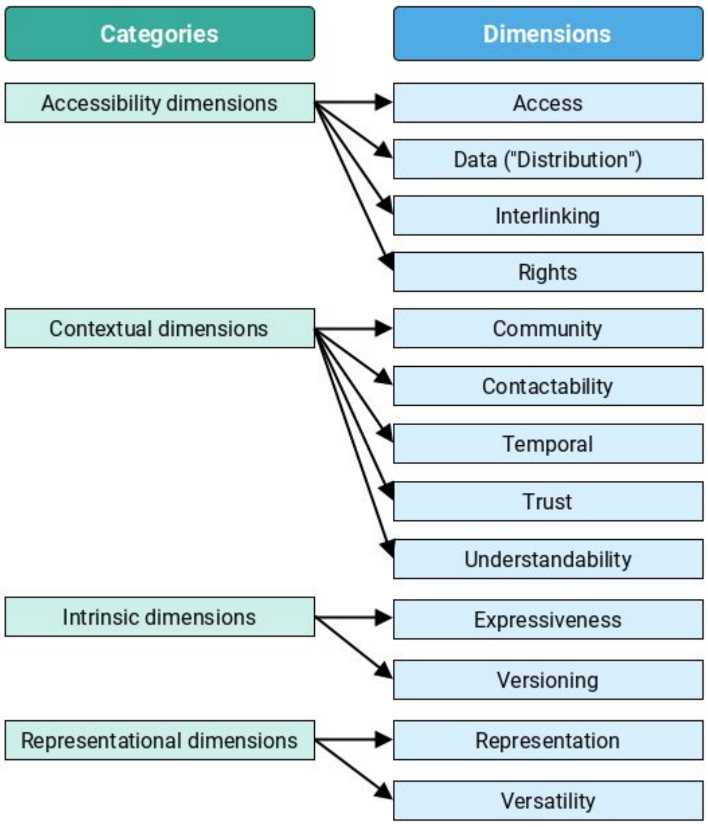

Figure: Top-level view of quality categories and dimensions

The vocabulary is utilized in the Civet component to efficiently access metadata fields and store metric values for metadata records. Currently available data predicates and datasets are analyzed in Sec. *Analysis of available data*.

1.1.4 Data Acquisition

Input data for the Civet component is first acquired by the crawler component Squirrel [OPAL.D2.1]. Second, the available data is converted by the component for conversion [OPAL.D4.2] according to the OPAL vocabulary [OPAL.D4.1]. The resulting structures and values are available in the OPAL TripleStore. The available prepared data is analyzed in the following section.

1.2 Analysis of available data

The quality component uses iterative data-driven development. To implement individual metrics, currently available types of data are analyzed to be utilized afterwards. The data depends on the implementation of precedent components and processed data sources.

RDF concepts

The most general data types are **concepts**. These represent basic object types, which provide data structures to be utilized. Main data sources are represented as dcat:Catalog, which refer to metadata records (dcat:Dataset). These records can be available in different data formats (dcat:Distribution). The following concepts are currently available:

```
SELECT DISTINCT ?concept FROM <http://projekt-opal.de> WHERE { [] a ?concept} ORDER BY
?concept
--------------------------------------------------
| concept                                        |
==================================================
| <foaf:Agent>                                   |
| <http://purl.org/dc/terms/Frequency>           |
| <http://purl.org/dc/terms/LinguisticSystem>    |
| <http://purl.org/dc/terms/Location>            |
| <http://purl.org/dc/terms/PeriodOfTime>        |
| <http://purl.org/dc/terms/ProvenanceStatement> |
| <http://purl.org/dc/terms/Standard>            |
| <http://www.w3.org/2006/vcard/ns#Kind>         |
| <http://www.w3.org/2006/vcard/ns#Organization> |
| <http://www.w3.org/2006/vcard/ns#VCard>        |
| <http://www.w3.org/ns/dcat#Catalog>            |
| <http://www.w3.org/ns/dcat#Dataset>            |
| <http://www.w3.org/ns/dcat#Distribution>       |
| <http://xmlns.com/foaf/0.1/Agent>              |
| <http://xmlns.com/foaf/0.1/Organization>       |
| <http://xmlns.com/foaf/0.1/Person>             |
| <vcard:Kind>                                   |
| "skos:Concept"                                 |
--------------------------------------------------
```

Listing: Available RDF concepts (as of January 2019)

Datasets: Structure

The most important concept is **dcat:Dataset**. It provides basic metadata fields (i.e. title, description, license), which can be processed to compute quality metrics. The following predicates are currently available for datasets:

```
SELECT DISTINCT ?predicate  FROM <http://projekt-opal.de> WHERE { ?s a
<http://www.w3.org/ns/dcat#Dataset> . ?s ?predicate ?o } ORDER BY ?predicate
-----------------------------------------------------
| predicate                                         |
=====================================================
| <http://purl.org/dc/terms/accrualPeriodicity>     |
| <http://purl.org/dc/terms/conformsTo>             |
| <http://purl.org/dc/terms/description>            |
| <http://purl.org/dc/terms/identifier>             |
| <http://purl.org/dc/terms/issued>                 |
| <http://purl.org/dc/terms/language>               |
| <http://purl.org/dc/terms/license>                |
| <http://purl.org/dc/terms/modified>               |
| <http://purl.org/dc/terms/provenance>             |
| <http://purl.org/dc/terms/publisher>              |
| <http://purl.org/dc/terms/relation>               |
| <http://purl.org/dc/terms/spatial>                |
| <http://purl.org/dc/terms/temporal>               |
| <http://purl.org/dc/terms/title>                  |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> |
| <http://www.w3.org/2002/07/owl#versionInfo>       |
| <http://www.w3.org/ns/adms#identifier>            |
| <http://www.w3.org/ns/dcat#contactPoint>          |
| <http://www.w3.org/ns/dcat#distribution>          |
| <http://www.w3.org/ns/dcat#keyword>               |
| <http://www.w3.org/ns/dcat#landingPage>           |
| <http://www.w3.org/ns/dcat#theme>                 |
| <http://xmlns.com/foaf/0.1/page>                  |
-----------------------------------------------------
```

Listing: Available predicates for the Subject dcat:Dataset (as of January 2019)

Datasets: Number of records

Besides the structure of datasets, the amount of available metadata records for datasets is important to provide an extensive catalog of evaluated metadata. At the current state, approximately **880,000 datasets** are available in the OPAL repository:

```
SELECT DISTINCT (COUNT(?dataset) as ?datasets)  FROM <http://projekt-opal.de> WHERE { ?dataset
a <http://www.w3.org/ns/dcat#Dataset> }
------------
| datasets |
============
| 879550   |
------------
```

Listing: Number of available datasets (as of 2019-01-16)

Distributions: Structure

The concept dcat:Distribution represents accessible data of dcat:Datasets in different formats. The following predicates are currently available:

```
SELECT DISTINCT ?predicate  FROM <http://projekt-opal.de> WHERE { ?s a
<http://www.w3.org/ns/dcat#Distribution> . ?s ?predicate ?o } ORDER BY ?predicate
-----------------------------------------------------
| predicate                                         |
=====================================================
| <http://purl.org/dc/terms/conformsTo>             |
| <http://purl.org/dc/terms/description>            |
| <http://purl.org/dc/terms/format>                 |
| <http://purl.org/dc/terms/issued>                 |
| <http://purl.org/dc/terms/language>               |
| <http://purl.org/dc/terms/license>                |
| <http://purl.org/dc/terms/modified>               |
| <http://purl.org/dc/terms/rights>                 |
| <http://purl.org/dc/terms/title>                  |
| <http://spdx.org/rdf/terms#checksum>              |
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> |
| <http://www.w3.org/ns/dcat#accessURL>             |
| <http://www.w3.org/ns/dcat#byteSize>              |
| <http://www.w3.org/ns/dcat#downloadURL>           |
| <http://www.w3.org/ns/dcat#mediaType>             |
-----------------------------------------------------
```

Listing: Available predicates for the Subject dcat:Distribution (as of 2019-01-21)

Distributions: Number of records

At the current state, approximately **1,330,000 distributions** are stored in the OPAL database:

```
SELECT DISTINCT (COUNT(?distribution) as ?distributions)  FROM <http://projekt-opal.de> WHERE
{ ?distribution a <http://www.w3.org/ns/dcat#Distribution> }
-----------------
| distributions |
=================
| 1332779       |
-----------------
```

Listing: Number of available distributions (as of 2019-01-21)

Catalogs: Instances

Instances of the type dcat:Catalog represent collections of dataset metadata records. Currently, three individual catalogs were integrated to the OPAL database.

```
SELECT DISTINCT ?catalog  FROM <http://projekt-opal.de> WHERE { ?catalog a
<http://www.w3.org/ns/dcat#Catalog> } LIMIT 5
-------------------------------------------------------
| catalog                                             |
=======================================================
| <http://projekt-opal.de/catalog/mcloud>             |
| <http://projekt-opal.de/catalog/govdata>            |
| <http://projekt-opal.de/catalog/europeandataportal> |
-------------------------------------------------------
```

Listing: Instances of type catalog (as of 2019-01-21)

Publisher/Agent: Structure

The concept foaf:Agent is used to represent contact information of publishers (data providers)

```
SELECT DISTINCT ?predicate  FROM <http://projekt-opal.de> WHERE { ?s a
<http://xmlns.com/foaf/0.1/Agent> . ?s ?predicate ?o } ORDER BY ?predicate
-----------------------------------------------------
| predicate                                         |
=====================================================
| <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> |
| <http://xmlns.com/foaf/0.1/homepage>              |
| <http://xmlns.com/foaf/0.1/mbox>                  |
| <http://xmlns.com/foaf/0.1/name>                  |
-----------------------------------------------------
```

Listing: Available predicates for the Subject foaf:Agent (as of 2019-01-29)

Publisher/Agent: Number of records

At the current state, approximately **200,000 agents** are available in the OPAL repository:

```
SELECT DISTINCT (COUNT(?agent) as ?agents)  FROM <http://projekt-opal.de> WHERE { ?agent a
<http://xmlns.com/foaf/0.1/Agent> }
----------
| agents |
==========
| 197553 |
----------
```

Listing: Number of available agents (as of 2019-01-29)

2 Architecture and implementation

This section describes the relationship among datasets, distributions, and resulting quality metrics in the vocabulary. Afterwards, the implementation concept of required properties to calculate metrics as well as the resulting value types are presented. Finally, a top-level view on the architecture is described.

2.1 Datasets, distributions, and quality metrics

The computation of quality metrics depends on data of the underlying *datasets* and *distributions*. If, for instance, titles are required to evaluate values for a metric, titles of both, dataset and its related distributions, are integrated. A computed result of a metric computation is stored as *dqv:QualityMeasurement* and categorized by the related quality *metric*, its *dimension*, and *category*. The following figure shows an example of a data instance on the right side and used concepts on the left side:

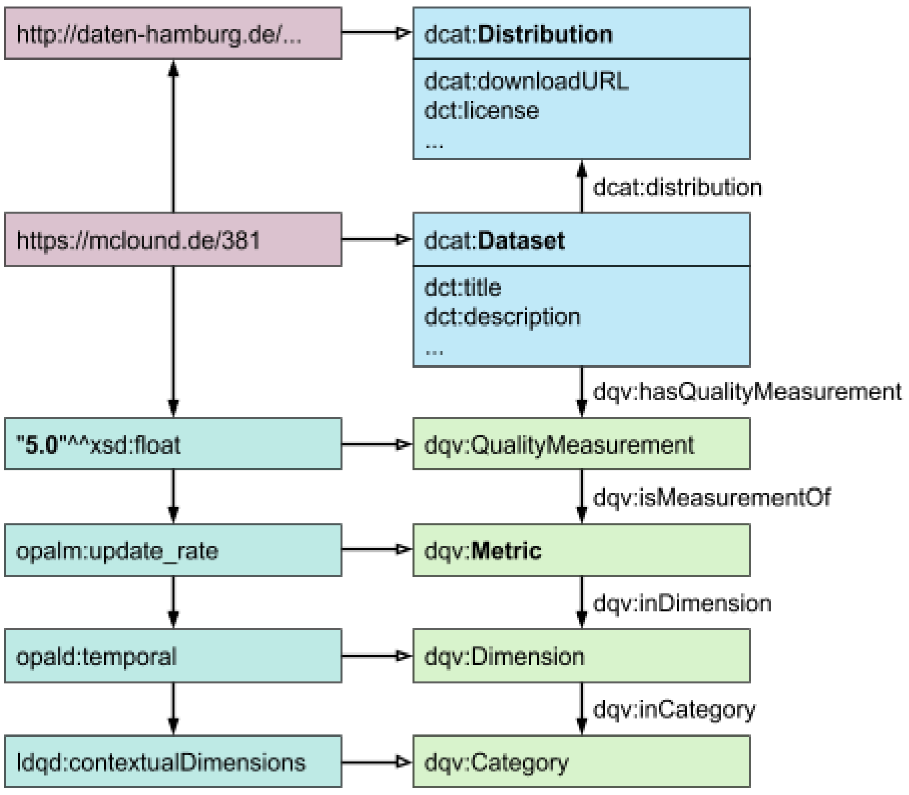

Figure: Example of data instances and types specified in the OPAL RDF vocabulary

2.2 Implementation of required metric properties and metric result types

The classification of individual data properties is implemented by instances of the class *DataObject*. A *DataObject* has a type (e.g. String or Integer), can contain multiple values (e.g. a list of Strings), and has an ID (e.g. TITLE). The IDs are based on the analysis (see above) of available data and vocabulary entries. IDs are defined as Java constants in an abstract class, as presented on the left side in the following figure. They are typically defined and mapped with regard to available properties in the OPAL TripleStore.

Each metric provides a list of *required properties*. This allows to exactly access required values to compute a result. Additionally, metrics provide an *ID*, *description*, *type*, and an *URI to store results*. Every result is a float value, which can be interpreted as a *counter*, a *scale* value between 0 and 1, or 0 to 5 *stars*. The interpretation of the value depends on the predefined *type* of the metric.

Finally, a method *getScore()* is provided to compute result values. This computation requires input values, which are provided by a *DataContainer* object. The *DataContainer* object has to be filled with required values before computing metrics. This is described in the following section.

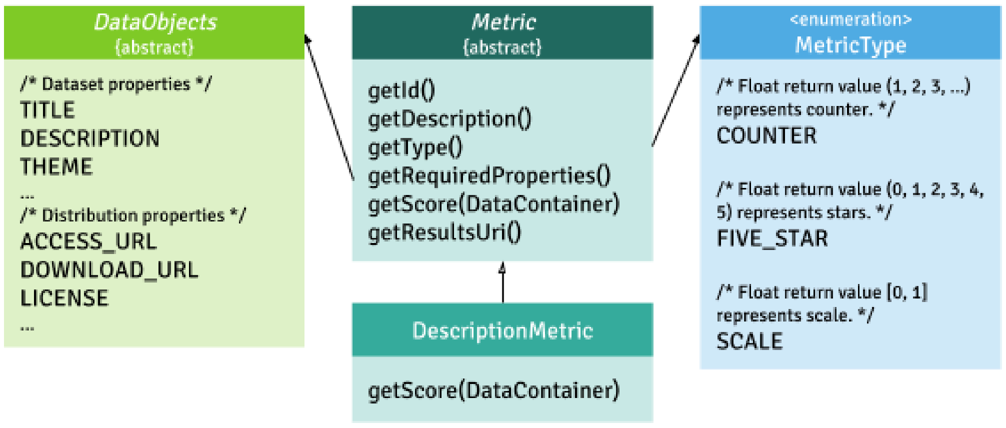

Figure: Required properties of metrics and metric types

2.3 Civet architecture

A top-level insight into the architecture including the data flow and the control flow is presented in the following figure. Note that the illustrated *DataContainer* occurrences (in light blue) represent the same object in different states.

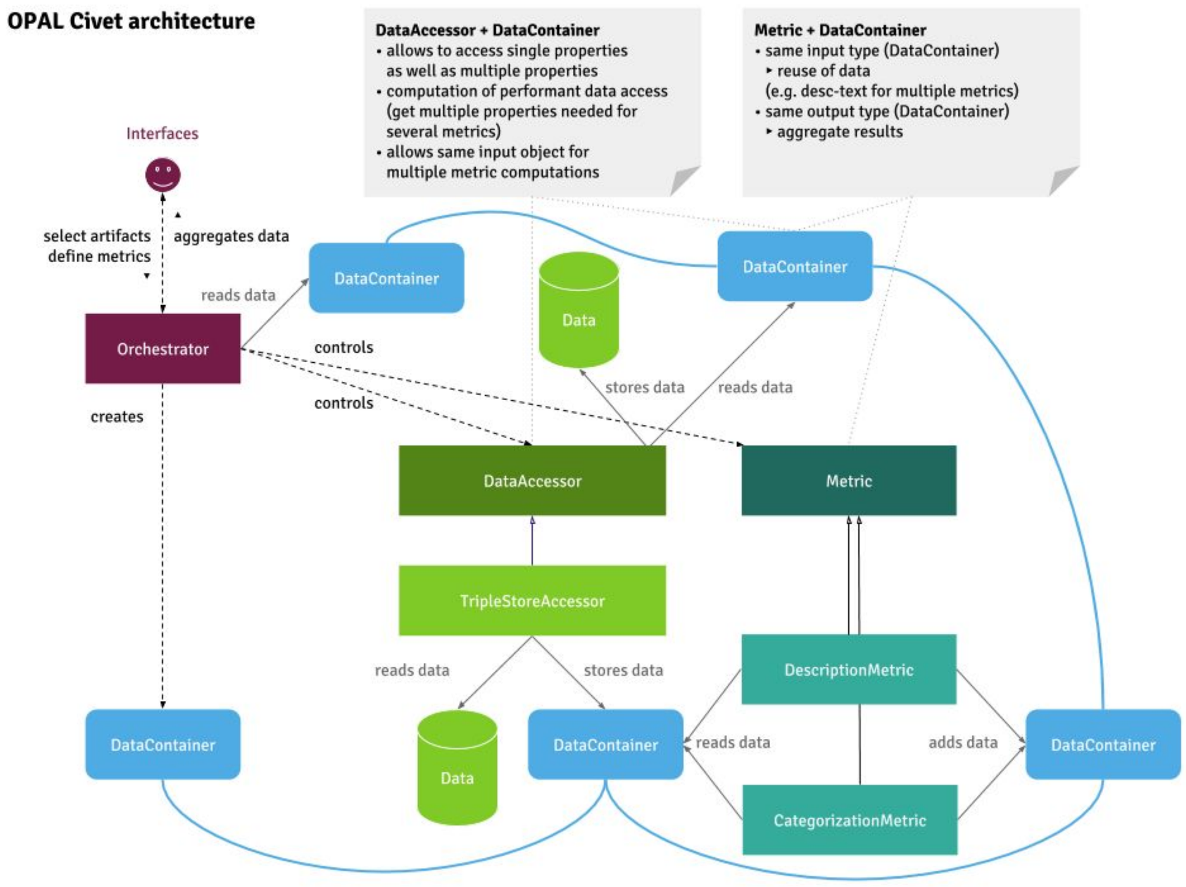

Figure: Top-level architecture of Civet components, data flow, and control flow

A typical execution of Civet can be described as follows:

1. An ***interface*** (e.g. the Java API or a command line interface) is used to select metrics and datasets.
2. The ***Orchestrator*** creates a new ***DataContainer***. Depending on the previously specified metrics, the required *DataObject*s are created. At this state it is defined, which values have to be loaded to compute the specified metric results.
3. The *Orchestrator* uses a ***DataAccessor*** to read the needed values. Values in **TripleStores** are typically related to dcat:Dataset and dcat:Distribution entries. The values are stored in the *DataContainer* for further processing.
4. The *Orchestrator* uses the previously specified metrics to create respective ***Metric*** objects. These objects are used to call ***getScore(DataContainer)*** and to compute metric result values.
5. The computed metric results are **stored** back in a database or **returned** via the initially used interface.

3 Usage of the component

In the current stage, Civet can be used for batch processing. This is designed to process large sets of datasets in named graphs. There are two options to access the component. These options are described afterwards.

3.1 Civet Java API

The Java API offers all required methods to run Civet. It can be used as follows:

- Download the Java code or clone the repository at [CivetGitHub]
- Execute the methods of org.dice_research.opal.civet.CivetApi

The following methods are provided:

● setSparqlQueryEndpoint	Sets the URL for SPARQL requests
● setSparqlUpdateEndpoint	Sets the URL for SPARQL modifications
● setNamedGraph	Sets the name of the database containing datasets
● computeAll	Computes all metrics for specified datasets
● compute	Computes specified metrics for specified datasets

3.2 Civet Command line interface (CLI)

Another option to run the Civet component is via command line interfaces. The current interface uses the Java API and provides the following arguments:

```
usage: java -jar civet.jar
Civet: OPAL quality framework
 -q,--query <URL>     SPARQL query endpoint, mandatory
 -u,--update <URL>    SPARQL update endpoint, mandatory
 -g,--graph <graph>   (Named graph, optional)
 -o,--offset <int>    Offset for results (not datasets), mandatory
 -e,--end <int>       Maximum number of results (not datasets), mandatory
 -l,--limit <int>     (Number of results per iteration, optional)
```

Listing: Command line interface options for batch processing

To build the required JAR file, the Maven Assembly Plugin is used. Therefore, a prepared configuration is provided via the Civet POM file, which is offered in the Civet repository [CivetGitHub].

4 Outlook: Future development

Depending on the final interaction of OPAL components, **additional interfaces** can be developed. Thereby, microservices would provide a solution for the requirements of message queues.

Further specifications of available **quality metrics** will also be implemented.

Especially for the upcoming topic and schema extraction (WP 3.4) as well as the data linking (WP 5.1) , **the quality of refined metadata records is expected to increase**.

5 References

[CivetGitHub] **OPAL Civet** https://github.com/projekt-opal/civet
[OPAL.D2.1] **OPAL Deliverable D2.1: Spezifikation der Crawler-Komponente**
[OPAL.D3.1] **OPAL Deliverable D3.1: Spezifikation von Qualitätskriterien**
[OPAL.D4.1] **OPAL Deliverable D4.1: Vokabularspezifikation**
[OPAL.D4.2] **OPAL Deliverable D4.2: Konvertierungskomponente**
[VocabDC] **Dublin Core** http://dublincore.org/documents/dcmi-terms/
[VocabDCAT] **Data Catalog Vocabulary (DCAT)** https://www.w3.org/TR/vocab-dcat/
[VocabFOAF] **Friend of a Friend (FOAF)** http://xmlns.com/foaf/spec/
[VocabDQV] **Data Quality Vocabulary (DQV)** https://www.w3.org/TR/vocab-dqv/
[VocabLDQD] **Quality assessment for Linked Data (LDQD)** https://www.w3.org/2016/05/ldqd
[VocabSKOS] **Simple Knowledge Organization System Namespace Document (SKOS)** https://www.w3.org/2009/08/skos-reference/skos.html