# DELIVERABLE D2.3

## Architecture and design documentation for COMSODE development tasks

<table>
<thead>
<tr>
<th>Project</th>
<th>Components Supporting the Open Data Exploitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acronym</td>
<td>COMSODE</td>
</tr>
<tr>
<td>Contract Number</td>
<td>FP7-ICT-611358</td>
</tr>
<tr>
<td>Start date of the project</td>
<td>1\textsuperscript{st} October 2013</td>
</tr>
<tr>
<td>Duration</td>
<td>24 months, until 31\textsuperscript{st} September 2015</td>
</tr>
</tbody>
</table>

| Date of preparation           | 30th May 2014                                   |
| Author(s)                     | Peter Hančák, Tomáš Knap, Sveto Krchnavý, Gabriel Lachmann, Martin Nečaský |
| Responsible for the deliverable | Peter Hanečák                                   |
| Email                         | peter.hanecak@eea.sk                            |
| Reviewed by                   | Flavio de Paoli, Miroslav Konečný               |
| Status of the Document        | Final                                            |
| Version                       | 1.0                                              |
| Dissemination level           | CO Confidential, only for members of the consortium (including the Commission Services) |
Table of Contents

1. Executive Summary .................................................................................................................. 4

2. Deliverable context .................................................................................................................. 5
   2.1. Purpose of deliverable ....................................................................................................... 5
   2.2. Related Documents .......................................................................................................... 5

3. Methodology used .................................................................................................................. 6
   3.1. Methodology .................................................................................................................. 6
   3.2. Partner contributions ....................................................................................................... 6

4. System overview ..................................................................................................................... 7
   4.1. Actors ............................................................................................................................. 7
   4.2. Inputs ............................................................................................................................. 8
   4.3. Outputs .......................................................................................................................... 8
   4.4. Features for data publishers ........................................................................................... 8
   4.5. Features for data consumers .......................................................................................... 8
       4.5.1. Citizen, data analyst, etc. ....................................................................................... 9
       4.5.2. Aggregator of Open Data (public body, NGO, SME, etc.) .................................... 9
       4.5.3. Application developer using Open Data (SME, NGO, etc., public body too) ........ 9
       4.5.4. Data administrator ................................................................................................. 9
   4.6. Use Cases ........................................................................................................................ 9
       4.6.1. Use Cases for Data Publisher ............................................................................... 10
       4.6.2. Use Cases for Data Consumers ............................................................................. 13
   4.7. Other inputs for Architecture decisions .......................................................................... 14

5. Open Data Node Modules ...................................................................................................... 15
   5.1. Module ODN/UnifiedViews ............................................................................................. 16
       5.1.1. UnifiedViews - state of the art ............................................................................... 16
       5.1.2. Structure of the ODN/UnifiedViews and its context ........................................... 19
       5.1.3. Interaction with other modules ............................................................................. 19
   5.2. Module ODN/Storage ..................................................................................................... 20
       5.2.1. Structure of the ODN/Storage and its context ...................................................... 21
       5.2.2. Interaction with other modules ............................................................................. 22
   5.3. Module ODN/Publication ............................................................................................... 23
       5.3.1. Structure of the ODN/Publication and its context ................................................. 23
       5.3.2. File dumps .............................................................................................................. 24
       5.3.3. SPARQL endpoint ................................................................................................. 25
       5.3.4. Rest API ................................................................................................................. 25
       5.3.5. Interaction with other modules ............................................................................. 26
   5.4. Module ODN/InternalCatalog ......................................................................................... 27
       5.4.1. Data Catalog ........................................................................................................... 27
       5.4.2. Data Catalog in ODN/InternalCatalog ................................................................. 29
       5.4.3. Interaction with other modules ............................................................................. 29
   5.5. Module ODN/Catalog ...................................................................................................... 29
       5.5.1. Interaction with other modules ............................................................................. 30
   5.6. Module ODN/Management ............................................................................................ 30
       5.6.1. Wizard for preparing the transformation task ......................................................... 31
       5.6.2. Structure of ODN/Management and its context .................................................. 31
       5.6.3. Interaction with other modules ............................................................................. 31
6. Development .................................................................................................................. 32
   6.1. Development process .............................................................................................. 32
   6.2. Used technologies ................................................................................................. 32
       6.2.1. Spring ............................................................................................................. 32
       6.2.2. Vaadin ........................................................................................................... 32
       6.2.3. Sesame openRDF ......................................................................................... 33
       6.2.4. EclipseLink .................................................................................................... 33
       6.2.5. XStream ......................................................................................................... 33
       6.2.6. KineticJS ........................................................................................................ 33
       6.2.7. OSGI framework .......................................................................................... 33
   6.3. Development tools ............................................................................................... 34
       6.3.1. Maven ............................................................................................................. 34
       6.3.2. Git + GitFlow ................................................................................................ 34
       6.3.3. IDE for developers ....................................................................................... 34
7. Deployment ................................................................................................................... 35
   7.1. Basic ODN Deployment Scenarios ........................................................................ 35
       7.1.1. Tight integration, at the publisher's premises .................................................. 36
       7.1.2. Loose integration ............................................................................................ 37
       7.1.3. No integration, deployed at 3rd parties ............................................................ 38
   7.2. Deployment of ODN Modules ............................................................................... 39
       7.2.1. Single Machine .............................................................................................. 39
       7.2.2. Distributed Environment ............................................................................. 40
8. Licensing ....................................................................................................................... 42
9. Abbreviations used ...................................................................................................... 43
10. Glossary ...................................................................................................................... 44
11. References .................................................................................................................. 46
12. List of figures and tables ............................................................................................ 47
1. Executive Summary

The purpose of this deliverable is to define architecture and design principles of the system to fulfill requirements defined in Deliverable 2.1 ‘User requirements for the publication platform from target organizations, including the map of typical environments’.

The document describes architecture of a software solution for open data publication (Open Data Node) that will be developed by COMSODE project and is also a basis for methodologies for open data publication (which will also be developed by the project too). Architecture was developed based on identified user requirements (Use cases).

The system has been split into modules that communicate using interfaces. Modules and interfaces are described in more details in particular sections. For each module is described what the role of the module is and how it fits into context of overall system, including also description of its interactions with other modules. Some modules are describes also with alternatives that can be used to fulfill the functionality of the module.

Due to modularity of the system, it is possible to use the system in different deployment environments and also to use only some modules of the system and integrate with other modules and systems. Some deployment schemes are also described in the document.

Modules shall be licensed as open source, so there could be contributors that could improve the modules or extend the system. Thus also description of development model and environment is included in the document.

This architecture is a base for next steps of the COMSODE project, mainly in WP4 - Development of software components and tools. However, it may happen that some new component(s) will be identified in development phase, fulfilling better the functionality of a particular module. In that case, we will update the module description in this document accordingly including explanation of the replacement. Subsequent updates to this document will be part of ODN documentation, distributed along with the software.
2. Deliverable context

2.1. Purpose of deliverable

One of main objectives of COMSODE is to create a publication platform called Open Data Node (ODN) that builds on results of previous research and development in the linked data field. Its mission is to bring results from research environment into real world for people, SMEs and other organizations to use and reuse.

ODN is supposed to become a foundation for a data integration platform based on Open Data which will allow the reuse of data not only between public bodies and end-users but also among public bodies themselves: public bodies can exchange information by using the same infrastructure and tools as end-users which will decrease costs of exchanging the data and in most cases also enhance the quality and speed of the exchange.

This deliverable is an output of Task 2.2 - Architecture and design of the publication platform (ODN). The purpose of the deliverable is to provide an overview of the architecture of Open Data Node, its modules, dependencies between modules and main communication interfaces.

2.2. Related Documents

- DOW, version date 2013-08-08, pages 7-8 and 42-43
- Deliverable 2.1: User requirements for the publication platform from target organizations, including the map of typical environments
3. Methodology used

3.1. Methodology

The architecture of the system and design were created in an iterative way – from outline in the DOW, through the meeting discussions within consortium and with User Board, to this deliverable. We took into account also previous development of components of the system, as some of them existed before the COMSODE project started or before work on this deliverable started.

The project uses internal collaborative space - called wiki - based on Atlassian Confluence technology with access of all members of the project team. There was established a dedicated space in the internal wiki for collection of inputs to the architecture and design from consortium members and from members of User Board. Also input (obtained by means other than wiki) from possible users of the platform was included in use cases.

ODN platform is planed to reuse, integrate and extend open source components, and architecture is based on the current status of those components and also on what additional features are possible to be included in those components.

It is expected, that the deliverable will be updated when significant design changes are required due to changes in existing components or when a new component that better fits the project needs becomes available and can replace the currently selected component.

3.2. Partner contributions

The architecture and design document was prepared under management of EEA, LTD. Tomas Knap and Peter Hanečáčk are the architects of ODN and UnifiedViews and DSE CUNI provided main input to Use cases.

The system design and architecture have been discussed internally on several meetings of the project team, including the weekly project calls and also with the User Board of the project – involving prospective users into the process.

UNIMIB, DSE CUNI and ADDSEN have reviewed the pre-final version of the document.
4. System overview

Open Data Node (ODN) is a publication platform which provides to governments, municipalities and other subjects (e.g. companies) an easy way of publishing their data as machine-readable (linked) open data.

ODN performs extraction and transformation (conversion, cleansing, anonymization, linking etc.) of data provided by governments and municipalities. ODN stores the result of data transformations in its own storage. ODN publishes the results of data transformations in open and machine-readable formats to data consumers (citizens, companies, governments).

4.1. Actors

Main actors dealing with the system

- **Data publishers** - government, municipalities, and other subjects providing data
- **Data consumers** - government, municipalities, non-profit organizations (NGOs), citizens (general public), companies (SMEs), application developers consuming the transformed data
- **Data administrators** - administrators, analysts, data curators, etc. (partially) responsible for configuring ODN - usually employees of data publisher
- **Administrators** - IT staff responsible for installation, maintenance and (partially) configuration of ODN - usually employees of data publisher

ODN helps publishers with the complexity of source data and their transformations to open data and deliver easy-to-use and high quality open data to the data consumers.

ODN helps data consumers get the data easily and efficiently in open, machine-readable formats.
4.2. Inputs

Input to the system is data of data publishers stored in heterogeneous environments, using wide variety of formats and employing a lot of different technologies to access and process that data.

4.3. Outputs

Outputs of the system are published open data in various forms, as linked data or as tabular data. Also API access to the data is included. Data consumer may be provided with:

- RDF data as a result of export from the storage
- RDF data as a result of SPARQL Query
- CSV data as a result of export from the storage
- REST API to access data in the storage

All forms of the data shall be available for data consumers under an open license.

4.4. Features for data publishers

- automated and repeatable data harvesting: extraction and transformation (conversion, cleansing, anonymization, etc.) of data, both:
  - initial harvesting of whole datasets (first import)
  - periodical harvesting of incremental updates
- integration tools for extracting data from publisher's internal systems (e.g. databases, data in files, information systems with API, etc.)
- internal storage for data and metadata; the metadata format will be based on DCAT (http://www.w3.org/TR/vocab-dcat/)
- data publishing in open and machine readable formats to the general public and businesses including automated efficient distribution of updated data and metadata (dataset replication)
- integration with data catalogs (like CKAN) for automated publication and updating of dataset metadata
- internal data catalog of datasets for maintenance of dataset metadata

4.5. Features for data consumers

Features for data consumers are discussed separately for different types of data consumers.
4.5.1. Citizen, data analyst, etc.

- user is typically accessing ODN instance maintained by someone else, user it not running his own instance
- user may download data dumps and call APIs to get the data which he is interested in
- data dumps changes are advertised as Atom feeds
- user may access the data indirectly, for example via 3rd party data catalog, which - in order to show the user preview or visualization of data - has to first download that data (in a similar manner as if user was accessing it: i.e. downloading a dump or accessing an API from ODN instance maintained by someone else)

4.5.2. Aggregator of Open Data (public body, NGO, SME, etc.)

- the same as in 4.5.1
- aggregator may easily replicate content in another Open Data Node
- aggregator may automate data integration and linking

4.5.3. Application developer using Open Data (SME, NGO, etc., public body too)

- the same as in 4.5.2
- application developer has tools for preparing automated generation of API and custom API of the published datasets

4.5.4. Data administrator

- possibility to set up the data extraction, transformation, and publication of open data
- possibility to monitor execution of data extraction and transformation tasks
- possibility to debug data extraction and transformation

4.6. Use Cases

In this deliverable, we depict use cases for data publishers and data consumers. Full list of use cases, also with related scenarios and mock-ups can be found at https://team.eea.sk/wiki/display/COMSODE/Use+Cases (note: This is the internal consortium wiki space mentioned in section “2.1. Methodology” - it is subject to change based on subsequent user requirements and will be most probably moved to public space once community around ODN grows).

For each use case below, we introduce its name, short description and modules in ODN participating on this use case (ODN/M = ODN/Management module, ODN/UV = ODN/UnifiedViews, ODN/P = ODN/Publication module, ODN/IC = ODN/InternalCatalog module, ODN/C = ODN/Catalog module). For full details, please see https://team.eea.sk/wiki/display/COMSODE/Use+Cases.
### 4.6.1. Use Cases for Data Publisher

**Figure 2:** Use cases for data publisher

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Short Description</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC</td>
<td>Create dataset record</td>
<td>As a data publisher I want to create new record about the intended published data, so that I can define for every dataset information about the source data, intended transformations and ways how the transformed data should be published</td>
<td>ODN/M ODN/IC</td>
</tr>
<tr>
<td>UC</td>
<td>Edit/Manage dataset record</td>
<td>As a data publisher I want to edit/manage dataset records</td>
<td>ODN/M ODN/IC</td>
</tr>
<tr>
<td>UC</td>
<td>Delete dataset record</td>
<td>As a data publisher I want to delete outdated/obsolete dataset record</td>
<td>ODN/M ODN/IC</td>
</tr>
<tr>
<td>UC</td>
<td>Configure transformation</td>
<td>As a data publisher I want to configure dataset transformation (cleansing, linking, enrichment,</td>
<td>ODN/UV</td>
</tr>
<tr>
<td>UC</td>
<td>Feature Description</td>
<td>User Story</td>
<td>Diagrams</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>4a</td>
<td>Configure transformation using wizard</td>
<td>As a data publisher I want to configure dataset transformation (cleansing, linking, enrichment, quality assessment, etc.) using a wizard, so that it is really simple to prepare typical dataset transformation</td>
<td>ODN/M</td>
</tr>
<tr>
<td>5</td>
<td>Configure publication</td>
<td>As a data publisher I want to configure how the transformed dataset is published, thus, how the dataset may be consumed by data consumers</td>
<td>ODN/P</td>
</tr>
<tr>
<td>6</td>
<td>Publish dataset</td>
<td>As a data publisher I want to publish the dataset</td>
<td>ODN/M, ODN/IC, ODN/UV, ODN/P, ODN/C</td>
</tr>
<tr>
<td>7</td>
<td>Transform dataset</td>
<td>As a data publisher I want to transform the dataset (the dataset is transformed but not published yet)</td>
<td>ODN/UV</td>
</tr>
<tr>
<td>8</td>
<td>Debug dataset transformation</td>
<td>As a data publisher I want to debug the dataset transformation, see intermediate results of the transformation, see debug message illustrating what happened during the dataset transformation</td>
<td>ODN/UV</td>
</tr>
<tr>
<td>9</td>
<td>Configure creation of RDF dumps</td>
<td>As a data publisher I want to configure creation of RDF dumps from my published datasets</td>
<td>ODN/P</td>
</tr>
<tr>
<td>10</td>
<td>Configure creation of CSV dumps</td>
<td>As a data publisher I want to configure creation of CSV dumps from my published datasets</td>
<td>ODN/P</td>
</tr>
<tr>
<td>11</td>
<td>Configure publishing data via REST API</td>
<td>As a data publisher I want to configure how the REST API is generated on top of my published data, which data is accessible via REST API, which users may use REST API, which methods of accessing my data is available to data consumers</td>
<td>ODN/P</td>
</tr>
<tr>
<td>12</td>
<td>Configure publishing to SPARQL Endpoint</td>
<td>As a data publisher I want to configure how data consumers may connect to SPARQL endpoints with my published data</td>
<td>ODN/P</td>
</tr>
<tr>
<td>13</td>
<td>Schedule dataset publication</td>
<td>As a data publisher I want to automate the publication process, so that it can run every week or every time when new version of dataset is</td>
<td>ODN/M, ODN/UV</td>
</tr>
<tr>
<td>UC 14</td>
<td>Schedule dataset transformation</td>
<td>As a data publisher I want to automate the transformation part of the publication process, so that it can run every week or every time when new version of dataset is available</td>
<td>ODN/M ODN/UV</td>
</tr>
<tr>
<td>UC 15</td>
<td>Monitor data publishing tasks</td>
<td>As a data publisher I want to monitor the publishing task to see how the data transformation and data publishing were executed for my datasets</td>
<td>ODN/M</td>
</tr>
<tr>
<td>UC 16</td>
<td>Basic overview about the transformation pipelines' execution</td>
<td>As a data publisher I want to monitor the publication of the dataset to see whether the publication was OK, or there were some errors</td>
<td>ODN/M ODN/UV ODN/P</td>
</tr>
<tr>
<td>UC 17</td>
<td>Detailed overview about the transformation pipelines' execution</td>
<td>As a data publisher I want to see the detailed overview about the transformations of the dataset</td>
<td>ODN/UV</td>
</tr>
<tr>
<td>UC 18</td>
<td>Browse transformation logs/events</td>
<td>As a data publisher I want to browse logs and events to see in detail what happened during the dataset transformation</td>
<td>ODN/UV</td>
</tr>
<tr>
<td>UC 19</td>
<td>Browse intermediate data</td>
<td>As a data publisher I want to browse the intermediate data produced as the dataset is being transformed</td>
<td>ODN/UV</td>
</tr>
<tr>
<td>UC 20</td>
<td>Overview about the publication of the transformed data</td>
<td>As a data publisher I want to be informed about the publication of the transformed dataset, whether there were some problems or not</td>
<td>ODN/P</td>
</tr>
<tr>
<td>UC 21</td>
<td>Schedule publishing of transformed RDF data</td>
<td>As a data publisher I want to automate the publishing of the transformed datasets; typical requested behaviour: whenever the dataset is transformed, it should be also published</td>
<td>ODN/P</td>
</tr>
<tr>
<td>UC 22</td>
<td>Publish transformed dataset</td>
<td>As a data publisher I want to publish the dataset, which has already been transformed</td>
<td>ODN/P</td>
</tr>
</tbody>
</table>

**Table 1: Use cases for data publisher**
4.6.2. Use Cases for Data Consumers

**Figure 3: Use cases for data consumers**

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Short Description</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC 101</td>
<td>Consume RDF data dump</td>
<td>As a data consumer I want to download RDF data dump, so that I can load it to my data store and work with it</td>
<td>ODN/P</td>
</tr>
<tr>
<td>UC 102</td>
<td>Consume CSV data dump</td>
<td>As a data consumer I want to download CSV data dump, so that I can load it to my data store and work with it</td>
<td>ODN/P</td>
</tr>
<tr>
<td>UC 103</td>
<td>Consume version of the data dump valid at certain time</td>
<td>As a data consumer I want to get the data dumps valid at certain time in the past</td>
<td>ODN/P</td>
</tr>
<tr>
<td>UC 104</td>
<td>Query SPARQL Endpoint</td>
<td>As an advanced data consumer I want to query RDF data directly using SPARQL endpoint</td>
<td>ODN/P</td>
</tr>
<tr>
<td>UC 105</td>
<td>Use REST API</td>
<td>As a data consumer I want to use REST API, so that I can work with the data from my app</td>
<td>ODN/P</td>
</tr>
<tr>
<td>UC 106</td>
<td>Browse Data Catalog</td>
<td>As a data consumer I want to browse and search the list of datasets (data catalog)</td>
<td>ODN/C</td>
</tr>
<tr>
<td>UC 107</td>
<td>Get metadata about dataset</td>
<td>As a data consumer I want to get metadata of the published dataset</td>
<td>ODN/C</td>
</tr>
<tr>
<td>UC 108</td>
<td>Browse (sample) data</td>
<td>As a data consumer I want to browse data sample to get idea what is in the dataset</td>
<td>ODN/P</td>
</tr>
<tr>
<td>UC 109</td>
<td>Data dump changes</td>
<td>As a data consumer I want to be notified (for example via RSS or Atom) when a data dump is updated or changed</td>
<td>ODN/P</td>
</tr>
</tbody>
</table>

Table 2: Use cases for data consumers

4.7. Other inputs for Architecture decisions

This is additional list of inputs extending Deliverable 2.1: ‘User requirements for the publication platform from target organizations, including the map of typical environments’.

- The system must be extensible with new DPU on the data transformation (ETL) pipelines.
- The overhead of managing data transformation (ETL) tasks and managing outputs from/preparing inputs to DPUs must be reasonable - any execution of ETL task must not take more than 200% of the time needed for manual execution of the DPUs' logic without the use of the ETL framework (under the condition than the pipeline is running alone)
- The system must be able to process big files - CSV files containing millions of rows, RDF files containing hundreds of millions of triples.
- Response time: The Web management GUI of ODN must respond in 99.9% cases in less than 1s (all components of ODN on one machine, client is connecting to server using 100Mb line at least).
- Target platform: Linux, Windows
- Preferred languages for the system implementation: Java, others only in case of reuse of existing components with sufficient added-value (for ODN and for ODN users)
- The internal format for all data being transformed during the ETL process is RDF, an universal machine readable data format.
5. Open Data Node Modules

Open Data Node consists of the following modules:

- ODN/UnifiedViews
- ODN/Storage
- ODN/Publication
- ODN/InternalCatalog
- ODN/Catalog
- ODN/Management

Modules listed above are discussed in more detail in the following sections.

SearchPortal, which allows users to search the published data, is not described in this document, as it is separate application, not part of ODN.
5.1. Module ODN/UnifiedViews

Module ODN/UnifiedViews is an ETL & data enrichment tool. It is responsible for extracting and transforming source data (datasets), so that they can be published as (linked) open data. The result of the transformation is stored in the database managed by ODN/Storage module.

ODN/UnifiedViews module is responsible for:

1. extracting data provided by data publishers
2. transforming these data to machine readable data format; such transformation may include enriching the data, cleansing the data, assessing the quality of the data
3. storing the machine readable data to the database managed by ODN/Storage.

Input of the module is the data provided by data publishers. Data is expected to be structured, mostly tabular or linked data (RDF). Module will support basic data formats out of the box, support for more complex data formats is available via plugins.

Module will work with different formats (in files), but preferred is data in RDF format. RDF format will allow usage of advanced data cleansing and enrichment techniques based on linked data also for use cases where output will not be in RDF (i.e. for example cases where ODN will be used to clean CSV files before publishing).

Output of the module is the extracted and transformed machine readable data stored in ODN/Storage. Again, data is expected to be structured, tabular or linked data.

5.1.1. UnifiedViews - state of the art

Module ODN/UnifiedViews will use as its base the tool UnifiedViews (https://github.com/UnifiedViews). It is an ETL framework with a native support for transforming RDF data. UnifiedViews allows users to define, execute, monitor, debug, schedule, and share data transformation tasks.

UnifiedViews was originally developed as a student project at Charles University in Prague and now it is maintained by Semantica.cz, Czech Republic, Semantic Web Company, Austria, and EEA, Slovak Republic.

UnifiedViews allows users to define and adjust data processing tasks (pipelines) using a graphical user interface (see Figure below); the core components of every data processing task are data processing units (DPUs). DPUs may be drag dropped on the canvas where the data processing task is constructed. Data flow between two DPUs is denoted as an edge on the canvas; a label on the edge clarifies which outputs of a DPU are mapped to which inputs of another DPU. UnifiedViews natively supports exchange of RDF data between DPUs; apart from that, files may be exchanged between DPUs.
UnifiedViews takes care of task scheduling. Users can plan executions of data processing tasks (e.g., tasks are executed at a certain time of the day) or they can start data processing tasks manually. UnifiedViews scheduler ensures that DPUs are executed in the proper order, so that all DPUs have proper required inputs when being launched.

A user may configure UnifiedViews to get notifications about errors in the tasks’ executions; user may also get daily summaries about the tasks executed.

To simplify the process of defining data processing tasks and to help users analyzing errors during data processing task executions, UnifiedViews provides users with the debugging capabilities. Users may browse and query (using SPARQL query language) the RDF inputs to and RDF outputs from any DPU.

UnifiedViews framework also allows users to create custom plugins - data processing units (DPUs). Users can also share DPUs with others together with their configurations or use DPUs provided by others.

Technical structuring and licensing of UnifiedViews allows DPUs to be licensed not just as Open Source, but also using proprietary license. This is a planned feature of the tool needed by use cases where commercial exploitation is needed. ODN will support same commercial use cases.

5.1.1.1. UnifiedViews components and dependencies

Figure below depicts current maven modules in UnifiedViews and its dependencies. Modules in the yellow box are visible to DPU developers. The most important modules are:

- frontend - Management GUI of UnifiedViews
● backend - Engine running the data transformation tasks
● commons-app - DAO & Services module, which is common to frontend and backend modules; it is used to store configuration for pipelines, DPUs, pipeline executions etc.
● dataunit-rdf, dataunit-file - Modules with interfaces for data units; DPU developers writing new DPUs use these modules to read data from input data units and write data to output data units

Figure 6: UnifiedViews components and dependencies
5.1.2. Structure of the ODN/UnifiedViews and its context:

Figure 7: Context of ODN and UnifiedViews

ODN/UnifiedViews comprises of the important components as follows:

- **DAO & Service** - used to access database where configuration of ETL tasks and its executions is stored (realized by module *commons-app* in Figure 6)
- **HTTP REST Transformation API** - Services from DAO & Services layer exposed as HTTP REST methods. Used by ODN/Management module (this component is not realized by any module in Figure XX)
- **Data Processing Engine** - Robust engine running the manually launched or scheduled transformation tasks - transformations may include data cleansing, linking, integration, quality assessment (realized by “backend” module in Figure XX)
- **Management GUI** - GUI used to manage the configuration of pipelines, debugging executions, etc. (realized by “frontend” module in Figure XX)

5.1.3. Interaction with other modules

1. ODN/UnifiedViews loads the transformed data to ODN/Storage. A special DPUs - *RDF data mart loader* and *Tabular data mart loader* must be provided to load transformed data to ODN/Storage to the corresponding data store. The data must be stored there together with metadata, so that ODN/Publication module knows which resources (tables, graphs)
are associated with which pipeline/dataset.

2. ODN/UnifiedView will provide RESTful management API, which will be used by ODN/Management to:
   - create new data transformation task (pipeline)
   - configure existing pipeline and get configuration of the pipeline
   - delete the pipeline
   - execute the pipeline
   - schedule the pipeline

An excerpt of the methods, which will be available to ODN/Management in a RESTful format is depicted below:

![Figure 8: Methods available to ODN/Management in RESTful format](image)

3. Management GUI of ODN/UnifiedViews is used by ODN/Management to:
   - show the pipeline detail in an expert mode (user may drag&drop DPUs, fine-tune pipeline configuration)
   - show the detailed results of pipeline executions (browse events/logs)
   - debug data being passed between DPUs
   - have an access to advanced scheduling options

### 5.2. Module ODN/Storage

The purpose of this module is to store the transformed data produced by ODN/UnifiedViews. ODN/Publication module uses ODN/Storage to get the transformed data, so that it can be published - provided to data consumers.
5.2.1. Structure of the ODN/Storage and its context

Two important components of ODN/Storage are:

- RDBMS data mart
- RDF data mart

5.2.1.1. RDBMS data mart

RDBMS data mart is a tabular data store, where data is stored when data publisher wants to prepare CSV dumps of the published dataset or provide REST API for data consumers.

ODN/Storage will use SQL relational database (such as MySQL, PostgreSQL, etc.) for storing tabular data.

Every transformation pipeline can contain one or more *Tabular data mart loaders* - DPUs, which load data resulting from the transformation pipeline to RDBMS data mart. Every loader loads data into a single table. The name for the table is prepared by ODN/UnifiedViews and is based on the dataset ID and ID of the tabular data mart loader DPU.

Since every published dataset may require more then one transformation pipeline, and not all results of every transformation pipeline should be published by ODN/Publication module, data publisher may decide which tables should be published by (1) manually specifying all the tables which should be published or by (2) specifying that all results of certain transformation pipeline should be published.

To support the above feature, data being stored to RDBMS data mart must be associated with metadata holding for every table at least:

- to which dataset the table belongs to
- which transformation pipeline produced the table

**Note:** Currently, UnifiedViews supports Openlink Virtuoso ([http://virtuoso.openlinksw.com/](http://virtuoso.openlinksw.com/)) as the only RDBMS implementation. As part of ODN, we will employ JDBC to add support for wider range of databases. Testing and validation will be done based on feedback from users (currently we plan to work also with PostgreSQL).
5.2.1.2. RDF data mart

Data is stored in RDF data mart when data publisher wants to prepare for data consumers RDF dumps of the published dataset or provide SPARQL endpoint on top of the published dataset.

Every transformation pipeline can contain one or more RDF data mart loaders - DPUs, which load data resulting from the transformation pipeline to RDF data mart. Every RDF data mart loader loads data to a single RDF graph. RDF graph represents a context for RDF triples, graph is a collection of RDF triples produced by one RDF data mart loader. The name for the RDF graph is prepared by ODN/UnifiedViews and is based on the dataset ID and ID of the RDF data mart loader DPU.

Since every published dataset may require more then one transformation pipeline, and not all results of every transformation pipeline should be published by ODN/Publication module, data publisher may decide which RDF graphs should be published by (1) manually specifying all the graphs which should be published or by (2) specifying that results of certain transformation pipeline should be published.

To support the above feature, data being stored to RDF data mart must be associated with metadata holding for every RDF data graph at least:

- to which dataset the graph belongs to
- which transformation pipeline produced the graph

Note: Currently, UnifiedViews supports Openlink Virtuoso (http://virtuoso.openlinksw.com/) and Sesame (http://www.openrdf.org/) as RDF data mart implementation. As part of ODN, we will employ SAIL API to add support for wider range of triplestores. Testing and validation will be done based on feedback from users.

5.2.2. Interaction with other modules

1. Every transformation pipeline (ODN/UnifiedViews) can contain one or more RDF/RDBMS data mart loaders - DPUs, which load data resulting from the transformation pipeline to the corresponding data mart (RDF/RDBMS).

2. ODN/Storage notifies ODN/Publication about changes which happened (dataset updates, etc.) so that ODN/Publication can adapt to the changes.

3. ODN/Publication uses data marts to get required graphs/tables to be published (exported as RDF/CSV dumps, made available via REST API/SPARQL Endpoint). ODN/Publication selects the relevant graphs/tables based on the data publishers preference and metadata associated with tables/graphs.

3. ODN/Management may query ODN/Storage to get statistics about stored data, at least:

- How many RDF graphs/tables is stored in RDF/RDBMS data mart in total/for the given dataset ID?
- How many RDF triples are stored in certain RDF graph in RDF data mart?
- How many records are in certain table in RDBMS data mart?
5.3. Module ODN/Publication

Module responsible for publishing data via REST APIs, SPARQL endpoint or as data dumps in RDF or CSV formats. Published data is already transformed as defined by data transformation pipelines in ODN/UnifiedViews and stored in ODN/Storage.

The module allows data administrators/publishers to select how the published datasets are provided to data consumers; in particular, ODN/Publication module allows users to select:

- publication of the dumps (CSV for tabular data, RDF for linked data),
- publication via API (SPARQL Endpoint for RDF data, REST API for tabular data).

Data administrators/publishers may also configure some specific settings per each publication option: to tweak dump generation process (like which RDF serialization to use: Turtle, XML, etc.), to select which resources (tables, graphs) associated with the transformed dataset (and stored in ODN/Storage) should be published - made available to data consumers, etc.

5.3.1. Structure of the ODN/Publication and its context

![Figure 10: ODN/Publication module](image-url)
ODN/Publication comprises of the important components as follows:

- **DAO & Service layer** - used to access database where configuration and results of publication tasks are stored
- **Publication Management API** which is called by ODN/Management when certain dataset should be published or when certain methods of data consumption (REST API, SPARQL Endpoint, dumps) should be enabled or disable
- **Publication Engine** - module, which is responsible for:
  - creating dumps for the given dataset
  - configuring SPARQL endpoint/REST API for the given dataset
- **Management GUI** - GUI used to manage the configuration of the ODN/Publication module

Note: As part of data publication, some metadata will be published by this module too (for example “Last Modification Time” will be included in appropriate HTTP headed in response). But publication of metadata is mainly responsibility of ODN/Catalog (see section 4.5).

### 5.3.2. File dumps

ODN/Publication module supports creation of file dumps in CSV or RDF formats. When dataset is transformed, it is being published. As part of the publishing of the transformed dataset, CSV or RDF dump may be created. The dump in the CSV/RDF format is created if the data publisher decides so.

To create the dump, ODN/Publication module exports the desired data in ODN/Storage. Afterwards, the dump is versioned using Git ([http://git-scm.com/](http://git-scm.com/)). Git allows data consumers to work with the latest or any other previous version of the dataset. ODN/Publication also publishes metadata of the dump, which are obtained from ODN/InternalCatalog.

Finally, new entry in the Atom feed ([http://en.wikipedia.org/wiki/Atom_(standard)](http://en.wikipedia.org/wiki/Atom_(standard))) associated with the processed dataset is created; such feed points data consumers to the file(s) in the git repository, where the published data and metadata is. Such feed must be reachable from the dataset record in the ODN/Catalog module.

#### 5.3.2.1. RDF dumps

RDF dump may be published only if the result of the dataset transformation is available in RDF data mart in ODN/Storage.

To create the dump, ODN/Publication queries the RDF data mart via SPARQL construct query to get dump in N-Triples ([http://www.w3.org/TR/2014/REC-n-triples-20140225/](http://www.w3.org/TR/2014/REC-n-triples-20140225/)) RDF serialization format. We use N-Triples as RDF serialization format, because it is line oriented serialization format which may be easily versioned by Git.

#### 5.3.2.2. CSV dumps

CSV dump may be published only if the result of the dataset transformation is available in
RDBMS data mart in ODN/Storage.
To create the dump, ODN/Publication module exports the desired table in RDBMS data mart as CSV dump.

5.3.3. SPARQL endpoint
ODN/Publication module supports publication of data via SPARQL endpoints. When dataset is transformed, it is being published. As part of the publishing of the transformed dataset, data may be made available via SPARQL endpoint. Data is made available via SPARQL endpoint only if the data publisher decides so. Data may be made available via SPARQL endpoint only if the result of the dataset transformation is available in RDF data mart.

To make the data available via SPARQL endpoint, ODN/Publication module provides data consumers with a simple querying interface, where data consumer may query the published data and associated metadata (obtained from ODN/InternalCatalog) using SPARQL query. There is no versioning in this case, only latest data is available via SPARQL endpoint.

5.3.4. Rest API
ODN/Publication module supports creation of REST APIs for data consumption. When dataset is transformed, it is being published. As part of the publishing of the transformed dataset, REST API may be generated for the published data. REST API is generated if the data publisher decides so. REST API may be generated only if the result of the dataset transformation is available in RDBMS data mart.

API is based on “Representational state transfer” software architectural style (https://en.wikipedia.org/wiki/Representational_State_Transfer) and - for the purpose of Open Data - will provide read-only functionality: Users will be able to get the data from datasets using HTTP protocol, getting results in JSON, XML, CSV or RDF formats based on their preference.

API is supposed to be used by programmers or similarly skilled users who can develop software or scripts. But given the truly simplistic nature of this kind of API, even causal user can work with it using common web browser.

There is no versioning in this case; only latest data is available via REST API.

4.3.6. Dataset replication
Automated efficient distribution of updated data and metadata will be achieved by careful implementation of two main methods mentioned earlier, e.g. file dumps and REST API, complemented with third option based on Git.

First two options are generic and interoperable: they will work regardless of the exact tool being used to replicate the data. At one end there will be ODN, on the other hand it can be anything.
The rest is sort of proprietary: technically based on open formats and protocols but limited to smaller/niche audiences.

Note: There is possibility also for fourth option based on combination of file dumps and peer2peer technologies (like BitTorrent). As of now we do not register a demand for that so it is not in the scope of the development.

4.3.6.1. Via file dumps

Proper publishing of file dumps, along with increments and Atom feeds, combined with proper usage of features of HTTP protocol (cache related headers, range requests, if-modified-since headers etc.) is one option.

4.3.6.2. Via REST API

REST API is another option, but that requires presence of “last modified” (or similar) fields within datasets at the line/record level.

Those two options are generic and interoperable: they will work regardless of the exact tool being used to replicate the data. At one end there will be Open Data Node, on the other hand ca be anything.

4.3.6.3. Via Git

Third option is to take advantage of Git versioning (see section “x.x.x. file dumps”):

- ‘git clone’ can be used to get a first copy of data
- ‘git pull’ can be used repeatedly to obtain subsequent updates

This method takes advantage of a lot of existing software and infrastructure, mainly Git versioning tool and for example GitHub (or GitHub like) repositories and is most suitable to software developers and subset of data analysts who already use such tools.

5.3.5. Interaction with other modules

1. ODN/Management initiates any publication process via Publication API of ODN/Publication. ODN/Publication module uses ODN/Storage to get the data which should be published.

2. ODN/Management uses Management GUI of ODN/Publication to set up the settings for creation of CSV/RDF dumps, settings for generating REST APIs, settings for preparing SPARQL endpoint.

3. ODN/Publication react to notifications from ODN/Storage by for example recreating file dumps or invalidating cached information for updated datasets.

4. Data consumers may (1) download CSV/RDF dumps, (2) use SPARQL endpoints, (3) use REST APIs.
5.4. Module ODN/InternalCatalog

Before introducing ODN/InternalCatalog module, the general concept of data catalog is introduced.

5.4.1. Data Catalog

Data catalog holds metadata about each published dataset. Data catalog allows its users to browse/search the list of datasets and to see the metadata for every published dataset. Screenshot of a sample data catalog provided by data.gov.uk is shown below.

![Data.gov.uk catalogue view](image)

**Figure 11**: Data.Gov.uk catalogue view

There are already available solutions that implement data catalog functionality, such as CKAN and DKAN.

5.4.1.1. Comparison of CKAN/DKAN

CKAN ([http://ckan.org/features/](http://ckan.org/features/)) is a powerful data management system that makes data accessible – by providing tools to streamline publishing, sharing, finding and using data. CKAN is aimed at data publishers (national and regional governments, companies and organizations) wanting to make their data open and available. Note: We may also consider Etalab ([https://github.com/etalab](https://github.com/etalab)), a fork of CKAN.

DKAN ([http://nucivic.com/dkan/](http://nucivic.com/dkan/), [https://drupal.org/project/dkan](https://drupal.org/project/dkan)) is an open source data platform with a full suite of cataloging, publishing and visualization features that allows governments, non-profits and universities to easily publish data to the public.
<table>
<thead>
<tr>
<th>Aspects &amp; Features</th>
<th>CKAN</th>
<th>DKAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>open source &amp; extend-able</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>primary language</td>
<td>Python</td>
<td>PHP</td>
</tr>
<tr>
<td>database supported</td>
<td>PostgreSQL</td>
<td>MySQL, PostgreSQL, SQL Server, or Oracle</td>
</tr>
<tr>
<td>data import via API</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>publish data and metadata</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>support for DCAT/DCAT-AP</td>
<td>Not complete</td>
<td>Yes</td>
</tr>
<tr>
<td>customized metadata fields</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>versioning dataset records</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>possibility to visualize data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>themable</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>statistics and usage metrics for datasets</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>extensions</td>
<td>CKAN extensions</td>
<td>Drupal modules</td>
</tr>
</tbody>
</table>

**Table 3: Comparison of CKAN and DKAN features**

When comparing CKAN and DKAN, the main difference is that DKAN is implemented on top of Drupal and CKAN on top of Pylons. Furthermore, DKAN supports DCAT-compliant format for expressing datasets’ metadata. DCAT (http://www.w3.org/TR/vocab-dcat/) is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web.

Although there is an extension to CKAN, which allows CKAN to expose and consume metadata from other catalogs using documents serialized using DCAT format.
(https://github.com/ckan/ckanext-dcat), we decided to use DKAN for the functionality of dataset catalog.

Sources used for the comparison:

- http://ckan.org/features/
- CKAN: extensions https://github.com/ckan/ckan/wiki/List-of-extensions

5.4.2. Data Catalog in ODN/InternalCatalog

Module ODN/InternalCatalog is the first module that encapsulates the functionality of data catalog. The data catalog provided by ODN/InternalCatalog module is used to manage datasets that should be transformed/published by ODN; it also allows data publishers to see details about the transformation/publishing process. It is an internal catalog, thus, it is not visible to public, but only data publisher/data administrator can use the catalog.

ODN/InternalCatalog module is internally using DKAN. Nevertheless, DKAN must be extended, so that it provides more data about the datasets being transformed and published, in particular ODN/InternalCatalog must have possibility to:

- depict the data processing pipeline, which is associated with the transformed & published dataset
- run data transformation/publishing from the catalog UI
- provide brief information about the status of the dataset transformation
- provide link to ODN/Publication module’s configuration dialog which configures how the dataset in the catalog is published

5.4.3. Interaction with other modules

ODN/InternalCatalog is used by ODN/Management to hold and present metadata about the datasets being transformed/published by the data publisher. On request, ODN/InternalCatalog publishes the internal data catalog records about the datasets already published to ODN/Catalog module.

5.5. Module ODN/Catalog

ODN/Catalog is the second module which encapsulates the functionality of data catalog. ODN/Catalog holds metadata about each dataset, which is published by ODN. This data
catalog is publicly visible, the primary users of this catalog are data consumers, who may browse/search the published datasets’ metadata; data consumer may also get a link to the dataset’s dump or API, so that they can consume the data in the dataset.

Every time a dataset is published by ODN/Publication module, it may be also published to the data catalog (module ODN/Catalog). Data is being exported to ODN/Catalog from ODN/InternalCatalog either automatically as new data is published by ODN/Publication module or manually on the request of data publisher/data administrator. The catalog in ODN/Catalog module must contain reference to the Atom feeds, so that dumps of the datasets and the associated metadata may be downloaded; the catalog has to also provide a link to REST API and SPARQL endpoint associated with the dataset.

Module ODN/Catalog is internally using the same tool as ODN/InternalCatalog to ensure the core data catalog functionality, i.e. DKAN.

5.5.1. Interaction with other modules

This module is used by ODN/Management to create new record or adjust the existing record in ODN/Catalog when the dataset is transformed by ODN/UnifiedViews and published by ODN/Publication module. The record in ODN/Catalog is built based on the metadata in ODN/InternalCatalog and based on the information about the location of REST APIs, Atom feeds referring data dumps, etc., provided by ODN/Publication module.

5.6. Module ODN/Management

Module responsible for managing the process of dataset transformation and publication. The diagram below shows the interaction of ODN modules when a dataset is published. The diagram below is showing the case when the dataset publication is launched manually; however, it may be also scheduled by ODN, so that it runs at certain times (e.g., every month).

Figure 12: ODN/Management module
5.6.1. Wizard for preparing the transformation task

ODN/UnifiedViews provides standard dialog for editing the data transformation pipeline. Further, ODN/Management provides a wizard (for inexperienced users) to prepare the transformation task. Wizard should be implemented by ODN/Management, using ODN/UnifiedViews HTTP REST Transformation API for interacting with transformation pipelines.

5.6.2. Structure of ODN/Management and its context

![Diagram](image)

**Figure 13:** Context of ODN/Management module

5.6.3. Interaction with other modules

ODN/Management allows management of the whole data transformation and publication process. ODN/Management uses ODN/InternalCatalog to store metadata about datasets to be transformed. ODN/Management calls ODN/UnifiedViews (its HTTP Transformation API) to create, execute transformation pipelines or get status of the transformation execution. ODN/Management can instruct ODN/Publication to publish data transformed by ODN/UnifiedViews and stored to ODN/Storage (based on request from ODN administrator/data publisher); this publication may also involve publication to ODN/Catalog.
6. Development

The main target development platform is Java. ODN/Publication may use PHP tool for generating APIs. Similarly, ODN/InternalCatalog and ODN/Catalog use already existing PHP tool DKAN to ensure data catalog functionality.

ODN must be able to run on both Linux and Windows operating systems.

6.1. Development process

We will create a github project(s) to host ODN source codes. Certain modules, such as ODN/UnifiedViews, are developed under a separated github project (https://github.com/UnifiedViews). For the whole ODN, we will follow the guidelines for sustainable development proposed for developing ODN/UnifiedViews (https://grips.semantic-web.at/display/UDDOC/Guidelines+for+Contributors).

In case of extensions of already existing tools, such as DKAN, we will fork their project (repository) and implement the extensions needed. Consequently, we will issue pull requests to the original project, so that the original forked repository may accept our changes.

6.2. Used technologies

6.2.1. Spring

Spring is an open source application framework and inversion of control container for Java platform. The core features of the Spring Framework can be used by any Java application, but there are extensions for building web applications on top of the Java EE (enterprise) platform. Spring will be used in all ODN modules implemented in Java.

6.2.2. Vaadin

It is a Java open source framework for building web applications. This framework incorporates event-driven programming and widgets, which enables a programming model that is closer to GUI software development than traditional web development with HTML and JavaScript. Developers may write pure Java code and the framework ensures the communication between server side code (application server) and client side code (browser), also Java code is automatically translated to JavaScript for the client code. Vaadin will be used in ODN/UnifiedViews, ODN/Management, ODN/Publication for implementing Management GUI. ODN/InternalCatalog, ODN/Catalog will be based on existing tool - DKAN - and its interface.
6.2.3. Sesame openRDF

Sesame openRDF is an open source framework for processing RDF data. This includes parsing, storing, inferencing and querying of/over such data. It offers an easy-to-use API that can be connected to all leading RDF storage solutions. It allows to connect to SPARQL endpoints and create applications that leverage the power of linked data and Semantic Web. OpenRDF is used in ODN/UnifiedViews, ODN/Storage, ODN/Publication to work with RDF database.

6.2.4. EclipseLink

EclipseLink is the open source Eclipse Persistence Services Project. The software provides an extensible framework that allows Java developers to interact with various data services, including databases, web services, Object XML mapping (OXM), and Enterprise Information Systems (EIS). EclipseLink is used to persist data objects in ODN/UnifiedViews.

6.2.5. XStream

XStream is a simple library to serialize objects to XML and back. XStream uses reflection to discover the structure of the object graph to serialize at run time and does not require modifications to objects. It can serialize internal fields, including private and final, and supports non-public and inner classes. XStream is used in ODN/UnifiedViews to store pipeline configurations.

6.2.6. KineticJS

KineticJS is an HTML5 Canvas JavaScript framework that enables high performance animations, transitions, node nesting, layering, filtering, caching, event handling for desktop and mobile applications.

You can draw things onto the stage, add event listeners to them, move them, scale them, and rotate them independently from other shapes to support high performance animations, even if your application uses thousands of shapes.

6.2.7. OSGI framework

ODN/UnifiedViews must support easy and smooth extension with custom DPUs added into the running application. Every DPUs may use its set of libraries. These libraries must not be in conflict. To ensure that, we use OSGI framework, in particular the Apache Felix implementation (http://felix.apache.org/).
6.3. Development tools

6.3.1. Maven

Maven (http://maven.apache.org/) is used for management of dependencies in the source code, portability between Java IDEs and easy application build.

6.3.2. Git + GitFlow

Git (http://git-scm.com/) is a version control system used to keep versions of source code and track changes.

GitFlow (https://github.com/nvie/gitflow) is a collection of Git extensions to provide high-level repository operations for Vincent Driessen's branching model (http://nvie.com/git-model).

SourceTree (http://www.sourcetreeapp.com/) is recommended GUI client (available for MS Windows and MAC OSX). For linux, command line is necessary to use to follow gitflow workflow.

6.3.3. IDE for developers

Project is not strictly bounded to specific IDE, but Eclipse (http://www.eclipse.org/), Netbeans (https://netbeans.org/) or InteliJ Idea (http://www.jetbrains.com/idea/) are recommended IDEs for developing Java applications.

Eclipse and Netbeans have also plugins for development of other type of applications (for example php/Drupal based).
7. Deployment

Firstly, we describe basic deployment scenarios of Open Data Node, considering Open Data Node as a single unit. Afterwards, we discuss deployment of ODN's modules.

7.1. Basic ODN Deployment Scenarios

Open Data Node can be deployed many times by many actors. ODN can help with needs specific to each particular actor:

- government organizations, municipalities, etc., want to publish majority of their information as Open Data
- other government bodies need to work with some data published by other government bodies
- non-profits and application developers want to run specific tasks using copies of official data, for example analytic and visualization applications, data integration, etc.

This section contains schemes for basic deployment options for ODN. Options are sorted based on achievable publishing quality from best (and most expensive) at the top to worst (and cheapest) at the bottom.
7.1.1. Tight integration, at the publisher's premises

Open Data Node is tightly integrated with publisher's internal application(s) - it has direct access to backend databases or is integrated into application(s) workflows via API. ODN is deployed alongside publisher's internal application(s), it has to respect network, security and other zones.

Typical scenario:

- publisher wants to achieve high quality and efficiency and is willing to invest more

Prerequisites:

- publisher is willing to update existing workflows and applications

Figure 14: Tight integration scenario
7.1.2. Loose integration

Open Data Node is integrated with publisher's application(s) in a loose way using some periodical data dumps or APIs. ODN can be deployed in several locations:

- at publisher's premises - access to the data is secured in a similar way as in the case of tight integration in Section 6.1.1.
- at collocated housing, for example data center shared by multiple government organizations - access to the source data secured for example using combination of IPsec, HTTPS and access controls (authentication and authorization)
- in the cloud - access to the source data secured with just HTTPS and access control (not suitable for sensitive data or sensitive internal systems)

Typical scenario:

- publisher wants to achieve high quality and efficiency but has limited resources so changes to existing infrastructure and applications have to be limited
- aggregator is affiliated with one or more publishers and is willing and able to invest into tighter integration with them (ministry aggregating data from municipalities or SME planning to make business using aggregated data)

Prerequisites:

- publisher is able to do minor modifications to existing workflows and applications
7.1.3. No integration, deployed at 3rd parties

Open Data Node is not explicitly integrated with publisher's applications, other existing means are used to get access to the data (either Open Data or other format or API, at worst case scrapping of data from website). ODN is deployed at 3rd party using their own hardware, collocated housing, or Cloud.

**Typical scenario:**
- publisher wants to publish Open Data but has severely limited resources or options so changes to existing infrastructure and applications have to be preferably none
- 3rd party aggregator or application developer wants to use data from one or more publishers but for some reason is not able or willing to implement tighter integration with them

**Prerequisites:**
- some usable form of access to source data is possible without changing existing workflows and applications
7.2. Deployment of ODN Modules

7.2.1. Single Machine

ODN modules may be deployed all on one machine as depicted below. The deployment requires:

- Application Server, which supports Java 7 EE applications. ODN will be tested on Apache Tomcat 7+. Application server is needed for management GUIs of ODN modules.

- Relational database management system (RDBMS) - Relational database is used mainly for storing configurations of the modules (definitions of data transformation and publishing tasks, configuration of data catalogs, etc.); in these cases, relational database is preferred, because the schema of the configurations is known in advance and should not change much in the future. ODN/Storage also uses relational database to store tabular data produced by ODN/UnifiedViews. Decision is still pending about the particular database management system we will use, however, due to object relational mapping frameworks, which abstract the underlying database system, changing the database system during the works on the project is easy and straightforward.

- RDF Storage - We may use any RDF store, which is supported by openRDF API, e.g.:
  - Openlink Virtuoso (http://virtuoso.openlinksw.com/rdf-quad-store/)
  - Sesame (http://www.openrdf.org/). RDF store is used by ODN/UnifiedViews to store intermediate results of the data transformations and also by ODN/Storage to store the RDF data produced by ODN/UnifiedViews.

- HTTP Server - ODN will be tested with Apache HTTP Server. HTTP Server is required for ODN/InternalCatalog, ODN/Catalog and ODN/Publication.
7.2.2. Distributed Environment

ODN modules support distributed deployment to more physical devices. Basically every artifact depicted in Figure above may be deployed on a different device. If ODN/UnifiedViews - Engine and ODN/UnifiedViews - Management GUI is placed on different devices, administrator has to set up shared network file system both these artifacts may use.

Typically, we expect such deployment in large organizations with more complicated IT architecture driven by (among other things) security requirements. In this case:

- ODN/UnifiedViews is expected to be deployed within internal, restricted segment along with ODN/Management,
- ODN/Publication is expected to be in DMZ segment accessible from the outside by
general public (like a typical webserver) and

- ODN/Storage is expected to be in internal or other appropriate segment, reachable by both ODN/UnifiedViews and ODN/Publication

7.2.3. Custom Environment

As the design of ODN is modular (driven in particular by our vision, plans, engineering experience, etc., but in general also by best practises in software development and Open Source development) individual user will have possibility to not just move the individual modules between multiple machines, but also to skip (do not deploy) certain modules, if they do not need them.

For example, ODN/InternalCatalog and ODN/Catalog may be skipped, in exchange for direct integration between ODN/UnifiedViews and ODN/Publication with the particular national data catalog.

Note: Given proper modular design and implementation, exact options and combinations are wide and depend strongly on particular user and his needs and use cases, so we are not able to (and will not) document all possible options here. Some most common cases will be later explained in ODN documentation and COMSODE Methodology.
8. Licensing

UnifiedViews (ODN/UnifiedViews module) is licensed under the combination of GPLv3 and LGPLv3, please see [https://github.com/UnifiedViews/Core](https://github.com/UnifiedViews/Core) for details.

ODN/Management, ODN/Storage, ODN/Publication, ODN/InternalCatalog, and ODN/Catalog will be licensed under a suitable open source license: GPL, APL or BSD - the exact choice will depend on what libraries will be selected for implementation and thus this will be concluded (and documented) during WP4.

ODN/InternalCatalog, ODN/Catalog may reuse already existing catalogs, such as Dkan. In that case, we have to respect the open source license under which the data catalog tool is already released.
9. Abbreviations used

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPU</td>
<td>Data Processing Unit - component in UnifiedViews that can execute a transformation of data</td>
</tr>
<tr>
<td>DCAT</td>
<td>Data Catalog Vocabulary (<a href="http://www.w3.org/TR/vocab-dcat/">http://www.w3.org/TR/vocab-dcat/</a>)</td>
</tr>
<tr>
<td>ETL</td>
<td>extract, transform, and load (ETL) refers to a process that: ● Extracts data from outside sources ● Transforms it to fit operational needs, which can include quality levels ● Loads it into the target store (database, data mart, or data warehouse)</td>
</tr>
<tr>
<td>ODN</td>
<td>Open Data Node</td>
</tr>
<tr>
<td>SPARQL</td>
<td>SPARQL Protocol and RDF Query Language (<a href="http://www.w3.org/TR/rdf-sparql-query/">http://www.w3.org/TR/rdf-sparql-query/</a>)</td>
</tr>
<tr>
<td>UV-core</td>
<td>Unified Views core components (<a href="https://github.com/UnifiedViews/Core">https://github.com/UnifiedViews/Core</a>)</td>
</tr>
<tr>
<td>UV-plugins</td>
<td>Unified Views plugin components (DPUs) (<a href="https://github.com/UnifiedViews/Plugins">https://github.com/UnifiedViews/Plugins</a>)</td>
</tr>
</tbody>
</table>

*Table 4: Abbreviations used*
10. Glossary

**Atom feed** - a published list (or “feed”) of recent articles or content in a standardized, machine readable format that can be downloaded by programs that use it ([http://en.wikipedia.org/wiki/Atom_(standard)]). It is an alternative to RSS.

**Data catalog** - a database where are stored metadata about datasets.

**Data catalog vocabulary (DCAT)** - an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web.

**Data mart** - access layer of the data warehouse environment that is used to get data out to the users ([http://en.wikipedia.org/wiki/Data_mart]). In ODN it is the database, which is used by ODN/Publication to generate data dumps, provide data as results of API calls.

**Data Processing Unit** - see DPU

**Dataset** - one or more related input files ([http://en.wikipedia.org/wiki/Data_set]). In ODN, dataset is processed as input and also ODN produces datasets as output.

**Dataset record** - record about the dataset containing metadata of the dataset, associated transformation pipelines, configuration of the dataset publication, status of the last dataset publication

**Data Transformation Pipeline** - oriented acyclic graph, where nodes represent DPU instances, and oriented edges represent data flow between these instances. Every pipeline consists of one or more DPs. ODN/UnifiedViews supports exchange of RDF data between DPs.

**Data Transformation Task** - see Data Transformation Pipeline

**Data unit** - an input to DPU or output from DPU is called data unit (DU). Every DPU may have more input data units and more output data units. DPUs may also use different types of DU, e.g., RDF data unit being able to read/write RDF data or File data unit being able to read/write generic files.

**DPU** - plugin on the data processing pipelines, which executes certain transformation, cleansing, quality assessment on top of the processed data. DPU encapsulates certain business logic needed when processing data (e.g., one DPU may extract data from a SPARQL endpoint or apply a SPARQL query). Every DPU must define its required/optional inputs and produced outputs.

**DPU instance** - placement of DPU on a pipeline. DPU instance is therefore created when DPU is placed on pipeline canvas.

**DPU configuration** - associative array of key-value pairs, which customize functionality of DPU instance.

**External data catalog** - for ODN it is data catalog that is not part of ODN but ODN can be
integrated with it. see also Data catalog

**Instance Configuration** - configuration for specific DPU instance. It is created at the time of placing DPU on the pipeline (canvas) as a copy of **Template Configuration**

**Internal data mart** - the database where the results of the ODN/UnifiedViews is stored when the pipeline finishes. see also Data mart

**Public Data catalog** - Data catalog that is part of ODN and is visible to public. In ODN is also referenced as Data catalog.

**RDF** - standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. ([http://en.wikipedia.org/wiki/Resource_Description_Framework](http://en.wikipedia.org/wiki/Resource_Description_Framework), [http://www.w3.org/RDF/](http://www.w3.org/RDF/))

**Resource Description Framework** - see RDF

**Representational state transfer** - (REST) software architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed hypermedia system and also applied to the development of web services. ([https://en.wikipedia.org/wiki/Representational_State_Transfer](https://en.wikipedia.org/wiki/Representational_State_Transfer))


**Staging database** - the database used by ODN/UnifiedViews to store intermediate results of the DPUs on the pipeline

**Template Configuration** - previously called "default configuration", default DPU configuration associated with each DPU.

**Transformation Pipeline** - see Data Transformation Pipeline
11. References

1. http://www.comsode.eu/ the main page of project COMSODE
12. List of figures and tables

Figure 1: Open Data Node business process
Figure 2: Use cases for data publisher
Figure 3: Use cases for data consumers
Figure 4: ODN modules overview
Figure 5: UnifiedViews introduction
Figure 6: UnifiedViews components and dependencies
Figure 7: Context of ODN and UnifiedViews
Figure 8: Methods available to ODN/Management in RESTful format
Figure 9: Structure and context of ODN/Storage
Figure 10: ODN/Publication module
Figure 11: Data.Gov.uk catalogue view
Figure 12: ODN/Management module
Figure 13: Context of ODN/Management module
Figure 14: Tight integration scenario
Figure 15: Loose integration scenario
Figure 16: A No integration scenario
Figure 17: Deployment on server - infrastructure

Table 1: Use cases for data publisher
Table 2: Use cases for data consumers
Table 3: Comparison of CKAN and DKAN features
Table 4: Abbreviations used