# metaphactory for Massive Graphs

Aaron Eberhart
metaphacts GmbH
Germany
ae@metaphacts.com

Peter Haase
metaphacts GmbH
Germany
ph@metaphacts.com

Wolfgang Schell
metaphacts GmbH
Germany
ws@metaphacts.com

## ABSTRACT

Knowledge Graphs and semantic technologies allow scientists and domain experts to model complex relations between data in a logically structured and machine readable format. metaphactory is a platform that enables users to build these kinds of semantic graphs easily and efficiently. metaphactory uses standards such as RDF in combination with OWL, SKOS, SHACL, and others to provide a flexible endpoint to interact with graphs of varying complexity and expressivity. As part of the Graph-Massivizer project, metaphactory is supporting integration and infrastructure consolidation for components developed in the project. Part of this work is to develop a toolkit which metaphactory uses to process very large graphs without sacrificing sustainability. In this paper we describe in detail the metaphactory platform and how it supports large-scale graph processing in the Graph-Massivizer project, as well as outlining the current efforts within the project and how they aim to increase capabilities in the present to support future work.

## CCS CONCEPTS

• **Information systems → Data exchange**; **Mediators and data integration**.

## KEYWORDS

Graph-Massivizer; metaphactory

**ACM Reference Format:**
Aaron Eberhart, Peter Haase, and Wolfgang Schell. 2023. metaphactory for Massive Graphs. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3578245.3585330

## 1 INTRODUCTION

Knowledge Graphs and semantic technologies allow scientists and domain experts to model complex relations between data in a logically structured and machine readable format. These graphs are often extremely useful for understanding the meaning of large datasets, but can be limited in size due to the need to manually curate many very specific expressions. This limitation is not inherent in the graph semantics or technology itself, but rather in the time and space required to develop and store such a graph. When we want to create and utilize very large graph datasets we require the proper tools to ensure that our solutions remain tractable.

metaphactory is a platform that enables users to build these kinds of semantic graphs easily and efficiently. By allowing users to visually interact with the knowledge graph and build applications that use the structured data as intended, metaphactory provides a streamlined path for developing and interacting with knowledge graphs that is unhindered by the need to do everything from scratch. This utility is also generally agnostic to the underlying graph store technology used, meaning that metaphactory is able to accommodate large graphs as well as small.

Its specific strengths mentioned previously allow metaphactory to play a central role in the Graph-Massivizer [5] project. In Graph-Massivizer, a team of scientists and industry experts from diverse yet related domains are working to develop a toolkit which enables users to gather insights and reason about large scale knowledge graphs while ensuring that efficiency and sustainability are not sacrificed. metaphactory acts as an integration platform for the different parts of the toolkit, providing data management and architectural coordination support.

In this paper we first describe in greater detail what metaphactory is and discuss its strengths which feature prominently in the Graph-Massivizer project. Following that we describe the overall goals of Graph-Massivizer. Then we briefly mention the work in the other parts of the project and show how they connect to metaphactory and the overall architecture. Following that we talk about strategies metaphactory uses to represent and handle the large amounts of data required for Graph-Massivizer, then finally we talk about future work.

## 2 metaphactory

metaphactory is a platform that allows new users and domain experts alike to visually model and use semantic data stored in a knowledge graph. In Figure 1 an example of metaphactory use is shown[1]. In this picture you can see the visual editor for knowledge graphs, allowing a user to simultaneously model and edit their knowledge graph and intuitively grasp the connections between different classes and relations.

metaphactory works with many existing graph databases and technologies, such as Amazon Neptune [2], GraphDB [3], RDFox [4], Stardog [5], and more. This flexibility is one of metaphactory's great strengths, and ensures that diverse use cases with both extremely large as well as small graphs can be supported by avoiding overcommitment to one particular technology. One of the major factors


*GraphSys '23, April 15–16, 2023, Coimbra, Portugal*

ACM ISBN 979-8-4007-0072-9/23/04…$15.00
https://doi.org/10.1145/3578245.3585330

[1] Additional examples: https://metaphacts.com/product/semantic-knowledge-modeling
[2] https://aws.amazon.com/neptune/
[3] https://www.ontotext.com/products/graphdb/
[4] https://www.oxfordsemantic.tech/product
[5] https://www.stardog.com/

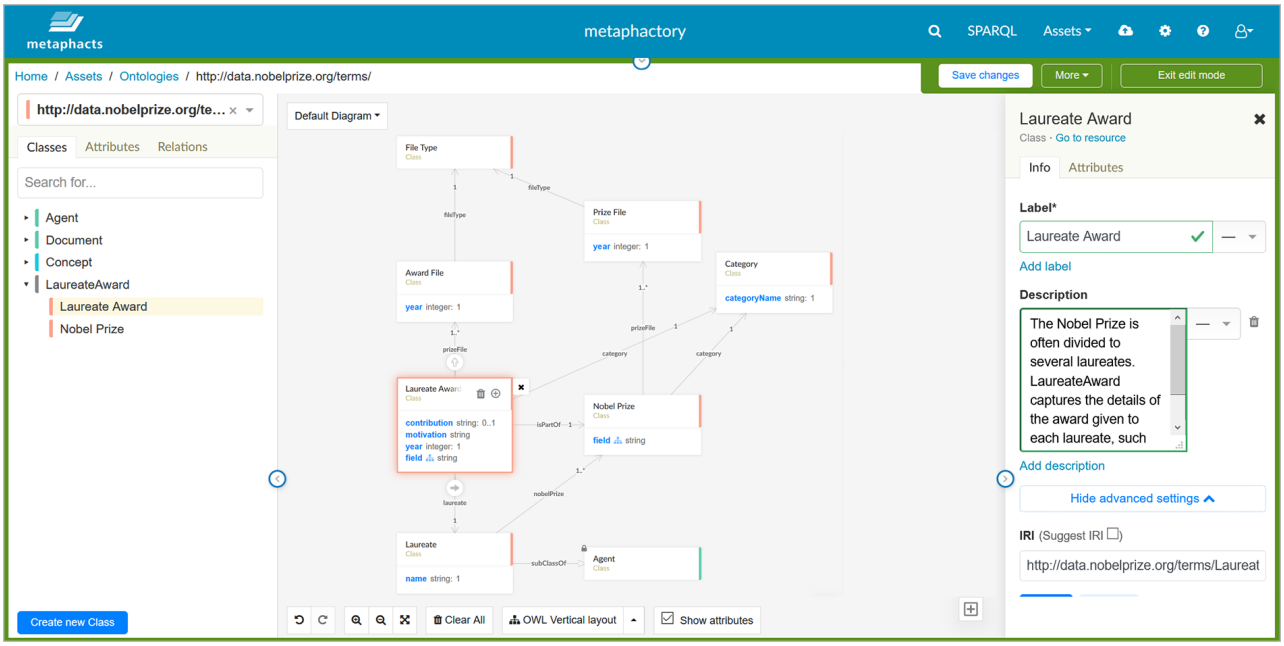


**Figure 1: Example Usage of metaphactory**

that makes this flexibility possible is by adhering to community standards so as to maintain broad capability to switch between different formats.

## 2.1 Standards

metaphactory uses standards such as the Resource Description Framework (RDF) [7] in combination with Web Ontology Language (OWL) [1], Simple Knowledge Organization System (SKOS) [3], Shapes Constraint Language (SHACL) [2], SPARQL Protocol and RDF Query Language (SPARQL) [**?** ], and others to provide a flexible endpoint to interact with graphs of varying complexity and expressivity. The subsequent list of standards we support is of primary relevance for the Graph-Massivizer project and should not be considered exhaustive.

*2.1.1 RDF.* A common format for data interchange on the web, RDF is also a common tool for representing knowledge graphs. RDF data is written as a set of node-edge-node triples which form a graph.

*2.1.2 OWL.* OWL is a format for expressing more abstract and complex axioms with logical notions than standard RDF. OWL is serialized in RDF so the two are compatible, even though the semantics are slightly different. In general OWL is more expressive than pure RDF but also has far more difficult reasoning problems due to the added complexity.

*2.1.3 SHACL.* A constraint language, SHACL, is used for augmenting RDF expressivity with shapes. It is in certain respects less expressive than OWL, but the two are both based in RDF and share much in common. SHACL is often used for data validation by placing constraints on data which determine how it should appear when

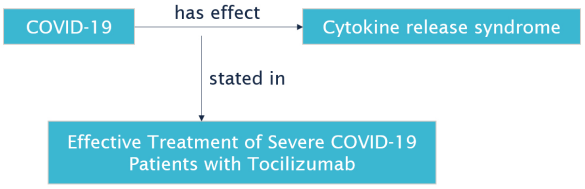


**Figure 2: A Labeled Edge In RDF-star**

it is valid. OWL and SHACL are often used when defining a data model or schema for an application.

*2.1.4 SPARQL.* Whenever semantic data in RDF needs to be queried SPARQL is the standard for this type of operation. SPARQL queries look similar to queries in other popular query languages, however they are specialized to work on RDF graphs. SPARQL can also be used to insert or remove data as needed.

*2.1.5 SKOS.* SKOS is a standardized way to improve data interoperability on the web using a set of RDF and OWL axioms. This works within existing technologies and contains a predefined way in which data should connect and interact across different sources and domains. SKOS can also be used to define controlled vocabularies an taxonomies.

*2.1.6 RDF-star.* RDF-star is a superset of RDF which allows labels on edges, as seen in Figure 2. This small addition is quite powerful with the additional types of annotations and descriptions it permits, yet has little effect on the technical complexity of problems that can be solved. Labelled edges can be textual descriptors or even numerical weights, it is even possible to use them as simplifications

for more complex expressions in native RDF such as reification. RDF-star can be used as a compatibility layer to Labeled Property Graphs (LPG) as it provides a bridge to the full set of expressivity of the LPG model.

## 2.2 Technical Advantages

The metaphactory platform has many technical advantages for the Graph-Massivizer project. The first has already been mentioned, though it is worth repeating, that metaphactory works with existing graph databases, and rather than reinventing the wheel it instead provides a flexible interface for working with exiting technologies. This means that the size of a massive graph is not inherently a limitation so long as technologies exist to interface with them, like the kind we see in this project.

Another advantage of metaphactory is that it enables the use of serverless deployment for graph processing technologies it interfaces with. This is especially useful when dealing with very large datasets as compute power can scale as-needed without the necessity of running a single massive instance on one server or machine. A serverless deployment also allows for simplified local infrastructure by relying on a coordinated data infrastructure in the cloud which can more efficiently manage resources and potentially leverage this for more sustainable execution of heavy workloads.

metaphactory also supports sustainability efforts in various ways. The first major sustainability aspect is direct: by using efficient and highly optimized infrastructure with metaphactory Graph-Massivizer ensures that there is minimal waste when graph computations are performed. There is also a more personal aspect to sustainability, whereby proper data management can lead to more efficient workflows and internal processes. Not only does metaphactory support sustainability directly, but also can be a powerful tool for managing metadata and efficient interoperability between separate subsystems, thereby reducing the indirect waste of effort and time.

# 3 GRAPH-MASSIVIZER

Graph-Massivizer seeks to develop a high-performance, scalable, and sustainable platform for information processing and reasoning based on the massive graph representation of extreme data. It contains a toolkit of five open-source software tools for use with graph datasets. In this section we will describe metaphactory's role in the Graph-Massivizer project and situate it within the other connected efforts. Finally we will give a brief overview of the system architecture for the Graph-Massivizer toolkit.

## 3.1 Graph-Massivizer and metaphactory

Within the Graph-Massivizer project, metaphactory is responsible for a precise range of goals. The abbreviated list of objectives for the project and development is shown next.

- Define the Graph-Massivizer platform comprising the five researched tools and their deployment, integration and testing protocols and procedures
- Define and provision a federated hardware and software infrastructure for the development, integration and testing of the five Graph-Massivizer tools and the four use cases
- Integrate the five Graph-Massivizer tools
- Develop, port, and deploy the pilot use cases on the Graph-Massivizer platform
- Perform testing of the Graph-Massivizer tools
- Deploy and operate the Graph-Massivizer platform

The four use cases identified in the project are: sustainable green finance, global environment protection foresight, green AI for the sustainable automotive industry, and a data centre digital twin for exascale computing. Clearly metaphactory is involved in all aspects of this effort by coordinating the architecture and providing a unified data model between subsystems.

## 3.2 Graph-Massivizer Components

As part of the Graph-Massivizer project there are many sub-tasks which all play an important role in the toolkit's overall functionality. metaphactory integrates data from them all, so they are now briefly summarized.

*3.2.1 Graph-Massivizer.* Graph-Massivizer is the toolkit which binds together all other parts of the architecture. It is available to all partners for use and runs with metaphactory.

*3.2.2 Graph-Inceptor.* The Graph-Inceptor realizes a massive graph for the system to use and has optimizations for creation and storage of large amounts of graph data.

*3.2.3 Graph-Scrutinizer.* The Graph-Scrutinizer has query and inspection capabilities as well as probabilistic reasoning for insights on the data.

*3.2.4 Graph-Optimizer.* The Graph-Optimizer ensures that large graph operations are completed in an efficient way, while capturing data about performance and workload for further insights and optimization.

*3.2.5 Graph-Greenifier.* Graph-Greenifier evaluates the energy consumption of massive graph operation and provides data about efficiency and the proportion of renewable energy used.

*3.2.6 Graph-Serverlizer.* The Graph-Serverlizer allows for serverless deployment of the Graph-Massivizer toolkit so that resources can be used on-demand whenever possible to avoid wasting computing resources.

## 3.3 Architecture

The intended architecture for the Graph-Massivizer toolkit is shown in Figure 3. There are two primary layers in the architecture that support respectively graph usage and graph optimization.

The first layer, called the operational layer, handles external facing graph operations such as graph creation and graph querying and analytics. The subsystems responsible for this activity are the Graph-Inceptor and the Graph-Scrutinizer. This layer handles the generation of massive graphs from input data, as well as the output of structured and potentially optimized graphs or graph data enhanced with analytics.

The second layer, called the processing layer, is responsible for internal optimization and sustainability processes. The Graph-Optimizer, Graph-Greenifier, and Graph-Serverlizer utilities comprise the backbone of this layer. Here we have a layer that runs

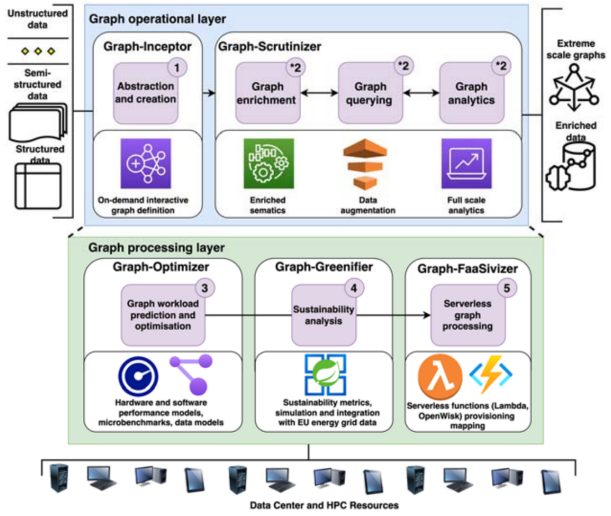


**Figure 3: Architecture Diagram**

in the background and ensures that all front-facing utilities are optimized and run sustainably.

Both layers are coordinated by the Graph-Massivizer toolkit that metaphactory orchestrates. This means that data interfaces with metaphactory are a paramount concern as all subsystems must be able to communicate in a coherent interchange format with metaphactory.

## 4 GRAPH-MASSIVIZER DATA

A primary role metaphactory plays in the Graph-Massivizer project is to facilitate data coordination across the architecture with the toolkit. In this section we discuss major relevant aspects to how metaphactory coordinates and represents data inside the system, starting with data integration and modeling, and then discussing data management and data visualization and search.

### 4.1 Data Modeling

There are various options for how a data model for Graph-Massivizer may be represented in metaphactory, and the toolkit may use any of them as required by the subsystems. metaphactory uses Findable, Accessible, Interoperable and Reusable (FAIR) [6] data principles for data management, which supports the coherent modeling of data from various disparate sources. The primary ways in which metaphactory models data follow.

- Ontology Modeling
  - metaphactory allows for visual ontology modeling and development
  - Ontologies can represent the data in a system architecture such as the Graph-Massivizer toolkit in a semantically rigorous way using OWL and SHACL
  - Visual modeling streamlines the development process and allows for improved understandability between ontology modelers who make the ontology and domain experts who utilize the ontology
- Vocabularies
  - A vocabulary is a highly reusable set of terms with definitions and often an accompanying taxonomy which defines hierarchical relations between them
  - Vocabularies expressed in RDF can be directly imported and used by metaphactory
- Combining Ontologies and Vocabularies
  - Vocabularies can link with other RDF and RDF-compatible data artifacts such as OWL or SHACL Ontologies
  - Properly linked vocabularies can enhance abstract models with large amounts of diverse and structured instance data and other resources
- RDF-star to Bridge Different Graph Models
  - When different subsystems make use of their own model, RDF-star can be used to annotate and formalize the connections between them
  - This can be done directly, or by using Property Graphs expressed with RDF-star, where the edge labels prescribe properties for each edge
  - RDF-star can also enhance analytics by associating numerical values with edges if required

### 4.2 Data Integration

In Figure 4 we can see how the different elements of the Graph-Massivizer toolkit need to communicate. metaphactory facilitates data integration by providing an exchange for data across subsystems which may have their own internal and independent data paths, as well as handling the data model for the overall architecture and allowing data queries for interactive exploration and visualization. Some systems may of course communicate directly when required, as you can see in some of the arrows in Figure 4, however metaphactory provides an interface with all components as the main entry point.

### 4.3 Data Management

As the Graph-Massivizer project grows and develops, it will likely become helpful or even necessary to provide a precise accounting of the data management plan, which metaphactory is positioned to support. Information such as provenance, licensing, versioning, and distribution identification can all be stored using RDF or RDF-star with metaphactory. All of this information is accessible just like any other data and can be queried or visualized as required by the system and use cases.

metaphactory uses data catalogs defined using the Data Catalog Vocabulary (DCAT) [4] to manage data and ensure that the data is accessible in a standardized format for reuse. Since these catalogs are often expressed as vocabularies they are able to integrate with data in metaphactory freely in the same way as a standard vocabulary.

### 4.4 Data Visualization and Search

metaphactory supports the Graph-Massivizer project not only with handling the graph data but also by providing an easy-to-use interface for accessing, visualizing, and understanding insights derived from the system. Visualization and search are both provided to end-users through simple to configure web components which can be used to implement a use case for specific information needs.

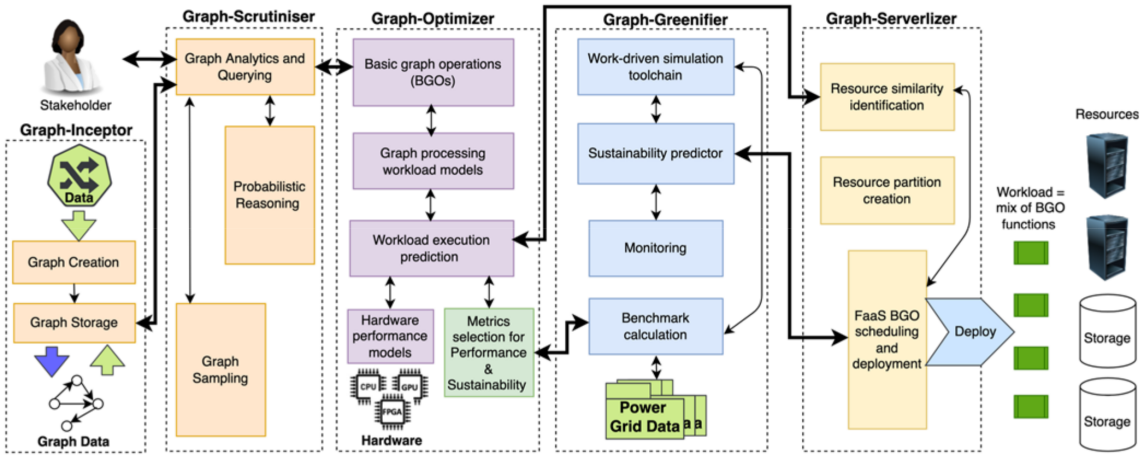


**Figure 4: Data Flow Diagram**

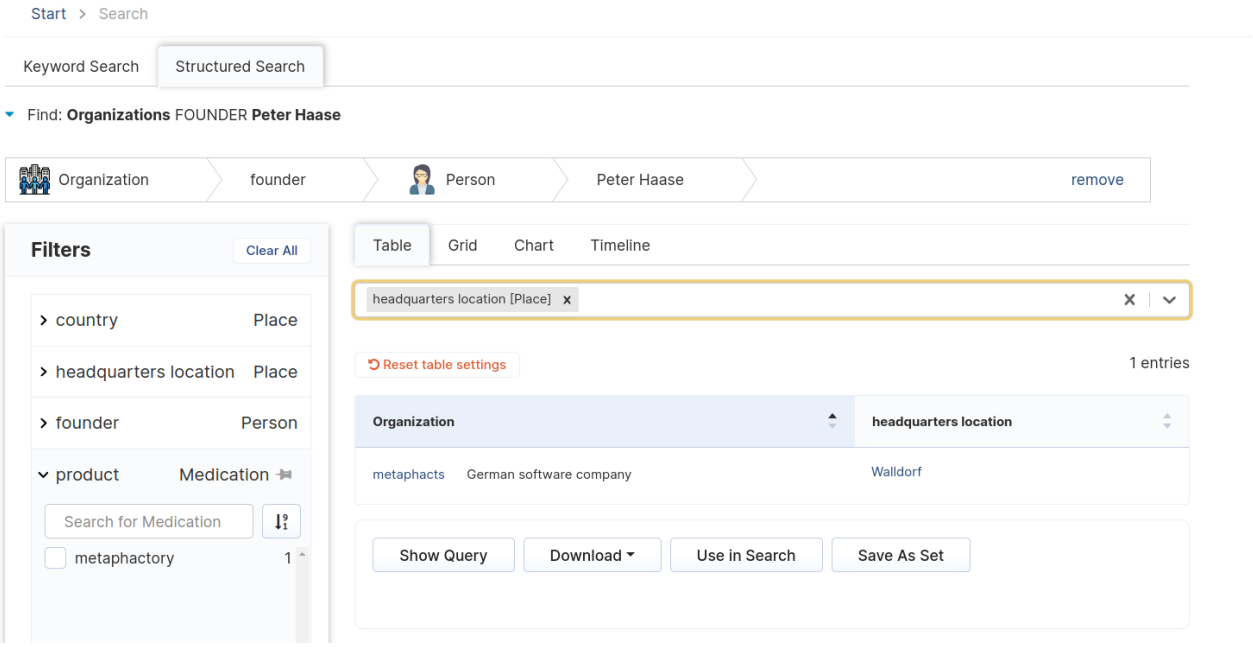


**Figure 5: Example Search**

The visualization components for knowledge graphs can be seen in Figure 1, which shows the Ontology Editor including rich graph visualization capabilities. Figure 5 shows an example of a structured semantic search. In this case we ask the system to look for organizations with the founder "Peter Haase" and then expand the result to include the headquarters location. In this case there is exactly one result but for other people and organizations we may find more entries in the table. The entire search would normally need to be performed with a complex SPARQL query in other systems, but with metaphactory we have the ability to specify the data we are looking for in a simple interactive clickable interface without needing to write or parse a dense query. This allows domain experts and non-experts to view, search, and use the data without a deep technical background in semantic technologies.

## 5 FUTURE WORK

Plans for future work for metaphactory and Graph-Massivizer are actively developing as work progresses in the project. Current plans for future work include both immediate concerns to develop the target infrastructure, as well as more ambitious high-level goals that motivate the work and may be possible in future systems.

In the short term, our primary goal is to fully initialize and coordinate the combined system in a way that can support each partner independently while also ensuring that the toolkit has

maximum interoperability. Developing data models and schemas will play a large part in these coordination efforts, and initial work has already begun on setting up an internal knowledge graph for use by the partners which can be expanded to link with other important shared resources for Graph-Massivizer.

Over the longer term, plans for future work also shows great potential. Important connections and research opportunities within the project are also being explored, with the possibility of related and mutually-beneficial research in topics such as large graph representation and management or Neuro-Symbolic AI for knowledge graphs.

**Acknowledgement** This project has received funding from the European Union's Horizon Research and Innovation Actions under Grant Agreement N° 101093202.[6]

## REFERENCES

[1] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah McGuinness, Peter Patel-Schneijder, and Lynn Andrea Stein. 2004. *OWL Web Ontology Language Reference.* Recommendation. World Wide Web Consortium (W3C). See http://www.w3.org/TR/owl-ref/.

[2] Dimitris Kontokostas and Holger Knublauch. 2017. *Shapes Constraint Language (SHACL).* W3C Recommendation. W3C. https://www.w3.org/TR/2017/REC-shacl-20170720/.

[3] Alistair Miles and Sean Bechhofer. 2008. SKOS Simple Knowledge Organization System Reference. World Wide Web Consortium, Working Draft WD-skos-reference-20080829.

[4] Andrea Perego, Simon Cox, Alejandra Gonzalez Beltran, Peter Winstanley, Riccardo Albertoni, and David Browning. 2020. *Data Catalog Vocabulary (DCAT) - Version 2.* W3C Recommendation. W3C. https://www.w3.org/TR/2020/REC-vocab-dcat-2-20200204/.

[5] Radu Prodan, Dragi Kimovski, Andrea Bartolini, Michael Cochez, Alexandru Iosup, Evgeny Kharlamov, Jože Rožanec, Laurențiu Vasiliu, and Ana Lucia Vărbănescu. 2022. Towards Extreme and Sustainable Graph Processing for Urgent Societal Challenges in Europe. In *2022 IEEE Cloud Summit.* 23–30. https://doi.org/10.1109/CloudSummit54781.2022.00010

[6] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* 3 (2016).

[7] David Wood, Markus Lanthaler, and Richard Cyganiak. 2014. *RDF 1.1 Concepts and Abstract Syntax.* W3C Recommendation. W3C. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[6] More information available at: https://graph-massivizer.eu/