SILESIAN UNIVERSITY OF TECHNOLOGY PUBLISHING HOUSE

SCIENTIFIC PAPERS OF SILESIAN UNIVERSITY OF TECHNOLOGY 2022
ORGANIZATION AND MANAGEMENT SERIES NO. 162

SEMANTIC PRODUCT PERSONALIZATION BASED ON THE COGNIPY ENVIRONMENT

Dariusz DOBROWOLSKI[1*], Jakub ŚWIĄTKOWSKI[2]

[1] Kielce University of Technology, Kielce; ddobrowolski@tu.kielce.pl, ORCID: 0000-0003-3676-9957
[2] Cognitum sp. z o.o., Warsaw; j.swiatkowski@cognitum.eu
* Correspondence author

Purpose: The aim of the publication was to visualize the process of creating knowledge on the example of the CP Factory production line. For this purpose, the data contained in the relational databases and data of the operating production line were used. This data was converted to the form required by the CogniPy environment to create a semantic ontology for product personalization.
Design/methodology/approach: With the available software based on ontologies and knowledge charts, the possibility of common human-computer reasoning has been opened, especially in production management.
Findings: During the activities carried out, the on-conceptualization of the knowledge contained in the production management system was brought to an ontological form. At the moment, the information contained in the database is not very different from the data existing in the relational database. However, further modeling of ontology can be directed towards the creation of rules, logic and axioms of production processes.
Practical implications: The operations and transformations performed presented the operation of CogniPy in the process of creating ontology and materializing the graph and queries. The created ontology takes the form of a universal set of knowledge that makes it open to wide integration with other systems.
Originality/value: The publication shows the possibilities of using the CogniPy environment in the construction of ontology and semantic product personalization.

Keywords: ontology, semantics, CogniPy, CP Factory.

Category of the paper: Research includes model construction and testing and dataset testing.

1. Introduction

In today's knowledge-based society, how it is represented is an important issue. Managing knowledge, and especially production knowledge, would be difficult if you relied only on informal records. Today, natural language sentences formulated in information systems are no

longer a problem for people. The problem arises when computers are unable to accept knowledge in the shared form. Machine-readable resources must be properly constructed, and this requirement is met by ontologies combining both formal and practical aspects.

Thanks to the developed applications based on ontologies and knowledge charts, many opportunities are opened to improve human-computer interaction, and especially in production management.

The publication focuses on the presentation of the process of creating a knowledge graph on the example of the CP Factory production line. For this purpose, it was necessary to use data managed by a relational database on a working production line. The next step was to transform them into the form required by the CogniPy community to create an ontology. CogniPy itself is an ontology editor in the form of an open-licensed library in the Python programming language. CogniPy argues about creating an ontology that complies with RDF, OWL standards and thus greatly facilitates the management of ontologies and other knowledge bases using CNL (Controlled Natural Language). The created ontologies can be both presented in the form of graphs and be a source of queries in the SPARQL language.

As a result, using CogniPy, a knowledge graph was constructed, which is an image of the ontology of the production process. This chart makes it easy to get acquainted with the new solutions, make it easier to edit and integrate with other production systems.

2. Ontology editors

The concept of ontology is known from philosophy, where ontology is understood as "... the theory of being, the basic branch of philosophy concerned with the study of the character and structure of reality" (Goczyła, 2011). In the context of semantic networks, the most well-known and widely accepted definition of ontology is that of Thomas Gruber: "Ontology is the formal specification of a common conceptualization" (Goczyła, 2011). In this case, formal means that it is machine-readable. Formal ontologies are defined in languages with strict syntax and precisely defined semantics. Today, there are many languages and systems based on formal ontology. There are systems derived from classical software engineering, such as ontologies expressed by UML (Unified Modelling Language), ERM (Entity Relationship Modelling) diagrams or ontologies expressed in RDF (Resource Description Framework) threes. These systems have inference capabilities that can be used in ontology processing. Logic-based (Goczyła, 2011) inference is an open-world assumption and leads to richer conclusions, while other data-based ontologies assume a closed world.

Among ontologies one can find a division into general, high-level, foundation ontologies – very general concepts taken from the theory of being from philosophy are defined and serve as foundations, on which further more specialized ontologies are superimposed, such as domain and application.

In domain ontologies, concepts from general ontologies are used, while defining entities specific to a given field, for example medicine, art, science, resulting in constructions useful in many contexts and applications (Goczyła, 2011).

Application ontologies are an additional narrowing of the field, and it concerns specific applications. It is possible to apply domain ontology in many applications, and thus in many application ontologies, but it is rare to use application ontology in a system other than the one for which it was created (Goczyła, 2011).

There are various applications for creating ontology, among which we can mention:

- Protégé.
- SWOOP.
- SemanticWorks.
- Fluent Editor.
- CogniPy.

For further analysis of tools for building ontology, three were selected: Protégé – the most popular ontology editor created by scientists from Stanford University and polish firm Cognitum applications – Fluent Editor – an editor running on Microsoft Windows and CogniPy – a multi-platform editor created in Python.

Protégé

The Protégé (Stanford University School of Medicine, 2022) project originated in the eighties at The University of Stanford and is still the most popular tool for programmers to construct usable ontologies and systems based on knowledge bases. It has already been awarded for its maturity, the length of this project makes it unique. Currently, it exists in the form of many "frameworks". The desktop version**Figure 1**) is supported by various tools supporting the management of ontologies. The browser version of "Protégé" is supposed to be simpler to use, and the functionality is not inferior to the desktop version, and is available immediately after logging in at the appropriate address in Internet. Protégé is built on the Java platform and fully supports OWL 2 (Web Ontology Language), which is a semantic network language. The tools also include visualizations with the ability to interactively navigate ontology connections – relationships, and active tracking of inconsistencies supports their avoidance and removal during the creation of the database. The project has in the past been supported by various funds and now operates in the form of open license software.

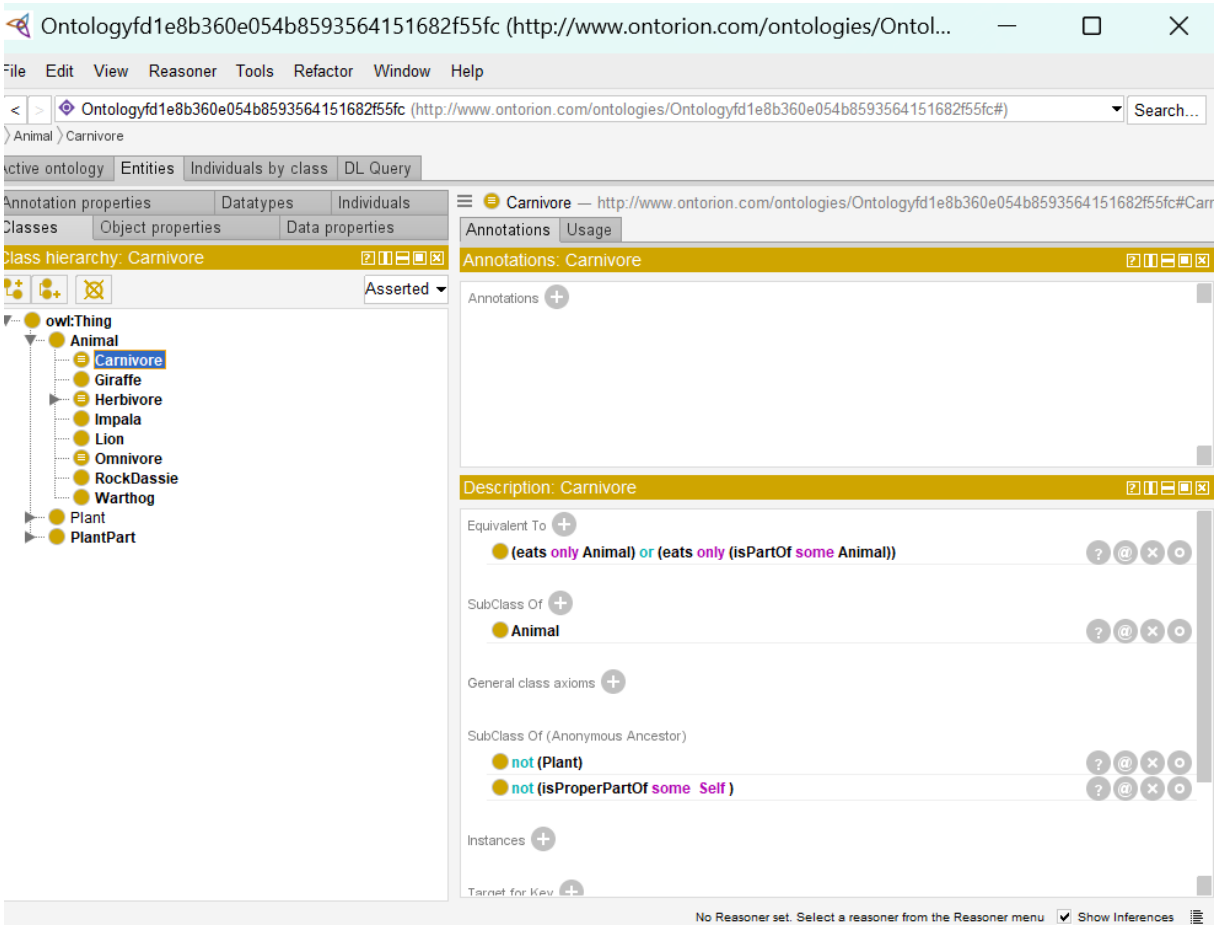

Figure 1. Example of Protégé editor.

Fluent Editor

The Cognitum FluentEditor (Cognitum, 2022) is designed for a user-friendly environment. Visually, the editor is similar to Microsoft Office, has the ribbon and customizable windows. The built-in graphics engine edits high-quality ontology diagrams (**Figure 2**). However, the most important element of the editor is the application-supported CNL (Controlled Natural Language). Allows you to easily create sentences for users who are not fluent in English.

Fluent Editor works with the aforementioned Protégé editor, which allows you to take full advantage of their capabilities.

Fluent Editor also has a built-in mechanism that allows you to monitor the course of modeling. Selecting text that contains potential errors and prompts you to edit words is a helpful attribute.

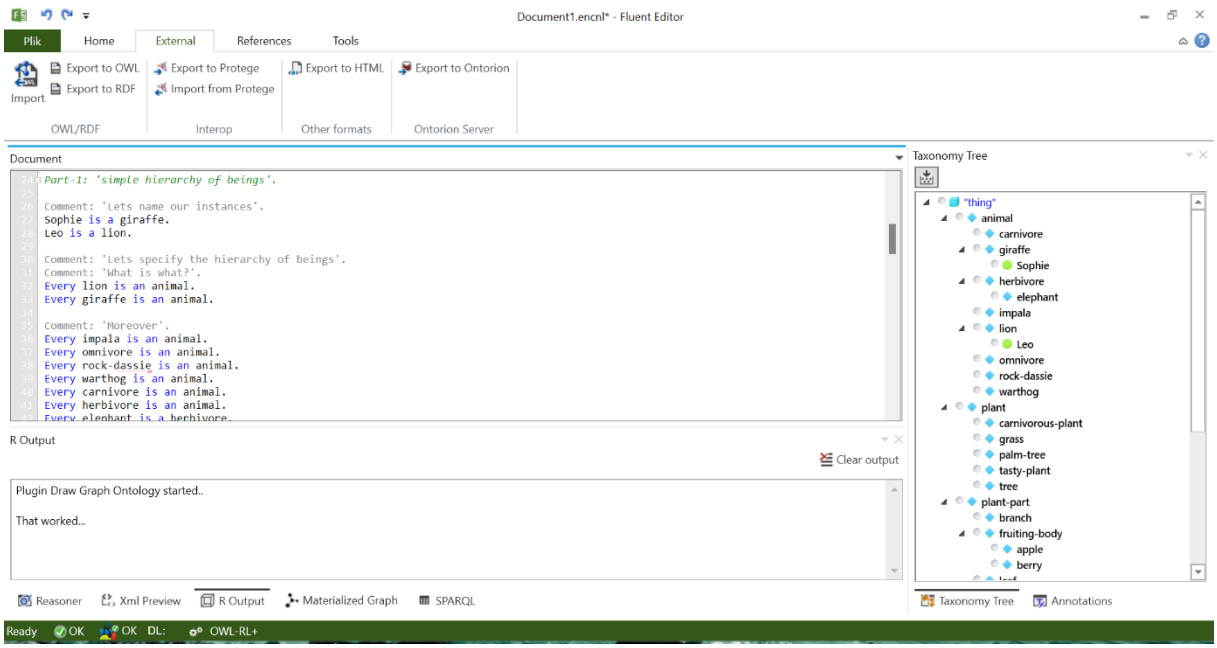

Figure 2. Fluent EditorEditor Window Example.

CongiPy is the successor to FluentEditor, it is an ontology editor in the form of a library with an open license for the Python programming language. It supports the creation of ontologies, allows them to be imported from files compliant with RDF, OWL standards and facilitates the management of them and other knowledge bases using CNL (Controlled Natural Language). Ontologies/graphs stored in the operating memory can be drawn in the form of graphs and SPARQL queries can be executed on them. The results in the form of data frames from Pandas used are a good source for further data processing. Based on the knowledge in the field of a given ontology, it is possible to infer new facts from existing ones. Going further, we can add inference rules (T-Box) on ontologies in CNL.

CogniPy is based on:

- IKVM.
- CommandLineParser.
- Newtonsoft.JSON.
- ELK.
- Hermit.
- Apache Jena.
- OWL API.

Like most programmable languages, CogniPy also presents its own example of "Hello World" (Listing 1) written in the Jupyter notebook. This is the importance of the Ontology class, creating a "hello.encl" file and typing two simple sentences into it, and then creating an object of the main CogniPy class. The last line of code prints the result of the graph query.

Listing 1

```
from cognipy. Ontology import Ontology #the ontology processing class
%%writefile hello.encnl
World says Hello.
Hello is a word.
onto = Ontology("cnl/file","hello.encnl")
print(onto.select_instances_of("a thing that says a word")[["says","Instance"]])
```

3. Production processes

The production process is a set of phenomena and carefully planned activities, performed in the right order. The process is used with raw material, which then, as part of the established scheme of activities, is subjected to processing, gradually shaping it and successively completing subsequent stages (Wawak, 2022). The result of completing all levels of production is the desired final product – the product. Currently, the production process is treated quite broadly, including research and development processes, distribution and customer service processes and the manufacturing process. The assembly system described in this publication implements the manufacturing process, and the semantic personalization of it is a process of research and development.

In the Laboratory of Modeling Intelligent Production Systems of the Kielce University of Technology, the CP Factory project is being implemented - a production assembly system. The system is divided into five modules that are separate production cells (**Figure 3**) (Luściński, 2021):

- Module 1: assembly robot station.
- Module 2: a basic linear module equipped with two application modules: tunnel heating and vision inspection.
- Module 3: basic module with branch equipped with mounting press.
- Module 4: automatic pallet storage and retrieval system.
- Module 5: basic module with branch equipped with a warehouse application (stack release).

Figure 3. CP Factory.

The dedicated production management and control system MES4 (Manufacturing Execution System) is an integral part of the modular production system. It is installed on the PC controlling the assembly process. It communicates with PLCs controlling the modules of the production system using the TCP/IP protocol.

The MES production management system enables (Lusinski, 2021):

- defining production order flows and planning a technological route,
- automatic updating of the status of orders,
- assigning a transport pallet to the order,
- defining the product and its structure, considering the graphical representation of the constituent elements,
- defining machines, including operating costs and energy consumption,
- creating storage data and material buffers,
- creating and managing customer data,
- defining the configuration of the production system; graphic, with the use of icons,
- automatic routing (routing) of the process flow in accordance with the work plan and production capacity of the machine,
- generating OEE, PLC, failures and faults reports, graphical representation of data,
- import/export of data as a file in .csv format,
- export of completed production orders in a file in .xls for further processing.

4. Implementation

The Microsoft Access database located on the production control computer on the CP Factory line has been exported to a file with the extension .accdb. Thanks to Microsoft Access, review the contents of the tables and that the exported database contains and the relationships connecting them, which is illustrated by**Figure 4**, which is a fragment of the entire database.

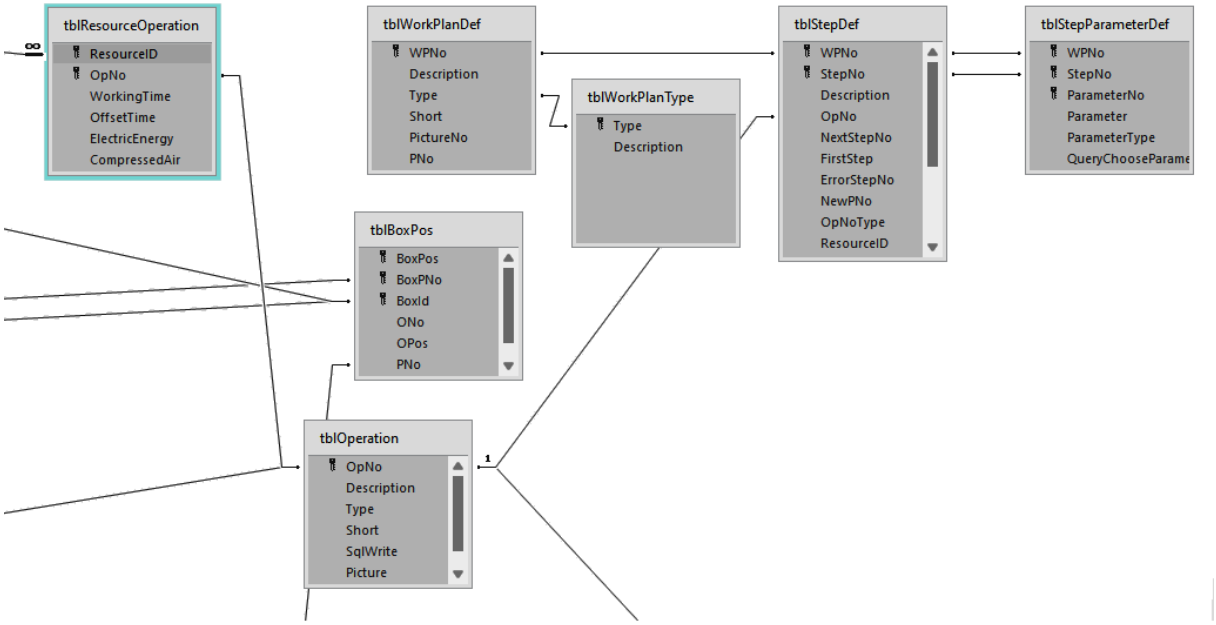

Figure 4. MES database structure.

Prepared sets of sentences based on data are provided by the Constructor of the Ontology class. Defined in Listing 2. Function a is helpful in calling the constructor. This function collects all lists with strings in one place by concatenating these strings into one string and giving them as an argument to the constructor. The "graph_attribute_formatter" function created at the beginning of the notebook is used here as the second argument to the constructor. Calling the "build Ontology" function and executing it flawlessly testifies to the correct construction of sentences and the correct creation of ontology in CogniPy.

Listing 2

```
def build Ontology():
return Ontology("cnl/string",
'\n'.join(resources_cnl)+'\n'.join(operations_cnl)+'\n'.join(work_plans_cnl),
graph_attribute_formatter = graph_attribute_formatter)
onto=build Ontology()
```

Operations on ontology

The variable "onto" is now a reference to the Ontology object. An object of this class has many methods that allow you to view the ontology in various forms and modify it. Having such an object, it is also possible to further develop ontology through methods that accept additional knowledge in cnl in the argument, getting rid of knowledge from ontology, making queries on it.

Graphical representation

To obtain a graphical representation of the accumulated knowledge, the functions from Listing 3 were called. This function creates an image based on the knowledge contained in the Ontology object and writes it to a disk with the name and extension given in the arguments of the function (**Figure 5**).

Listing 3

onto.create_graph(filename="CPFactoryOntologywithoutnoresurce.png", format="png", layout="force directed")

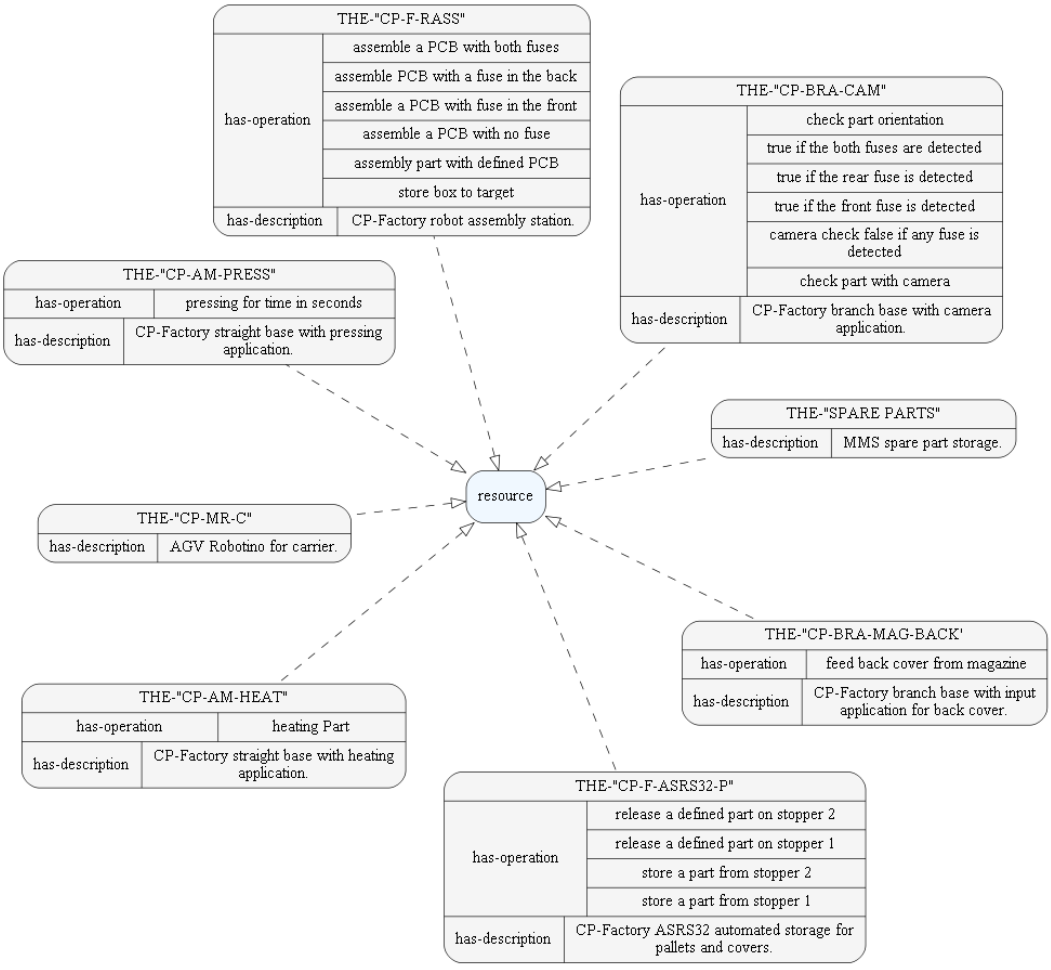

Figure 5. Knowledge Representation Graph.

The concept is a "workplan", and instances are given work plans. Each plan has its own name and attribute "has-operation" more often with many values than one. These values are operations in steps as components to be executed in the plan (**Figure 6**).

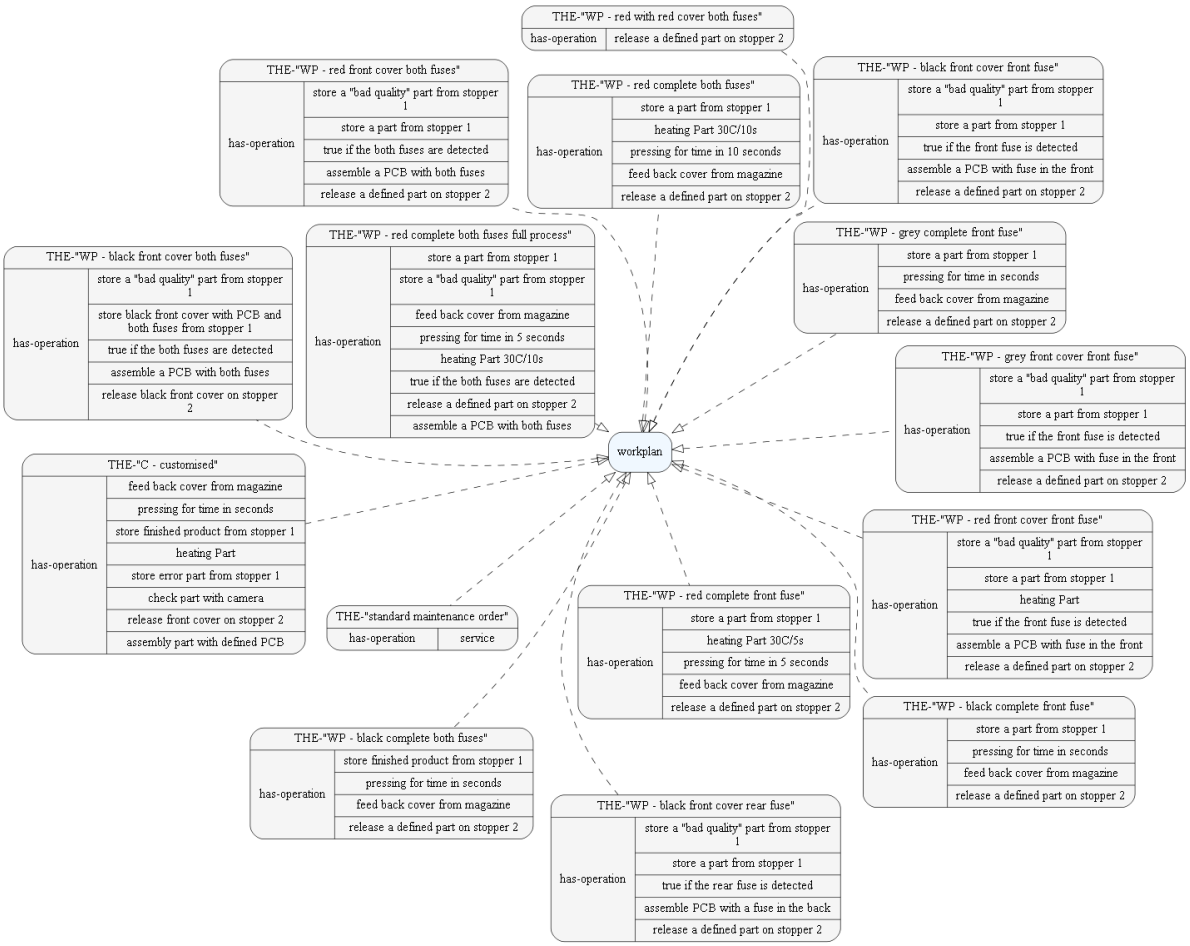

Figure 6. Workplan ontology.

Queries in ontology

Another of the possibilities offered by the ontology supported by CogniPy are graph queries. A query constructed as in Listing 4 allows you to discover the identity relationship of the values of the "has-operation" attributes in resource instances with instances of work plans. Such a comparison gives an answer to the potential question of what resources perform what operations, in which work plan. A SPARQL natural language query can be understood as follows: "Select plans, operations, resources, where the resource is a "resource" instance and has a "has-operation" attribute of a given value, and where a work plan is a "workplan" instance that has a "has-operation" attribute of the same value." After the query, the function printing the results is called, sorting them according to work plans, which is partly presented by**Figure 7**.

Listing 4

```
df=onto.sparql_query(CQL("""select ?workplan ?operation ?resource {
?resource rdf:type <resource>.
?resource <has-operation> ?operation.
?workplan rdf:type <workplan>.
?workplan <has-operation> ?operation.
}""""))
df.sort_values(['workplan'])
```

	workplan	operation	resource
1	THE-"C - customised"	feed back cover from magazine	THE-"CP-BRA-MAG-BACK"
28	THE-"C - customised"	assembly part with defined PCB	THE-"CP-F-RASS"
20	THE-"C - customised"	check part with camera	THE-"CP-BRA-CAM"
7	THE-"C - customised"	heating Part	THE-"CP-AM-HEAT"
9	THE-"C - customised"	pressing for time in seconds	THE-"CP-AM-PRESS"
31	THE-"WP - black complete both fuses"	release a defined part on stopper 2	THE-"CP-F-ASRS32-P"
5	THE-"WP - black complete both fuses"	feed back cover from magazine	THE-"CP-BRA-MAG-BACK"
11	THE-"WP - black complete both fuses"	pressing for time in seconds	THE-"CP-AM-PRESS"
41	THE-"WP - black complete front fuse"	store a part from stopper 1	THE-"CP-F-ASRS32-P"
12	THE-"WP - black complete front fuse"	pressing for time in seconds	THE-"CP-AM-PRESS"
34	THE-"WP - black complete front fuse"	release a defined part on stopper 2	THE-"CP-F-ASRS32-P"

Figure 7. Query result.

5. Summary

This publication presents the conceptualization of the knowledge contained in the production system configured by FESTO to ontological form. At the moment, the information contained in the database is not very different from those in the relational database, but that further modeling of ontology may be aimed at establishing rules, logic and axioms. The performed operations and transformations presented the operation of CogniPy in the process of creating ontology and materializing the graph and queries.

The created ontology takes the form of a universal set of knowledge, making it open to integration with other systems thanks to RDF formalism. A narrow field, production line modules and the operations performed by them can be developed by including other branches of knowledge about the production process. In this way, unified knowledge bases can be created.

This work is an introduction to the construction of an ontological production management system.

References

1. Cognitum (2022.05.30). *Cognitum.eu*. Retrieved from FluentEditor: https://cognitum.eu/semantics/fluenteditor/.
2. Goczyła, K. (2011). *Ontologies in information systems.* Warsaw: Academic Publishing House EXIT.
3. Luściński, S. (2021). *CP Factory.* Kielce: Internal materials of the Kielce University of Technology.
4. Stanford University School of Medicine (2022.05.30). *Protégé*. Downloaded from Protege: https://protege.stanford.edu/.
5. Wawak, S. (2022.05.30). *Encyclopedia of Management*. Retrieved from Production Process: https://mfiles.pl/pl/index.php/Proces_produkcyjny.