Description and Implementation of Production Factory Ontology

Filip Hardt, Martin Žáček*

Department of Computers and Informatics, Faculty of Science, University of Ostrava, Faculty of Science, 30. dubna 22, Ostrava, Czech Republic

*Email: Martin.Zacek@osu.cz

Abstract. The article follows the previous work of the authors. The paper aims to elaborate ontology into a solid form using directly for the production factory. The authors divided the article into two parts. The first part explains the basics of conceptualization, and the second part is devoted to creating and describing ontology using the RDF model. As an example of ontology's practical use, there are codes in the programming language #C.

1. Introduction

The motivation of this work of the authors is cooperation with the production factory. A manufacturing factory would like to keep up with the modern world and create a knowledge base about the factory. The company, which owns a production plant, has several factories. Therefore, there is a necessity to develop a domain ontology that, once modified, will apply to every manufacturing facility. The output of the work is ontology, which will be useful to the mPoint information system. Ontology and its application to the RDF model will model the correct hierarchy of the factory (production machines - see the basic description of the factory). The main reason for this ontology is to help record machine deterioration and predict repairs in ordering spare parts for continuous operation by factors.

2. Ontology and RDF Model

Nowadays, ontologies are in the focus primarily in relation to IT. Ontologies started to occur in acquiring, representing, and sharing knowledge in the area of knowledge engineering – a subfield of AI (artificial intelligence) [1].

We use ontology here in the sign Knowledge Engineering that tackles every single activity related to the creation of applications based on knowledge. Such activities are able to classify acquisition, representation, retention and knowledge management [2].

An ontology engineer examines relevant entities and sorts them to concepts and relations that are represented in the form of binary and unary predicates. The ontology pillar composes of generalisation/specialisation concept hierarchy [2]. Here, we devote to viewpoints connected with the productive plant. For instance, a factory sector includes a group of devices. Such devices and the sector could represent relevant concepts where the second is a super-concept of the former one. Cooperates-with might be regarded as a suitable relation valid between devices and the sector.

The RDF language, a part of Protégé, was used to describe the ontology.

Knowledge Representation (initially those included in internet resources), which take grounds in a domain ontology, was developed in the RDF framework (Resource Description Framework) model. A
model of RDF handles the semantic sides of terms defined in URI references to resources, where the meanings are clarified using a specific place in the corresponding ontology. A graphical model in RDF can be easily and simply understood by users without any knowledge of formal modelling [3]. The foundation is built on a plain statement regarding relations of items (resources) as basic vectors (see Figure. 1):

![Figure 1. RDF vector.](image)

RDF metamodels create frameworks and approaches for the purposes of expressing knowledge by a directed, labelled graph linking concerned resources. If RDF expressions describe entities, they must be dealt with as resources defined by resource identifiers - URIs. The metamodel states that an RDF statement in an RDF graph visualisation composes of two labelled nodes (subject and object) connected by a labelled edge (predicate). Each subject and object label has its URI name. Concerning the object, a literal value can be used as well). The RDFS part of the RDF enables to establish vocabularies (new terms). Such can be applied in RDF statements to particular classes or types of resources having particular attributes [4].

The representation of RDF statements in a graph is the simplest and easily understandable visual interpretation. If the first-order logic is considered, an RDF graph representation is created by atomic vectors known as elementary network statements. In the first-order predicate logic, these are symbolised by binary predicates: predicate_symbol (attribute_1, attribute_2).

3. The Description of the Production Factory
The basis of the information system for collecting data on defects in production equipment is the division of the production hall into individual parts that logically follow one another and have similar or the same semantics.

The production hall (see figure 2) of the factory (thing in Protégé) is divided into areas according to the type of equipment that is located there. These areas are further subdivided into groups of devices having a similar or identical machine ledger.

Furthermore, the individual groups are divided into lines according to the mutual connection of the equipment. The lowest level of the ontology will be the individual devices, which will have a unique identifier by which they will be distinguishable from the others. Some types of equipment may then include robotic arms to manipulate the products.

We will use the RDF language and Protégé software application to create this construction called Ontology. Based on this construction, we can perform statistical analyses of
- individual areas,
- groups,
- lines and
- devices.
4. The Ontology and Implementation

4.1. Protégé

A special Protégé tool (figure 3) was used as a tool for modeling and recording knowledge about production factors.

Each factory ontology modelled using ontology will contain a Device (class) concept that will represent the production facility.

The factory will be divided into individual FactoryDomains sectors according to the type of activity of the facility (figure 4).

Each sector contains a group of devices that have the same or similar internal structure - MachineLedger.

Each group will be divided into individual lines (Machine), interconnected production equipment Device.

Each factory will have its own specific ontology, which will differ but have the same internal structure.
Edge oriented **ontograph** representing modeled ontology of test imaginary factory named OSU (see figure 4). **Factory** is the highest possible concept (class) of ontology that represents the factories, a subset of the root class **Thing (OSU)**.

![RDF graph of ontology.](image)

**Figure 4.** RDF graph of ontology.

RDF: **Factory** - **IsSubGroupOf** – **Thing: OSU**  
RDF: **OSU** – **IsNameOf** – **Factory**

FactoryDomains is a division of a production hall according to the type of activity of the devices located in a given area; it is a logical semantic division.

RDF: **FactoryDomain** – **IsSubGroupOf** - **Factory**

According to figure 4 and figure 2 we already know that the factory is divided into individual parts - FactoryDomain. The factory is a complex factory. In order to be more oriented and able to correctly identify each device, we need to divide the areas into groups - Group.

Group represents a group of devices of the same internal arrangement/manufacturer.

In addition to the location (FactoryDomani, Group) in the factory, we also have to deal with individual equipment (see the following example).

**Example 1**

**Presses**

Since the factor contains several presses. For this reason, we must introduce a new concept - Presses. On the left side of figure 5 we can see the definition of the presses instances in the factory hierarchy.
The presses are part of the factors (according to figure 2). On the right side of figure 5, we can see defining the properties of the press object by assertions; it is part of the Factors domain.

We must not forget that each press has its name, according to its function. In figure 6, we see the definition of instance CzechPresses:

\[
\text{RDF: CzechPresses} \rightarrow \text{IsInstanceOf} \rightarrow \text{Group} \\
\text{SIMULTANEOUSLY} \\
\text{RDF: CzechPresses} \rightarrow \text{IsLocatedIn} \rightarrow \text{Pressess}
\]

**Figure 6.** Definition of CzechPresses.

**Example 2**

*CzechPress-125*

Figure 7 defines a new instance called CzechPress-125. It shows that our new press is part of the concept and the Machine and CzechPresses have the following features: Type, Name, Stations, CycleCount, CostCenter.
The relation between individual instances (Figure 8) can be written using RDF as a trio:

RDF: \textit{CzechLine-125} – \textit{IsPartOfGroup} – \textit{CzechPresses}
RDF: \textit{CzechPresses} – \textit{IsPartOfFactoryDomain} – \textit{Presses}
RDF: \textit{CzechPress-125} – \textit{IsInstanceOf} – \textit{Device}
RDF: \textit{CzechPress-125} – \textit{IsPartOfMachine} – \textit{CzechLine-125}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure7}
\caption{Definition of CzechLine-125.}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure8}
\caption{Definition relationship of CzechPress-125 and CzechLine-125.}
\end{figure}

4.2. \textit{Protégé} and RDF language

The \textit{Protégé} editor provides an advanced user interface and access to the knowledge server to make it easier for the user to model domain ontologies, modify forms for data entry, and insert individual class instances. We could see this environment in figures 3-8. In addition, \textit{Protégé} provides several plugins to editors, creating a broad base for individualizing application development. Among the most useful
is Protégé-OWL, which can generate knowledge base to RDF language [5], but also OWL (Ontology Web Language) [6].

The RDF model and OWL language use XML syntax with its own semantics for the purpose of machine understanding and transmission of the information contained in the model.

Now the examples of ontologies in XML format will follow:

Example 3
Definition of classes and their individuals
<ClassAssertion>
  <Class IRI="#Device"/>
  <NamedIndividual IRI="#CzechPresses-125"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="#Group"/>
  <NamedIndividual IRI="#CzechPresses"/>
</ClassAssertion>
<ClassAssertion>
  <Class IRI="#MachineLedger"/>
  <NamedIndividual IRI="#CzechPressesMachineLedger"/>
</ClassAssertion>

Example 4
Definitions of object properties
<ObjectPropertyAssertion>
  <ObjectProperty IRI="#isGroup"/>
  <NamedIndividual IRI="#CzechLine-125"/>
  <NamedIndividual IRI="#CzechPresses"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="#isPartOfMachine"/>
  <NamedIndividual IRI="#CzechPress-125"/>
  <NamedIndividual IRI="#CzechLine-125"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
  <ObjectProperty IRI="#isFactoryDomain"/>
  <NamedIndividual IRI="#CzechPresses"/>
  <NamedIndividual IRI="#Presses"/>
</ObjectPropertyAssertion>

Example 5
Definitions of data properties
<DataPropertyAssertion>
  <DataProperty IRI="#CycleCount"/>
  <NamedIndividual IRI="#CzechPress-125"/>
  <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">100</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="#Name"/>
  <NamedIndividual IRI="#CzechPress-125"/>
  <Literal>CzechPress-125</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
  <DataProperty IRI="#Stations"/>
  <NamedIndividual IRI="#CzechPress-125"/>
</DataPropertyAssertion>
4.3. Implementation

Now, we have shown the division of the factory into individual parts. We have explained the basic hierarchy of the factory. Since we put the individual devices in line, we need to create a new Machine (see figure 7) concept representing interconnected devices.

Figure 9 shows the implementation of the Machine concept in the Machine class in C#. Here we see that the Machine consists of interconnected devices - for that, we will use the list.

**RDF:**

\[ \text{InstanceOfDevice} \rightarrow \text{IsPartOfMachine} \rightarrow \text{InstanceOfMachine} \]

Device class is a subclass Factory which represents the production hall and includes/represents all individual instances of production facilities in the hall.

**RDF:** \[ Device \rightarrow \text{isSubclass} \rightarrow Factory \]

This trio RDF can be seen in figure 10.
5. Conclusion
The output of the article is the use of ontology in practice, in which the authors succeeded. The ontology was based on a knowledge-based production factory and was later implemented in the C# programming language. The ontology will serve as a template to describe the various factories of one company. These ontologies will only differ syntactically, but the internal structure and hierarchy will be preserved. The benefit of this approach is the interconnection of knowledge bases between individual companies that have a different internal structure.

The ontology modeling was used here Protégé unique tool that helps users to understand the problem domain.

Another continuation of the authors of this approach is to identify and define the knowledge of patterns for the production factory because as we can see in Figure 2, the factory consists of similar devices, which differs from each other only a few properties.

Acknowledgments
The research described here has been financially supported by the University of Ostrava grant SGS03/PřF/20. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the sponsors.

References
[1] Hardt F and Žáček M 2019 Ontology as a tool to describe the factory in International Conference of Numerical Analysis and Applied Mathematics 2019, ICNAAM 2019, AIP Conference Proceedings edited by Simos T.E, Simos T.E, Tsitouras Ch (American Institute of Physics, Melville, NY, 2002). In progress.
[2] Žáček M 2016 Ontology or Formal ontology in International Conference of Numerical Analysis and Applied Mathematics 2016, ICNAAM 2016, AIP Conference Proceedings 1863, edited by Simos T.E., Simos T.E., Tsitouras Ch. (American Institute of Physics, Melville, NY, 2002), DOI: 10.1063/1.4992234.
[3] Lukasová A, Žáček M and Vajgl M 2016 Carstairs-McCarthy’s Morphological Rules of English Language in RDFCFL Graphs In: 1st International Conference on Applied Physics, System Science and Computers, APSAC 2016: Lecture Notes in Electrical Engineering, Springer Verlag, 2018. pp. 169-174. ISBN 978-331953933-1.
[4] Lukasová A, Vajgl M and Žáček M 2010 Reasoning in RDFgraphic formal system with quantifier In: International Multiconference on Computer Science and Information Technology, IMCSIT 2010: Proceedings of the International Multiconference on Computer Science and Information
Technology, IEEE Computer Society, 2010. IEEE Computer Society, 2010. pp. 67-72. ISBN 978-836081027-9.
[5] Sajjad R, Bajwa I S and Kazmi R 2019 Handling Semantic Complexity of Big Data using Machine Learning and RDF Ontology Model Symmetry, 11(3), 309.
[6] Emani C K, Da Silva C F, Fies B and Ghodous P 2019 NALDO: From natural language definitions to OWL expressions Data & Knowledge Engineering. 122 130-141