DOMAIN ONTOLOGY OF THE EQUIPMENT IN MANUFACTURING SYSTEMS

ДОМЕЙН ОТНОЛОГИЯ НА ЕКИПИРОВКАТА В ПРОИЗВОДСТВЕНИ СИСТЕМИ

M.Sc. Stoyanov K.[1], Ph.D. Gocheva D.[2], Prof. Ph.D. Batchkova I.[2], Prof. D.Sc. Popov G.[1]
Technical University of Sofia[1], University of Chemical Technology and Metallurgy - Sofia[2], Bulgaria
E-mail: kostadin_sto@abv.bg, dani@uctm.edu, idilia@uctm.edu, gepop@tu-sofia.bg

Abstract: *This paper presents the developed domain ontology of the equipment in manufacturing systems in order to be used in the field of reconfigurable manufacturing systems (RMS). The ontology meets the requirements of the meta-class "Equipment", which is part of the developed meta ontology based on the standard for integrated systems for the production and management IEC / ISO 62264. The article explains various properties and class restrictions of the developed ontology. Web Ontology Language (OWL 2) and Protégé 4.3 as an editor and knowledge acquisition tool are used. Special attention is given to the use of ontology reasoning to infer additional information from the facts stated explicitly in ontology - an important feature, used to perform classification, sorting and assembly operations and consistency checking.*

Keywords: *IT, ontologies, interoperability, IEC / ISO 62264, RMS.*

1. Introduction

In the modern world the manufacturers need to satisfy the fast changing demands of the consumers in order to be competitive on the globalized market. This leads to developing new solutions in the development and manufacturing of the final product. One of the possible solutions is the development of reconfigurable manufacturing systems. These systems would allow faster implementation of the designed manufacturing systems and the result would be reduction of the time needed for the manufacturing of the final product. However, these reconfigurable manufacturing systems still have too many problems – hardware and software – that prevent their mass deployment on the world market. One of the main software problems is the interoperability issue that leads to the use of too many devices and the implementation of different software solutions in order to establish a connection and proper transfer of information between the different machines in the system. The results of this problem are increased manufacturing time and increased price for the final product. According to [1] the interoperability issue may be solved through the use of some standards such as IEC/ISO 62264, ISO 10303 (STEP) and IEC 61499 [2, 3, 4] and shared ontology. The developed domain ontology presented in this paper is one of four ontologies that are developed to match the IEC/ISO 62264 standard and merged in a developed meta ontology based on the same standard in order to work together. The purpose of the development of the four ontologies is to capture the basics of the manufacturing processes where the *Equipment* ontology is the most complex of the four because of the relations between the different individuals and classes that corresponds to different sensors, tools, machines, systems and so on. The developed ontologies will be used by specially developed software for the generation of control programs and in such way, illustrating an approach for solving the interoperability issue in reconfigurable manufacturing systems.

2. A short overview of the applied techniques

The Web Ontology Language OWL 2 [5] accepted as a World Wide Web Consortium (W3C) Recommendation from 11 December 2012 is a powerful knowledge representation language; it has been applied successfully for knowledge modelling in many application areas [6]. As a descriptive language, OWL 2 is used to express expert knowledge in a formal way, and as a logical language, it is used to draw conclusions from this knowledge. The formal semantics allows humans and computer systems to exchange OWL ontologies without ambiguity as to their meaning, and also makes it possible to use logical deduction to infer additional information from the facts stated explicitly in an ontology [7]. Every OWL 2 ontology is a machine-processable formal description of a domain of interest and consists of the following three different syntactic categories: *Entities*, *Expressions* and *Axioms*. *Entities*, such as classes, properties, and individuals are identified by IRIs. They form the primitive terms of an ontology and constitute the basic elements of ontology. *Expressions* represent complex concepts in the domain being described: new classes as a result of *Intersection*, *Union*, *Negation*, *Existential* and *Universal Property Restrictions, Cardinality Restrictions*. *Axioms* are statements that are asserted to be true in the domain being described and allow relationships to be established between *Expressions*: *Subclass Axioms, Equivalent Classes, Disjoint Classes, Subproperties, Equivalent Properties, Disjoint Properties, Inverse Properties, Property Domain, Property Range, Inverse, Functional, Transitive Properties, etc.* These three syntactic categories are used to express the logical part of OWL 2 ontologies - that is, they are interpreted under a precisely defined semantics that allows useful inferences to be drawn [5]. The ability to infer additional knowledge (deductive *reasoning* [6]) is of great importance for designing and deploying OWL ontologies. A particular kind of deductive reasoning on the *ClassAssertion* axiom, the task of computing the individuals that belong to a given class (or set of classes) is called *instance retrieval*. If the task is to find out whether one particular individual belongs to the given class, it is called *instance checking*. Analogous tasks exist for *SubClassOf* axioms: computing all subclass relationships between a set of classes is called *classification*, and checking a particular subclass relationship is called *subsumption checking* [6]. Very important reasoning task is *consistency checking*, the task of determining whether a class or an ontology is logically consistent or contradictory. Instance retrieval and classification tasks can be solved by using many individual instance and subsumption checks. The concepts of *soundness* (all computed inferences are really entailed), *completeness* (all entailed inferences are really computed) and *computational complexity* (time and resources needed for a reasoning task) are very important for the choice of suitable reasoner. Lack of *completeness* is sometimes acceptable if it allows for simpler or more efficient implementations, but the lack of *soundness* is usually not desirable [6]. Sound and complete OWL 2 reasoning is of high complexity - double exponential computational complexity - N2ExpTime. The OWL 2 new profiles (OWL 2 EL, OWL 2 QL, OWL 2 RL) restrict the used syntactic categories to improve complexity and practical performance, however with limitation of expressivity. Since the best balance between language expressivity and reasoning complexity depends on the intended application [7], and modeling of the equipment in the manufacturing systems needs more language constructions than the new OWL 2 profiles offer, we decided to tackle computational complexity using the optimized reasoning algorithm - HermiT. In [8] the performance of reasoning in HermiT is compared with that of FaCT++ and Pellet - two other popular and widely used OWL 2 reasoners. HermiT ontology reasoner supports all features of the OWL 2 ontology language, and it correctly performs both object and data property classification/reasoning tasks and is much faster than other reasoners. HermiT consists of components that together implement a sound and complete OWL reasoning system [8]. HermiT also implements a novel classification algorithm that greatly reduces the number of consistency tests needed to compute the class and property hierarchies [8].

3. Domain ontology “Equipment”

The developed ontology contains 156 classes, 1411 axioms, 34 properties, 57 different individuals. Protégé 4.3 is used as the most popular free ontology editor and HermiT 1.3.8 is used as a highly-efficient OWL Reasoner suitable for the domain of the equipment in the manufacturing systems.

3.1. Classes

The main class in this ontology is the class *Enterprise*. The complementary class *Value Partitions* is used as a “design pattern”. The class *Value Partitions* contains covering axioms used in different classifications, sorting and assembly operations using the HermiT reasoner. The class *Enterprise* follows the terminology, models and structure of a general enterprise in accordance with the world-wide accepted standard IEC/ISO 62264 for enterprise-to-control system integration [3] and has the following hierarchy – *Enterprise, Site, Area, ProductionLine, WorkCell, EquipmentModule* and *ControlModule* (each class is a sub class of the previous one). The class *ControlModule* is the lowest level of the hierarchy defined by the IEC/ISO 62264 standard and contains two classes - *Sensor* and *Actuator*. The *Sensor* class contains a collection of classes associated with different types of sensors. The physical object (the sensor itself) is presented via individuals described by various data properties and asserted to their appropriate class. The class *EquipmentModule* contains few additional classes except the class *ControlModule* which are *Accessory, Module and Tool*. The *Accessory* class includes a collection of classes for different accessories that could be added to the different machines such as vises, safety guards, and lamps and so on. The named *Module* class contains *CNCMachineTool_Module* and *FESTOModule* where the latter contains collection for the modules needed for the assembly of the FESTO Sorting, Processing and Handling workstations. Similarly, to the sensors each module is described via individuals and data properties. The named class *Tool* contains a collection of named and defined classes that have restrictions and can classify different individuals by different characteristics. For example, the two defined classes *AllCarbideTool* and *AllHSSTool* will collect all tools of the ontology which are made of carbide or HSS (High Speed Steels) respectively when the ontology is inferred. Each tool is described in a similar way as the modules and the sensors. The other sub classes of the class *Tool* are *DrillingTool, MillingTool, ReamingTool* and *TappingTool*. Each one of the classes contains individuals describing different tools and asserted to appropriate tool type class. Through the use of defined sub classes, the ontology infers some sorting operations such as parting the different tools according to the material they are made of, according to preliminary defined diameter range or according to the tool size (large, medium or small). For instance, the named class *DrillingTool* contains the following named classes *DrillingTool_Diameters, DrillingTool_MadeOf* and *DrillingTool_Sizes* and one defined class that will sort all of the drilling tools according to preliminary defined range for the diameters of the drills when the ontology is inferred - *DrillingTool_DiametersRange*. The class *DrillingTool_Diameters* contains named classes for the different diameters of the drills and each individual is asserted to the appropriate class according to their diameter. For example, the named class *DrillingTool_Fi_1.6* contains all drilling tools of the ontology that have a diameter of 1.6 [mm]. The purpose of this classification is to create a location for the created individuals through some of their main properties. In our case this is the diameter of the tool (for cylindrical tools). From that point on it is better to create defined classes that can allocate the individuals according to their properties and the intended needs of the ontology because every single individual may have hundreds of properties and doing the job manually is nearly impossible. The *DrillingTool_MadeOf* contains two defined classes that can allocate the individuals according to the material they are made of (*DrillingTool_Carbide* and *DrillingTool_HSS*). As for the last class - *DrillingTool_Sizes* – it sorts the named classes for the diameters of the individuals into groups in a broader meaning (*Large*, *Medium* or *Small*) according to preliminary defined diameter range for each group. The classes and the operations for the other types of tools are similar to the *DrillingTool* class and will be not discussed any further. The hierarchy of the *Tool* class is shown on Fig.1.

The *WorkCell* class is equivalent to machine level. On this level sets of named and defined classes can be created and through the use of object properties different individuals from the lower levels of the hierarchy can be linked to form machines. For instance, there is a sub class of the *WorkCell* class (*FESTOWorkstantions*) that when the ontology is inferred all of the appropriate sensors and modules of the lower levels of the hierarchy will be allocated to form groups of the FESTO workstations (Sorting, Handling, Processing and so on) as single machines. From this point on for the super classes of the *WorkCell* class (*ProductionLine, Area, Site*) it is better to use covering axioms for the creation of different systems, areas and sites. For example, the *ProductionLine* class contains four defined classes. These classes correspond to the level of a system (each class represent different system) so when the ontology is inferred these four defined classes form four different systems. Each system is composed of at least two FESTO workstations that work together and that were defined on the lower level (on the level of the *WorkCell* class) as machines There is no limitation to the maximum number of FESTO workstations that are composing each one of the systems.

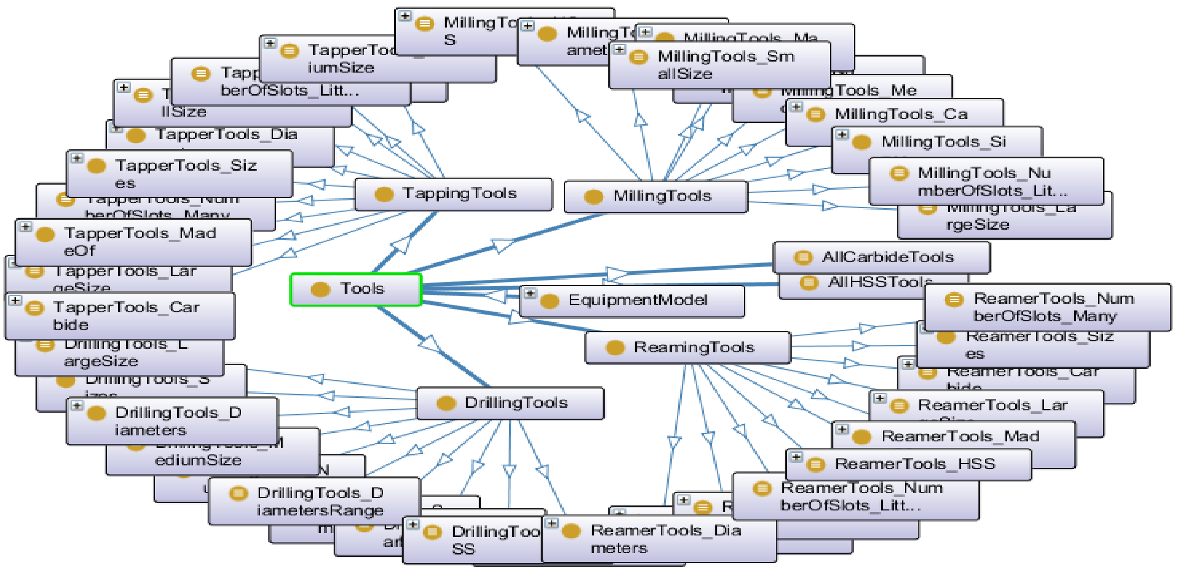


Fig. 1: Tool class hierarchy

3.2. Properties

A. Object properties

The ontology has ten object properties divided into three groups. The first one is *hasDiameter* and it is used alongside with the defined class *Tools_Size* as a covering axiom for the size of different tools. It is used as a restriction of the classes and defines three groups of sizes – *Large* for cylindrical tools with diameters larger than 10 [mm], *Medium* diameters in the range of 5 [mm] to 10 [mm] and tools with *Small* diameters which are smaller than 5 [mm]. As you can notice all of the ontology tools are cylindrical but that doesn't mean that the covering axiom is restricted to cylindrical tools only. For example, we may create another object property *hasInsertSize* and to use it to create covering axioms for prismatic tools for instance and so on. So when the class hierarchy is inferred, all of the tool classes are allocated to their places, so in this example we have allocation of classes while the second object property of the ontology is used to allocate individuals. The second object property group of the ontology is *isPartOf* and contains the sub properties *isPartOfSensor*, *isPartOfModule*, *isPartOfMachine* which are used to illustrate the relation of the different individuals among each other. As was mentioned earlier in this paper the classes *Modules* and *Sensors* hold collections of different modules and sensors. Each individual in this collection is asserted the appropriate object property. For example, the individuals that represent different sensors are related to the appropriate module via the object property *isPartOfModule*. In this way if we accept that a collection of sensors that are part of some module, and that a collection of different modules forms a machine then we can create defined sub classes of the class *WorkCell* and easily infer the knowledge for the parts that are needed to assemble a machine by allocating the different individuals to their respective defined class. This method is illustrated by three of the FESTO Workstations – Sorting station, Handling station and Processing station. Each station consists of assembly plate (in the ontology it is referred as the appropriate *Base*), modules and sensors. The *Base* is (as an individual) located in the named class of the station, then we create defined sub classes for the available modules and sensors and by inferring the ontology all of the individuals are allocated to these classes by forming a compact description of the elements needed to assemble the appropriate workstation. The relations among the individuals are shown on Fig. 2.

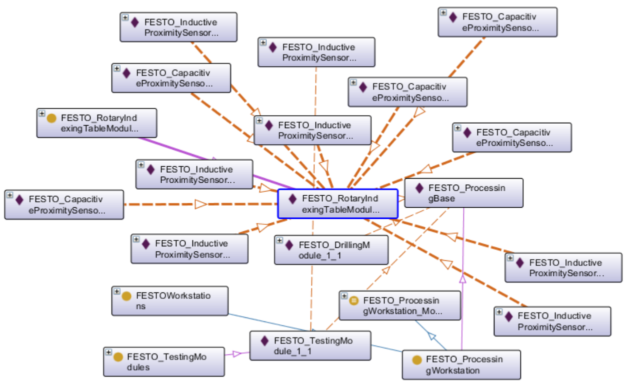
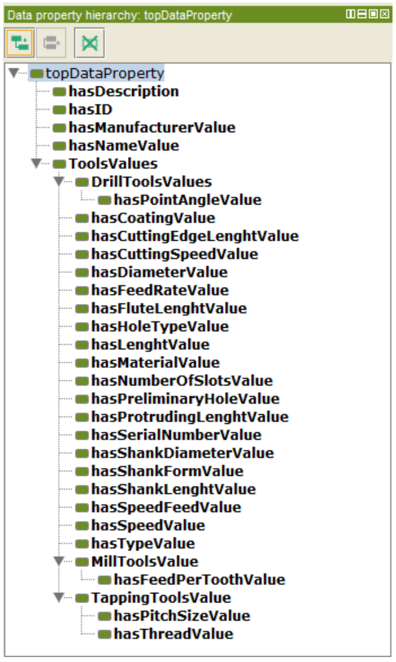


Fig. 2: Relation of individuals

The third object property group of the ontology is *consistOf* and contains the sub properties *consistOfMachine*, *consistOfSystem*, *consistOf*Area, *consistOf*Site which are used alongside the defined classes *System_Creation*, *Area_Creation*, *Site_Creation* as covering axioms for the creation of different systems and to illustrate the relation between the different classes. These properties are used in a way similar to the first and second group of properties so they won't be discussed any further.

B. Data properties

The data properties are used to describe the characteristics of the different individuals. There are 32 data properties in the ontology. As you can see from Fig. 3 the data properties are divided in some categories. The major one is *ToolsValues* which are used to describe the different properties of drills, mills, tappers and reamers and assign them values. Some of the properties are additionally divided into groups because they are specifically reserved for only one type of tools. For example, in the *TappingToolsValue* group, the data properties are *hasPitchSizeValue* and *hasThreadValue* which are specific for the tapping tools. Some of the properties do not have groups and that is because they are universal for the ontology and each individual must have them in order to be described properly.


Data property hierarchy: topDataProperty

```
topDataProperty
    hasDescription
    hasID
    hasManufacturerValue
    hasNameValue
    ToolsValues
        DrillToolsValues
            hasPointAngleValue
        hasCoatingValue
        hasCuttingEdgeLenghtValue
        hasCuttingSpeedValue
        hasDiameterValue
        hasFeedRateValue
        hasFluteLenghtValue
        hasHoleTypeValue
        hasLenghtValue
        hasMaterialValue
        hasNumberOfSlotsValue
        hasPreliminaryHoleValue
        hasProtrudingLenghtValue
        hasSerialNumberValue
        hasShankDiameterValue
        hasShankFormValue
        hasShankLenghtValue
        hasSpeedFeedValue
        hasSpeedValue
        hasTypeValue
        MillToolsValue
            hasFeedPerToothValue
        TappingToolsValue
            hasPitchSizeValue
            hasThreadValue
```

Fig. 3: Data properties

3.3. Restrictions

In this ontology the main types of restrictions that are used are *Quantifier restrictions* or more specifically *Existential restrictions* and *hasValue restrictions*. The *Existential restrictions* are used in the *Tools* section of the ontology so that the individuals could be allocated according different criteria. For example, there are two defined sub classes – *MillingTools_Carbide* and *MillingTools_NumberOfSlots_Little* – of the *MillingTool* class. The *MillingTools_Carbide* class has the *Existential restriction* `MillingTools and (hasMaterialValue some xsd:string[pattern "Carbide"])` and the other class - *MillingTools_NumberOfSlots_Little* - has *Existential restriction* `MillingTools and ((hasNumberOfSlotsValue some xsd:integer[>=3]) and (hasNumberOfSlotsValue some xsd:integer[<=3])) and (hasMaterialValue some xsd:string[pattern "Carbide"])`. When the ontology is inferred the *MillingTools_Carbide* class is collecting all of the mill tools that are made of carbide and the second defined class *MillingTools_NumberOfSlots_Little* is collecting all mill tools that are made of carbide and have three slots. Because of the additional restriction to the second class, it is inferred as a sub class of the defined class *MillingTools_Carbide* although that these two classes are created on the same hierarchical level. In this way the sub class is going to collect two of the three mills because the third one has four slots and does not meet the specific restrictions of the *MillingTools_NumberOfSlots_Little* class. So the third mill will be inferred in the class *MillingTools_Carbide*. Another example is made by adding the *Existential restriction* `Tools and (hasMaterialValue some xsd:string[pattern "Carbide"])` to the class *AllCarbideTools* which is sub class of the class *Tool*. The class is collecting all of the tools in the ontology that are made of carbide. So having in mind the previous example, when the ontology is inferred the group of the mill classes is allocated as sub class of the *AllCarbideTools* class.

4. Conclusions

The developed domain ontology in this study aims to represent the equipment in the manufacturing systems and the relations between the different equipment levels. The ontology, merged along with three others in meta ontology is going to be used by specially developed software for the generation of control programs thus illustrating an approach for solving the interoperability issue in the field of reconfigurable manufacturing systems. Some of the main benefits of the developed ontology are:

- the hierarchy is based on the IEC/ISO 62264 standard which defines the basic structure of the ontology;
- through the use of object properties, it is possible to create sensors, modules and machines (collections of different individuals);
- through the use of object properties and covering axioms, it is possible to create systems, areas and sites (collections of different defined and named classes);
- through the use of Quantifier restrictions and hasValue restrictions, it is possible to create sorting operations for the different individuals based on data properties.

Acknowledgments

The creation of this piece of work was possible thanks to the Ph.D. project "Developing approach for interoperability for reconfigurable manufacturing systems" - contract № 152 ПД 0015-05.

Literature

[1]. Chen D., Dassisti M., Elvesaeter B., "Enterprise Interoperability Framework and knowledge corpus", 2007.

[2]. IEC61499, International Standard IEC61499, Function Blocks, Part 1 - Part 4, International Electrotechnical Commission (IEC), Technical Committee TC65/WG6, IEC Press, January, 2005.

[3]. IEC 62264-1:2003, Enterprise-control system integration -- Part 1: Models and terminology.

[4]. Pratt M. J., „Introduction to ISO10303 – the STEP Standard for Product data exchange", Journal of Computing and Information Science in Engineering 1(1), pp.102-103, 2001.

[5]. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) W3C Recommendation 11 December 2012, https://www.w3.org/TR/owl2-syntax/.

[6]. Krötzsch M., "OWL 2 Profiles: An Introduction to Lightweight Ontology Languages," Proc. 8th Reasoning Web Summer School, LNCS 7487, Springer, 2012, pp. 112–183.

[7]. Krötzsch M., Simančík, F., Horrocks, I., Description Logics. In IEEE Intelligent Systems, volume 29:1, pp. 12–19. IEEE 2014.

[8]. Glimm B., Horrocks I., Motik B, Stoilos G, Wang, Z., HermiT: An OWL 2 Reasoner, Journal of Automated Reasoning, October 2014, Volume 53, Issue 3, pp 245-269.