Creating Resource Combinations Based on Formally Described Hardware Interfaces

Niko Siltala, Eeva Järvenpää, Minna Lanz

Creating Resource Combinations Based on Formally Described Hardware Interfaces

Niko Siltala*[0000−0001−6456−1251], Eeva Järvenpää[0000−0001−6513−135X], and Minna Lanz[0000−0003−2182−4669]

Tampere University of Technology, Laboratory of Mechanical Engineering and Industrial Systems, Tampere, Finland
[niko.siltala, eeva.jarvenpaa, minna.lanz]@tut.fi

Abstract. This paper introduces the Resource Interface ontology intended to formally capture hardware interface information of production resources. It also proposes an interface matchmaking method, which uses this information to judge if two resources can be physically connected with each other. The matchmaking method works on two levels of detail, coarse and fine. The proposed Resource Interface ontology and matchmaking method can be utilised during production system design or reconfiguration by system integrators or end users. They will benefit from fast and automatic resource searches over large resource catalogues. In the end of the paper, a validation of the method is provided with a test ontology.

Keywords: Interface, Resource description, Production system reconfiguration, System design

1 Introduction

Responsiveness of manufacturing is an important strategic goal for manufacturing companies operating in a highly dynamic environment characterised by constant change. Such responsiveness and adaptivity is related to the need to reconfigure and adjust the production and corresponding production system as efficiently as possible to the required changes in processing functions, production capacity, and the dispatching of the orders. [1, 2] To do this, the production system needs an inherent ability to facilitate continual and timely change in its structure and in its functional operations. Structure refers to the way in which the functional building blocks of a production system are assembled to form a holistic, interoperable system, while the term function describes the abilities of the building blocks or the production system as a whole to realise a defined purpose. [3] During system design and re-configuration, new structural configurations are built to fulfil the functional requirements set by the product. In order to achieve a feasible structural configuration, the combined resources must have compatible interfaces.

Traditionally, the system design and reconfiguration has been purely a human-driven and time consuming process, relying on the expertise and tacit knowledge of the system integrators and the end users of the system. The realisation of the

requirements for fast response calls for new methods and solutions that would drastically reduce the time and effort put into planning and implementing the alterations in a factory, such as plug and play interfaces, modern information and communication technologies, simulations and new planning methods [4]. Within the past decade, there have been multiple different projects and research trying to provide computerised support for this reconfiguration planning process. Important steps towards modular assembly equipment and standardised hardware and control interfaces was made by, for example, the EU-funded project called EUPASS [5]. According to [6], the modular architecture paradigm for new production systems, which focuses on the clear functional decoupling of equipment module functionalities and the use of standardised interfaces to promote interchangeability, presents the possibility for developing automated reconfiguration methods. The currently running project ReCaM [7] aims to develop a set of integrated tools for rapid and autonomous reconfiguration of production systems. The approach relies on a unified functional description of resources, providing a foundation for rapid creation of new system configurations through capability-based matchmaking of product requirements and resource offerings.

The aim of bringing automation to the system design, re-configuration, and order dispatching requires a formal, structured representation of the product requirements as well as resource's capabilities, properties, and interfaces. For the past two decades, there has been an increasing interest in manufacturing domain on using emerging technologies such as ontologies, semantics and semantic web, to support the collaboration, interoperability and adaptation needs [8–11]. The detailed formal interface descriptions have been out of the scope of these works.

In our previous research, we have studied capability matchmaking and creation of new resource combinations, which have combined capabilities to satisfy the processing requirements of the specific product [12]. The Web Ontology Language (OWL)-based Capability Model allows such combinations to be created from the capability perspective [13]. In order to make sure that the resources can be physically connected, a formal resource interface description and associated matchmaking is also required.

The objective of this paper is to present recent development of a Resource Interface ontology that can be used for finding production resources, which can be connected together. The Resource Interface ontology shall provide a response to the following use cases: Find resources that can be connected with particular interface of a resource; Find resource combinations with connectable interfaces; and Ensure that suggested resource combinations found by an external system (e.g. capability matchmaking [12] or system designer propositions) are physically connectable. The work builds upon our existing work on formalised interface description as a part of XML-based Resource Description concept [14, 15].

The paper is organised as follows. The second chapter represents the Resource Interface ontology and its details. Next chapter illustrates the interface matchmaking first on coarse level, followed by fine level matchmaking. One example SPARQL query is shown. The fourth chapter shows first evaluation

results through a few test cases with our test ontology. Finally, the paper ends with discussion and conclusion.

2 Interface Description

2.1 Interface description in Resource Description concept

The Resource Description (RD) concept formalises characteristics of production resources by creating a comprehensive description of resource's features. This description, the RD [14, 15], is then published and used to exchange information between resource provider and various systems used for production system design and execution.

One section of the RD model formalises the information defined by electro-mechanical interface standards. The interface information captured by a RD are: name, identifier (ID), category, gender, and force and torque limits of the interface; additional properties characterising the interface; and implementations of it. Each interface implementation provides its physical pose (frame origin in 3D space) and the kinematic model, if the interface is a movable one. Additionally, the RD defines the organisation that defines the interface standard, and what is the general purpose of the interface. The latter is done through link with the abstract interface model, which is a shared contract between similar kind of resources.

Only a subset of information required for the interface matchmaking is retrieved from a RD, and read into the Resource Interface ontology. This is done to collect together information from several resources and to utilise the search methods provided by ontologies.

2.2 Resource Interface Ontology

OWL is used to codify the Resource Interface ontology and for storing the actual resource instances and their interface information instances. Figure 1 shows the structural model of the Resource Interface ontology. Next, the key components of this model are described.

Class ***InterfaceDefinition*** is the main entry point for the Interface Model ontology. It defines an implementation of the hardware interface and links it to an interface standard (***IfStandard*** with *implementsStd* object property), purpose of the interface port (***InterfacePurpose*** with *hasPurpose*), a set of characterising information (***IfCharacteristic*** with *hasIfCharacteristic*), set of physical interface port implementations (***IfPortImplementation*** with *hasIfPortImplementation*), and ***ForceAndTorqueLimit*** (with *hasForceAndTorqueLimit*), which defines the maximum forces the interface can handle at different directions. In addition, the ***InterfaceDefinition*** defines the gender and category of the interface implementation. Optionally it can define human readable description, and occurrence minimum and maximum, which define a range of how many occurrences of this interface implementation can exist within this resource.

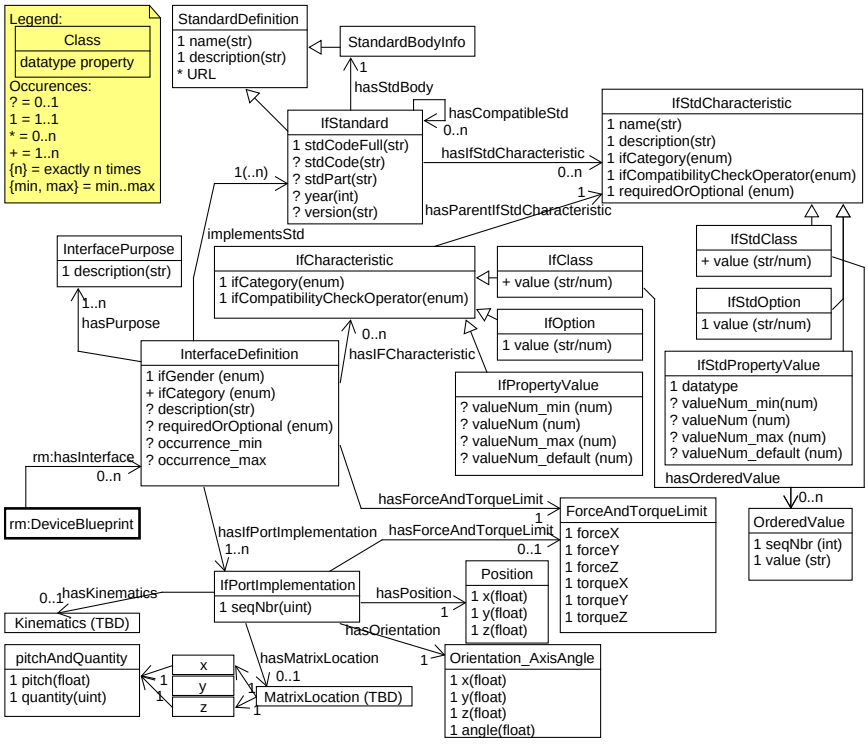

Fig. 1. Resource Interface ontology

First, we focus on the interface standard, because it is the primary linking factor when judging, if two interface implementations, present in two resources, can be connected together. The ***IfStandard***'s property *stdCodeFull* is used as ID and primary key connecting two implementations together. Human readable name and description of each interface standard must be provided. Furthermore, the ***IfStandard*** can have the designation represented as elementary items. These can be stored into the corresponding properties *stdCode*, *stdPart*, and *year*. The ***IfStandard*** links to ***StandardBodyInfo***, which groups the interface standards under a standardisation body, and to ***IfStdCharacteristic***, which provides characteristics for this interface standard. Additionally, the ***IfStandard*** has object property *hasCompatibleStd*, which states that another interface standard can be used as compatible replacement of the other. This link is only unidirectional, and all compatibles must be explicitly stated.

The gender (*ifGender* data property) of ***InterfaceDefinition*** is the second linking factor and it can have one of three enumerated values - male, female, or neutral. This defines polarity of the interface, and which implementations of the

same ***IfStandard*** can be connected together. Rules are simple - male and female or two neutrals can be connected together. Examples of setting up a gender are such as a plug is male, a socket is female and a plain flange with mounting through holes could form a neutral gender.

The *ifCategory* data property defines the category of the interface. It has six available enumerations - Mechanical, Electrical, Service, Communication, Other, and N/A (Not Available). Category is used to classify the role and purpose of the interface. Similarly *ifCategory* is used to categorise interface characteristics.

Interface characteristics formalise designations, variants, or configuration options defined in the interface standards. Examples of these are size, pneumatic supply configuration, fieldbus used, and accuracy. These characteristics are presented in two places - ***IfStdCharacteristic*** and ***IfCharacteristic***. The difference between these two is that the former defines all values for the characteristic, which are directly coming from the standard, and the latter provide the value(s), which is(are) applicable for the interface implementation. The latter is also linked with the former through *hasParentIfStdCharacteristic* object property, that enables the name and description be given only in one place at ***IfStdCharacteristic***. Both types of characteristics have three similar sorts of implementations - class, option, and property value. The interface class is used in cases where characteristic defines a finite set of predefined values, of which usually one is present in the implementation. Values can also be an ordered set, if they can be compared and set to growing order. Interface option is used in cases when there is an option, which is, or is not, present at the implementation of the interface. Only one value is bound to the option. The third is property value that is used to represent a range or a real number characterising the interface and its usage. Even the resource would have several instances of the same interface standard, but they are not sharing the same set of ***IfCharacteristic***, a new ***InterfaceDefinition*** must be defined.

Every physical connection port (or interface) in the resource has its own ***IfPortImplementation***. This defines every physical implementation of the ***InterfaceDefinition***, having representation for the position and orientation of the interface in the spatial space. Additionally, it can define kinematics of the interface; matrix locations, which represent repetition of the interface such as a matrix of a threaded holes; and finally, it can refine more stricter force limits.

InterfacePurpose is used to group different interfaces and interface implementations by the overall purpose of the interface. There is a predefined set of general purposes, which are linked to interface definitions. This allows user to search for detailed interface implementations across different resources and interface standards fulfilling a specific intended purpose such as main mounting point, material transfer in, or tool interface.

Finally, a resource needs to state that is has an interface or interfaces. For example, a class ***DeviceBlueprint*** from the Resource Model ontology can do this by assigning *hasInterface* object property linking it to an instance of ***InterfaceDefinition***.

3 Interface Matchmaking

The interface matchmaking means in this case a process for finding out connectable and compatible production resources from the hardware interface point of view. This can be illustrated with a few use case scenarios, which utilise the Resource Interface ontology: 1) To find all resources, which can be connected with the selected resource; 2) To find all resources, which can be connected with one particular interface of the selected resource, instead of all interfaces; 3) To find all possible resource combinations, which are connectable together. This is generalised case of the first; and 4) To analyse if the connections in the proposed system layout are connectable also from the interface point of view. This has the similarities with the second case. In all previous four cases the matching can be done at two different levels of detail. The first, more coarse level (a), is to analyse only interface ID and the gender information. The second level (b) is finer and uses also the ***IfCharacteristic*** information and associated rules (*ifCompatibilityCheckOperator*). Section 3.1 focuses on the coarse level and Section 3.2 on the fine level.

3.1 Process for Comparing the Interfaces

SPARQL Protocol and RDF Query Language (SPARQL) is used to make queries to the Resource Interface ontology. At the moment these queries are executed manually with an ontology editor such as Protégé, but in the future a Software (SW) Application Program Interface (API) will be developed to run and process them in sequences, and to provide additional filtering of the results.

The following SPARQL query (Listing 1.1) is used as an example. It finds resources, which are connectable with resource X's interface Y, and query result is a listing of records of matching resources. Matching is done in this case at the coarse level i.e. using only the interface ID and gender information. Only values of X (on line 2) and Y (on line 3) are changed, when matching with another resource or interface, and the rest of the query remains the same. In the validation case, values could be: X = '*Manipulator_2-axis*' and Y = '*ISO 29262*'. Similar SPARQL queries are prepared for finding answers for other scenarios, mentioned in the beginning of the chapter.

```
1 SELECT ?fromMO ?fromIF ?fromGender ?toGender ?toIF ?toMO ?
      fromIFStdCode WHERE {
2 ?fromMO rdfs:label "X" .
3 FILTER regex(?fromIFStdCode, "Y", "") .
4 ?fromMO a rim:TempDeviceBluePrint ;
5   rim:hasInterface ?fromIF .
6 ?fromIF rim:implementsStd ?fromIFStd ;
7   rim:ifGender ?fromGender .
8 ?fromIFStd rim:stdCodeFull ?fromIFStdCode .
9 ?toIFStd rim:stdCodeFull ?fromIFStdCode .
10 ?toIF rim:implementsStd ?toIFStd .
11 bind(xsd:string(if(?fromGender="NEUTRAL","NEUTRAL", xsd:string
      (if(?fromGender="MALE","FEMALE","MALE")))) as ?toGender) .
```

```
12 ?toIF rim:ifGender ?toGender .
13 ?toMO rim:hasInterface ?toIF .
14 FILTER (?fromMO != ?toMO) . }
```

Listing 1.1. SPARQL example finding interface matches with resource X's interface Y

Listing 1.1 works as following: Line 1 starts the SPARQL query and defines what information is shown on the resulting records. Line 2 selects the named resource as a focus module (from). Line 3 uses regular expressions to look for the specific interface standard. Lines 4..8 find and select the resource(s), which are used as focus modules for the matching. Also its interface is found out. Lines 9..10 look for counter part resources implementing the same interface standard. Line 11 defines which gender is accepted for the counterpart, and lines 12..13 find and select such resources as targets (to). Finally, line 14 filters out the records connecting the focus module to itself.

3.2 Rules for Interface Characteristics

The finer level interface matching needs further information from the interface implementation, and the choices made by the resource provider. The concept of interface characteristic provides this additional information. It provides not only the IDs and values of the characteristic, but also a compatibility operator that defines how these values must relate against each other in case of a positive match. The two resources are connectable at finer level, if and only if all (mandatory) ***IfCharacteristic***s of an interface provide a positive match.

Each of the ***IfCharacteristic*** has one *ifCompatibilityCheckOperator*. This operator specifies how the values from the source resource are compared with the target resource. Table 1 defines these twelve different interface compatibility operators, and for which kind of ***IfCharacteristic*** types a compatibility operator can be assigned to. A mathematical formulation of the compatibility operator is represented at the end of the description.

The three ***IfCharacteristic*** types are source set, ordered set, and value. In case of Source set (S), values of the ***IfCharacteristic*** at source resource's side are compared with Target set (T), i.e. remote resource's corresponding values, according to the corresponding compatibility operator.

In case of 'Ordered set' type of ***IfCharacteristic***s (column Type has O in Table 1) it applies: There exist an ordered set (OS), where $OS = \{a_1, \ldots, a_i, a_j, \ldots, a_n\} \wedge a_i < a_j$. This set is actually defined by the interface standard and stored in ***IfStdCharacteristic***. For target set (T) applies $\forall y : y \in T \wedge T \subset OS \wedge y \in OS$.

In case of 'Value' type of ***IfCharacteristic***s (column Type has V in Table 1) it applies: The target set (T) is defined such as $T = [t_{min}, t_{max}]$. Source set (S) is a range ($[a, b]$) or a single value ($a = b$).

4 Test Cases for Interface Matchmaking

The developed Resource Interface ontology is validated with a test ontology containing some representative resources. Table 2 shows these resources, which

Table 1. Compatibility operators with descriptions and which interface characteristic types the operator is applicable for

Type	Compatibility Operator	Description
S,O,V	SAME_SET	The source and the target contain exactly the same set of values (or a value or a range). $\forall x : x \in S \wedge x \in T \wedge S = T$.
S,O	ALL_FROM_SET	All items in the source set can be found from the target set. $\forall x : x \in S \ \wedge \ S \subseteq T$.
S,O	ANY_FROM_SET	Any item(s) of the source set can be found from the target set. $\exists x : x \in S \ \wedge \ x \in T$.
S,O	ONE_FROM_SET	Exactly one item from the source set can be found from the target set. Size of target set is one. $\exists x : x \in S \wedge x = T \ \wedge \ T \subset S$.
O	LOWER_OR_EQUAL	Standard defines an ordered set of values. Source value is lower or equal than the target value. $\forall x : x \in S \wedge S \subset OS \ \wedge \ x \in OS \ \wedge \ x \leq y$.
O	HIGHER_OR_EQUAL	Standard defines an ordered set of values. Source value is higher or equal than the target value. $\forall x : x \in S \ \wedge \ S \subset OS \ \wedge \ x \in OS \ \wedge \ x \geq y$.
V	INSIDE_RANGE	Source value or range is inside the target range. $S = [a, b] \wedge a \leq b \wedge a \geq t_{min} \wedge b \leq t_{max}$.
V	PART_OF_RANGE	There is an overlap between the source range (or value) and the target range (or value). $S = [a, b] \wedge a \leq b \wedge a \leq t_{max} \ \wedge \ b \geq t_{min}$.
V	PART_OF_RANGE_OR_LOWER	The source range (or value) is lower than the target range (or value) or there is an overlap. $S = [a, b] \wedge \ a \leq b \ \wedge \ b \leq t_{max}$.
V	PART_OF_RANGE_OR_HIGHER	The source range (or value) is higher than the target range (or value) or there is an overlap. $S = [a, b] \wedge \ a \leq b \ \wedge \ a \geq t_{min}$.
V	CAPTURES_RANGE	The source range (or value) captures completely the target range (or value). $S = [a, b] \ \wedge \ a \leq b \ \wedge \ a \leq t_{min} \ \wedge \ b \geq t_{max}$.
-	NO_RULE	No compatibility operator is defined for ***IfCharacteristic***.

Legend: Type of ***IfCharacteristic*** values { S=Set, O=Ordered Set, V=Value }

interface standard they implement, and at which gender. If the interface standard has some characteristics, these are illustrated with applied characteristic values and compatibility operator. These are discussed in the following.

Looking for a pair for a specific resource – coarse. Scenario 1.a Various resources from Table 2 are used as parent resource, and in all cases, the exactly expected list of counter resources is found. I.e. 'Manipulator_2-axis' connects with all

Table 2. Resources and their interfaces in the test ontology

Resource	Interface standard	Gender	Characteristics	Comp. Operator
basePlate	PROP.BasePlate. Grooved	N	-	-
Manipulator_2-axis	ISO 29262:2011	F	size = 20 service = PP service = PV	SAME_SET ANY_FROM_SET ANY_FROM_SET
	PROP.RobotBase	N	-	-
FingerGripper1	ISO 29262:2011	M	size = 20 service = PV	SAME_SET ANY_FROM_SET
	PROP.GripperTCP	F	-	-
FingerGripper2	ISO 29262:2011	M	size = 25 service = PV	SAME_SET ANY_FROM_SET
	PROP.GripperTCP	F	-	-
	SHAPE.CYLINDER	F	diameter = [5.0, 20.0] Holding length = [5.0, 40.0]	CAPTURES_RANGE PART_OF_RANGE
FingerGripper3	ISO 29262:2011	M	size = 25 service = PV	SAME_SET ANY_FROM_SET
	PROP.GripperTCP	F	-	-
BoringChuck	SHAPE.CYLINDER	F	diameter = [3.0, 16.0] holding length = [20.0, 80.0]	CAPTURES_RANGE PART_OF_RANGE
DrillBit_1mm	SHAPE.CYLINDER	M	diameter = 1.0 holding length = [10.0, 30.0]	INSIDE_RANGE PART_OF_RANGE
DrillBit_12mm	SHAPE.CYLINDER	M	diameter = 12.0 holding length = [30.0, 70.0]	INSIDE_RANGE PART_OF_RANGE

Legend: Gender { N=Neutral, M=Male, F=Female }
service values: { PP=Pressure-Pressure, PV=Pressure-Vacuum }

grippers, 'BoringChuck' with all drills, and a drill bit with 'BoringChuck' and 'FingerGripper2'.

Looking for a pair for a specific resource and its particular interface – coarse. Scenario 2.a Listing 1.1 is used as SPARQL query. This works as the previous one, but focuses only on one interface and provides only those matches as results.

Looking for a pair for a specific resource and its particular interface. Taking into account interface properties – fine. Scenario 2.b Some tested queries are 'DrillBit_1mm' & 'CYLINDER'; 'DrillBit_12mm' & 'CYL'; and 'Manipulator_2-axis' & 'ISO 29262'. The first query does not have any matches – as expected,

because the drill's diameter is so small. The second found both 'BoringChuck' and 'FingerGripper2', with a total of four records, as all given drill bit characteristics match in this case. The result is also as expected. The third provides as a result four records – all three grippers in case of ***IfCharacteristic*** = 'service', but only 'FingerGripper1' in case of ***IfCharacteristic*** = 'size'. Thus, some additional filtering of results is needed to judge that from these only 'FingerGripper1' can be connected from the interface matching point of view. This is because it is the only one conforming to all the characteristics requirements from the manipulator side. Therefore, this use case requires additional SW application to execute several SPARQL queries and filter out the final result of the interface match.

Looking for pairing resources from a larger set of resource – coarse. Scenario 3.a Scenario works and provides results. However, the result has every match listed twice, having each resource presented both as focus and target module. Further filtering of the result is needed.

5 Discussion and Conclusions

This paper represents the developed Resource Interface ontology and how it can be utilised for hardware interface matchmaking. In case of the interface matchmaking, the paper limits only to the proof of concept. Four test cases were defined and corresponding SPARQL queries were developed. These queries were run in our small test ontology, and they were able to sort out from the ontology the results indicating which resources are connectable together. However, the results show that further filtering and processing of the results is needed. As future work, the authors will prepare an API for getting the parametric inputs for the queries, running multiple queries in sequence, and further filtering and purifying the results.

The Resource Interface ontology will be complementing our Resource Description concept by collecting together a snippet of information from several RDs, and offering more powerful search abilities over the information. Especially important are the links where information from different resources comes together. RD remains as comprehensive information container focusing a single resource.

The proposed concept was successfully applied and it found out resources with the matching interfaces. This makes it possible to search easily, fast, and automatically for resources, which can be connected physically, from large resource catalogues. This will support system integrators and end users during system design and reconfiguration, to find out faster the working system configurations.

Acknowledgements

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 680759 [7].

References

1. Koren, Y., Shpitalni, M.: Design of reconfigurable manufacturing systems. J. Manuf. Syst. 29(4), 130–141 (2010).
2. Wiendahl, H.-P., ElMaraghy, H.A., Nyhuis, P., Zäh, M.F., Wiendahl, H.-H., Duffie, N., Brieke, M.: Changeable Manufacturing - Classification, Design and Operation. CIRP Ann. 56, 783–809 (2007).
3. Rahimifard, A., Weston, R.H.: A resource-based modelling approach to support responsive manufacturing systems. Int. J. Adv. Manuf. Technol. 45, 1197–1214 (2009).
4. Westkämper, E.: Factory Transformability: Adapting the Structures of Manufacturing. In: Reconfigurable Manufacturing Systems and Transformable Factories. pp. 371–381. Springer Berlin Heidelberg, Berlin, Heidelberg (2006).
5. EUPASS – Evolvable Ultra-Precision Assembly SystemS - project. EU FP6. GA No 507978. Accessed 30.10.2017, http://cordis.europa.eu/project/rcn/75342_en.html, (2006).
6. Ferreira, P., Lohse, N., Ratchev, S.: Multi-agent Architecture for Reconfiguration of Precision Modular Assembly Systems. In: Ratchev, S. (ed.) Precision Assembly Technologies and Systems. pp. 247–254. Springer (2010).
7. ReCaM – Rapid reconfiguration of flexible production systems through capability-based adaptation, auto-configuration, integrated tools for production planning - project. EU Horizon 2020. GA No 680759. Accessed 30.10.2017, http://www.recam-project.eu/, (2017).
8. Ameri, F., Urbanovsky, C., and McArthur, C.: A Systematic Approach to Developing Ontologies for Manufacturing Service Modeling. Proceedings of the Workshop on Ontology and Semantic Web for Manufacturing 2012, 14 p (2012).
9. Borgo, S., Leitão, P.:Foundations for a core ontology of manufacturing. Integrated Series in Information Systems, vol 14, (2007).
10. Strzelczak, S.: Towards Ontology-Aided Manufacturing and Supply Chain Management A Literature Review. In: Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth, pp. 467–475 (2015).
11. Terkaj, W., Urgo, M.: Virtual Factory Data Model to support Performance Evaluation of Production Systems. CEUR Workshop Proceedings, Vol. 886, pp. 44–58 (2012).
12. Järvenpää, E., Siltala, N., Hylli, O., Lanz, M.: Capability Matchmaking Procedure to Support Rapid Configuration and Re-configuration of Production Systems. Procedia Manuf. 11, 1053–1060 (2017).
13. Järvenpää, E., Siltala, N., Lanz, M.: Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems. In: 2016 IEEE International Symposium on Assembly and Manufacturing (ISAM). pp. 120–125. IEEE (2016).
14. Siltala, N.: Formal Digital Description of Production Equipment Modules for supporting System Design and Deployment, PhD dissertation, Tampere University of Technology, http://urn.fi/URN:ISBN:978-952-15-3783-7, (2016).
15. Siltala, N., Järvenpää, E., Lanz, M.: Formal Information Model for Representing Production Resources. In: Nääs I. et al. (eds.) Advances in Production Management Systems. Initiatives for a Sustainable World. APMS 2016. IFIP Advances in Information and Communication Technology, vol 488. pp. 53–60. Springer, Cham, Iguassu Falls, Brazil (2016).