Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 72 (2018) 1094–1099

www.elsevier.com/locate/procedia

51st CIRP Conference on Manufacturing Systems

Product Model ontology and its use in capability-based matchmaking

Eeva Järvenpää*[a], Niko Siltala[a], Otto Hylli[b], Minna Lanz[a]

[a]Laboratory of Mechanical Engineering and Industrial Systems, Tampere University of Technology, Korkeakoulunkatu 6, 33720 Tampere, Finland
[b]Laboratory of Pervasive Computing, Tampere University of Technology, Korkeakoulunkatu 6, 33720 Tampere, Finland

* Corresponding author. Tel.:+358-40-849-0869. *E-mail address:* eeva.jarvenpaa@tut.fi

Abstract

Capability-based matchmaking aims to support rapid design and reconfiguration of modular plug-and-produce type production systems. It relies on formal ontological descriptions of product requirements and resource capabilities. This paper introduces the structure and content of the developed Product Model ontology, and explains its role as a part of the capability matchmaking procedure. A case product is modelled in order to visualize a matchmaking scenario. We expect that such matchmaking will reduce the workload of system designers and reconfiguration planners as it can automatically suggest potential resources for a certain need from large resource catalogues.

Keywords: Product Model; Ontology; Information Model, Capability-based matchmaking; Production system design; Reconfiguration

1. Introduction

Rapid responsiveness and reconfigurability of production systems are important strategic goals for manufacturing companies operating in a highly dynamic environment. These requirements call for new methods and tools, which can reduce the time and effort put to the system design, both in brownfield and greenfield scenarios. Currently, the system design and reconfiguration planning are manual processes, which rely heavily on the designers' expertise to find feasible solutions by comparing the characteristics of the product to the technical properties of the available resources. This slow process sets limitations to the amount of potential configuration alternatives that can be considered. The aim of bringing automation to this design process requires a formal, structured representation of the product requirements as well as resource capabilities, properties and constraints.

For the past two decades, there has been an increasing interest on using ontologies and Semantic Web technologies in the manufacturing domain. In the context of distributed intelligent systems, ontologies play a key role as they provide a shared, machine-understandable vocabulary for information exchange among dispersed actors [1,2]. The currently running project ReCaM aims to develop a set of integrated tools for rapid and autonomous reconfiguration of production systems through capability-based matchmaking of product requirements and resource offerings. In ReCaM, the authors have developed a Manufacturing Resource Capability Ontology (MaRCO), which is an OWL-based information model for describing the capabilities of manufacturing resources [3]. However, also a formal way to describe the product requirements is needed.

Many researchers have utilized skill-based approach to describe the products processing requirements and functionalities provided by the resources, and to facilitate autonomous setup and execution of production tasks (e.g. [4,5]). Bengel [6] presented an ontological workpiece-centered approach for model-based configuration, focusing on the representation of the skills the product requires. The Manufacturing-as-a-Service paradigm has been adopted by many researchers developing different approaches to formally describe service requests and offerings [2,7,8]. Ontology-based

Manufacturing Service Description Language (MSDL) was used in a matchmaking methodology, which aims to connect buyers and sellers of manufacturing services in distributed digital manufacturing environments [9].

These approaches use the same skill model to represent both the products and the resources. This facilitates easy matchmaking between requirements and offerings. However, we think it would be better to clearly separate these two aspects. This is because the parameters used to represent the requirements may be different than those used to represent the skills, e.g. the requirement may present an exact value for a certain parameter, while the skill may provide a range. In addition, the resource representation often contains additional parameters, e.g. related to the performance of the resource, which are not relevant from the product's perspective. Furthermore, it is desired that the designer doesn't need to specify the exact process while describing the product requirements, if that is not crucial.

In this paper, we will introduce the Product Model ontology, which was developed to provide the product related input information for the capability-based matchmaking (described in [10]). In section 2 we will first introduce the other model used during the matchmaking, namely the MaRCO model. It laid the foundation for the development of the Product Model, which we introduce in section 3. In section 4 we explain the principles of the capability matchmaking, and in section 5 we give a practical example of the matchmaking. Section 6 concludes the paper.

2. Manufacturing Resource Capability Ontology Model

Manufacturing Resource Capability Ontology (MaRCO) is an OWL-based information model that can be used to describe capabilities, i.e. functionalities, of resources and resource combinations. Fig. 1 presents the main classes and relations of the MaRCO model. Detailed information about MaRCO can be found from our earlier publications [1,3].

MaRCO imports another ontology called Process Taxonomy Model. It categorizes different manufacturing and assembly processes in a hierarchical structure, as illustrated in Fig. 2. It is a pure taxonomy, which doesn't contain any properties or instances. It is adapted from multiple existing taxonomies and process categories, including the German standard DIN 8580, the EUPASS processes [11] and the production taxonomy used in the CO2PE!-initiative [12]. The capability classes are linked to the *ProcessTaxonomyElement* classes depending on what kind of process they can provide. This reference is implemented as a direct is-a (sub-class) relationship between the *Capability* class and *ProcessTaxonomyElement* Class.

MaRCO model defines relations between simple (atomic) and combined capabilities. For instance, robot has a simple capability "Moving" and gripper has a simple capability "Grasping". Together they have a combined capability "Transporting". Based on these relations, the potential device combinations that have a certain combined capability can be identified programmatically by utilizing information provided by SPARQL queries. SPARQL is a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format, including OWL-ontologies [13]. We use SPIN (SPARQL Inferencing Notation) rules to automatically infer the parameters of the combined capabilities [14]. SPIN is a W3C Member Submission that has become the de-facto industry standard to represent SPARQL rules and constraints on Semantic Web models [15].

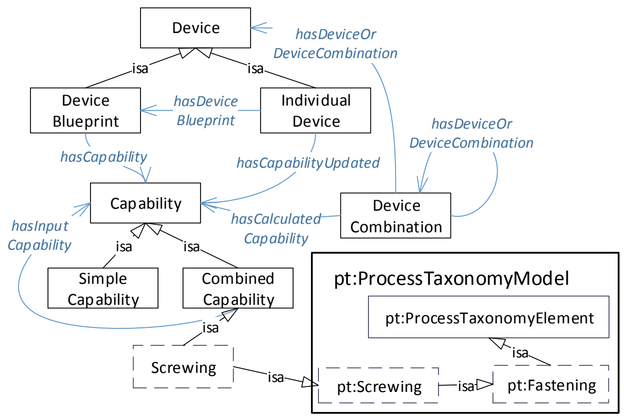

Fig. 1. Main elements of the MaRCO model.

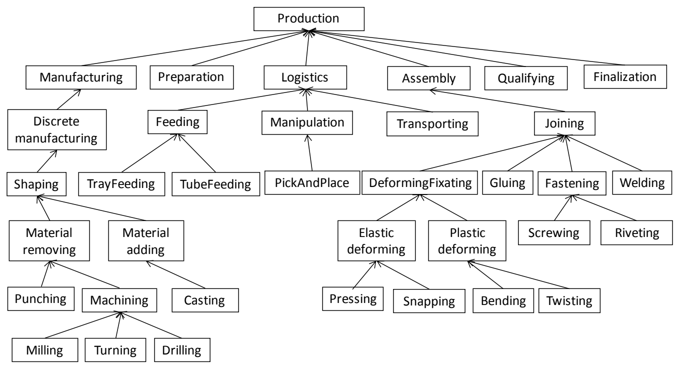

Fig. 2. Example and partial view of the process taxonomy.

3. Product Model Ontology

We followed the Ontology engineering methodology [16] during the development of the Product Model ontology. This section describes the results of the kickoff and refinement phases. In kickoff, the requirements for the ontology were detailed and important concepts and their relations were analysed. In the refinement phase, the ontology was formalized and represented in OWL-language.

3.1. Requirements definition

The main purpose of the Product Model ontology is to represent the processing requirements of the product in a manner that these requirements can be matched against the resource capabilities. As the MaRCO and Process Taxonomy models already exist, the Product Model should be designed to be compatible with these models. The model should provide flexibility to the product designers or process planners to describe the requirements on different levels of detail. This means that they should not be forced to specify the exact processing method, if it is not crucial from the product perspective. E.g. the designer may want to specify that some sort of riveting process is needed to join two parts together, but the selection about the actual riveting method (e.g. impact, radial, orbital,…) may be left after the matchmaking depending on the found results

The capability matchmaking process sets the requirements for the model's information content. From the perspective of the product designer or process planner, the definition of the product requirements should be as easy as possible. This implies that it should contain only that kind of information that is generally collected, or defined, during process planning. To be able to perform the matchmaking, and to design the production system configuration, some basic characteristics of each part has to be available, such as the shape, dimensions, mass, material, specific geometric features and tolerances. The model needs to describe the structure of the product, including the sub-assemblies and their contained parts and subassemblies. For each part and subassembly, the related manufacturing, assembly or auxiliary processes should be modelled. Also the process sequence and parametric requirements related to these processes need to be included.

3.2. Refinement

In the refinement phase we codified the ontology based on the requirements defined in the kickoff phase. Fig. 3 illustrates the main classes of the Product Model ontology and their relations. Description of these classes and main properties are given in Table 1. The focus is on the matchmaking perspective and thus some of the properties are omitted.

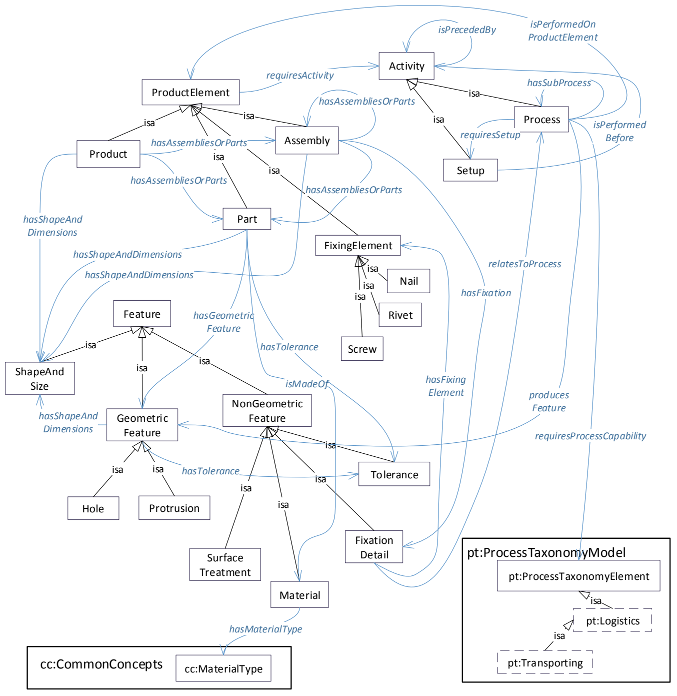

Fig. 3. Product Model.

The Product Model ontology imports the Process Taxonomy, and models the product's processing requirements as instances of the correct *ProcessTaxonomyElement* sub-classes. E.g. if the product requires a screwing process, this requirement is modelled as an instance of the taxonomy class *Screwing*. In the Product Model the parametric requirements related to the processes are modelled as property restrictions of the *ProcessTaxonomyElement* sub-classes. An example of such parameter may be the minimum torque required for screwing.

Table 1. Main classes of the Product Model.

Class	Description
Product Element	Parent class for different elements of a product. Refers to Activity through *requiresActivity* object property. Defines also weight by *mass* datatype property.
Assembly	Represents assembly of two or more parts or other assemblies through *hasAssembliesOrParts* property. Assembly may refer to FixationDetail through *hasFixation* property. It also refers to ShapeAndSize through *hasShapeAndDimensions* property.
Part	Part is a single entity in the product. Part may refer to geometric features through *hasGeometricFeature* property. The material is represented through *isMadeOf* property and tolerance information through *hasTolerance*.
Product	Product is the final product that is delivered to the customer. It consists of assemblies and/or parts.
Activity	Parent class for different processes that may take place in the production system. Precedence order of the activities is modelled through *isPrecededBy* property.
Process	Process is the actual assembly or manufacturing process that is to be performed for the product. Process refers to ProductElements through *isPerformedOnProductElement* property. It may also refer to Feature by *producesFeature*. The link to the ProcessTaxonomy is done by *requiresProcessCapability* object property.
Setup	Activities related to setting up the devices.
Support Process	Processes which are not directly required by the product, but are still necessary, e.g. feeding or storing.
Feature	Parent class for different product features.
Geometric Feature	Parent class for geometric features, which may affect to the required capabilities. Refers to ShapeAndSize through *hasShapeAndDimensions* object property.
NonGeometric Feature	Parent class for non-geometric features, e.g. SurfaceTreatment and Tolerances.
FixationDetail	Contains details about fixation. Refers to certain kinds of fixing elements through *hasFixingElement* property. Includes property *numberOfFixingElements* to indicate how many similar fixing elements are used within the same process. Refers to the Process through *relatesToProcess* property.
Material	Used to store specific material instances with certain lot number. Refers to the MaterialType through *hasMaterialType* property.
ShapeAnd Size	Parent class for different shape-related sub-classes: BoxShape, ConeShape, CylinderShape, PyramidShape and SphereShape. These sub-classes contain relevant dimensional datatype properties.
Process Taxonomy Element	Parent class for the hierarchical classification of different manufacturing and assembly processes. The sub-classes contain the relevant product requirement related process parameters. E.g. Screwing class has parameter restrictions for defining the required torque (exact, max, min) of the needed screwing process.

4. Capability matchmaking

The capability matchmaking uses the product requirement description as an input information from the product side, and resource capability descriptions from the other side, and tries to make a match between these two. The matchmaking is done with a Matchmaking Ontology Model (OWL), which imports the Product Model and Resource Model. In the following sections we will discuss the matchmaking viewpoints and the rules needed during the matchmaking process.

4.1. Matchmaking viewpoints and process

The overall matchmaking process [10] has three viewpoints, which all require their specific rules. We have discussed the combined capabilities and their parameter calculation in [14] and the interface matchmaking in [17]. This paper and section focuses on the matchmaking between product requirements and resource capabilities.

The Process Taxonomy model, imported by both the Resource Capability Model and Product Model, allows matching between the product requirements and the resource capabilities at the capability concept name level. The activities in the product requirement description refer to specific instances of the *ProcessTaxonomyElement* sub-classes. The capability instances defined in the Resource Model also refer to a certain level in the process taxonomy, or actually they are instances of the taxonomy parent classes. Therefore, there is a direct conceptual match between the products requesting the capabilities and the resources providing the capabilities. The Process Taxonomy also takes care of the sub-sumption based reasoning, i.e. an instance belonging to any of the classes is also an instance of its parent classes. This means that if the product requires material removing process, any capabilities providing material removing (e.g. milling or turning) can be suggested. To make the detailed matching the capability parameters need to be compared against the parameters of the product requirement. This is discussed in the next section.

The matchmaking requires as an input the product requirement description and the available resource descriptions. In case of reconfiguration scenario, the existing system description may also be provided as an input. The resource information is collected from a resource catalogue, where resource providers have provided descriptions of their offerings in the Resource Description format [19]. The product requirement information can be collected from any process planning tool used by the process designer, assuming there is a mapper, which translates the information into Product Model structure and format. These inputs are provided to the matchmaking software by external design and planning systems, which control the matchmaking process. The capability-matching algorithm takes the capability requirements (i.e. *Activity* instances), and match them with the existing capabilities or create new resource combinations that match with the requirements. The found matches are provided to the external design tools, which will then make the decision about resource selection and system configuration based e.g. the availability and other valued criteria.

4.2. Matchmaking rules

Capability matchmaking rules are used to compare the parametric requirements of the product with the parameters of the resource capabilities. For instance, if product requires drilling a hole with diameter D = 10 mm and length L = 50 mm, then the rules will be used to check if the provided “Drilling” capability has *hole_diameter* exactly 10 mm, and *max_drilling_depth* the same or larger than 50 mm.

For rule implementation we use SPIN (SPARQL Inferencing Notation). SPIN can be used to link class definitions with SPARQL queries to capture constraints and rules that formalize the expected behavior of those classes. A suitable reasoner tool such as SPIN API can then infer the extra information created by the rules and use it for example in SPARQL query execution [18].

In the Matchmaking Ontology, there are SPIN rules attached to the *ProcessTaxonomyElement* classes. For instance, there is one rule (Rule 1), attached directly to the *ProcessTaxonomyElement* parent class. It identifies the capabilities, which match with the requirement on the concept name level – this means the capability instances, which belong to the required process class or one of its sub-classes. The found matches are saved by linking them to the requirement through *hasCapabilityNameMatch* object property.

Rule 1: Finding match with capability name level

```
CONSTRUCT {
   ?this :hasCapabilityNameMatch ?instance .
}
WHERE {
   ?this a ?class .
   ?capability (rdfs:subClassOf)+ ?class .
   ?capability (rdfs:subClassOf)* cm:Capability .
   ?instance a ?capability .
}
```

Secondly, there are specific rules attached to each sub-class of the *ProcessTaxonomyElement*. They identify the capabilities that match with the product requirements on parameter level, and save those by linking them to the requirement with *canBeImplementedWith* object property. The Example Rule 2 finds capabilities, which match with the required screwing process based on the screw type, size and torque requirements.

Rule 2: Finding match with capability parameter level

```
CONSTRUCT {
   ?this :canBeImplementedWith ?instance .
}
WHERE {
   ?this pm:screwType ?requiredType .
   ?this pm:screwDiameter ?diameter .
   ?this pm:requiredTorque_exact ?requiredTorque_exact .
   ?this pm:requiredTorque_max ?requiredTorque_max .
   ?this pm:requiredTorque_min ?requiredTorque_min .
   ?this :hasCapabilityNameMatch ?instance .
   ?instance cm:screwType ?instanceType .
   ?instance cm:screwSize ?size .
   ?instance cm:torque_max ?torque_max .
   ?instance cm:torque_min ?torque_min .
   FILTER (?requiredType = ?instanceType) .
   FILTER (?diameter = ?size) .
   FILTER (?requiredTorque_exact >= ?torque_min) .
   FILTER (?requiredTorque_exact <= ?torque_max) .
   FILTER (?requiredTorque_max >= ?torque_min) .
   FILTER (?requiredTorque_min <= ?torque_max) .
}
```

5. Case example

Figure 4 exemplifies a matchmaking scenario in a case of a cell phone body, where four screws need to be attached. On the upper part of the figure, the product requirements are described. The lower part of the figure shows the current system and its capabilities. Only some part of the capabilities and capability parameters are displayed, in order to maintain the readability. We presented the conceptual idea of this case study initially in [20]. However, at that time both the Capability Model and Product Model ontologies had a very different structure, and the matchmaking relied on hard-coded rules on program level, not on semantic rules.

Based on the given descriptions and the SPIN rules it is possible to reason out if the existing system has the required capabilities to perform the screwing operations. For example, in case of the "Picking" process, the high-level matching finds the "Picking" capability of the Screwing robot combination. The detailed capability matchmaking will then check if the parameters match, i.e. if the screws can actually be grasped with the resource combination. The combined capability parameters shown in the lower rightmost of the figure are automatically calculated based on the combined capability rules. For instance, the payload of the "Picking" capability is calculated as: Minimum of {Robot payload minus (screwdriver mass + screwing head mass); holding force / 9.81 m/s^2}. The matchmaking results that the combined payload of the screwing robot is bigger than the mass of the screw, and thus matches with the requirement.

Similarly for the "Screwing" step, the high-level matchmaking finds the "Screwing" capability of the Screwing robot combination. For detailed matchmaking, the Rule 2 from the previous section is used. It ends up finding a match, as the screw type, size, and the provided torque range match with the requirement. The matching shows that the screws can be fastened with the existing screw driver. Any change in screw size would require physical changes to the system.

All the parameter examples given in the product requirement side, e.g. "speed" or "feedRate", are not actually determined by the product itself, but by the production performance targets set by the process or system designer. The input for the matchmaking may be either a pure product requirement description, or it may also include these production performance related parameters. Such parameters can be used, for instance, in later phases of the system design and reconfiguration, when making decisions about the optimal resource selection.

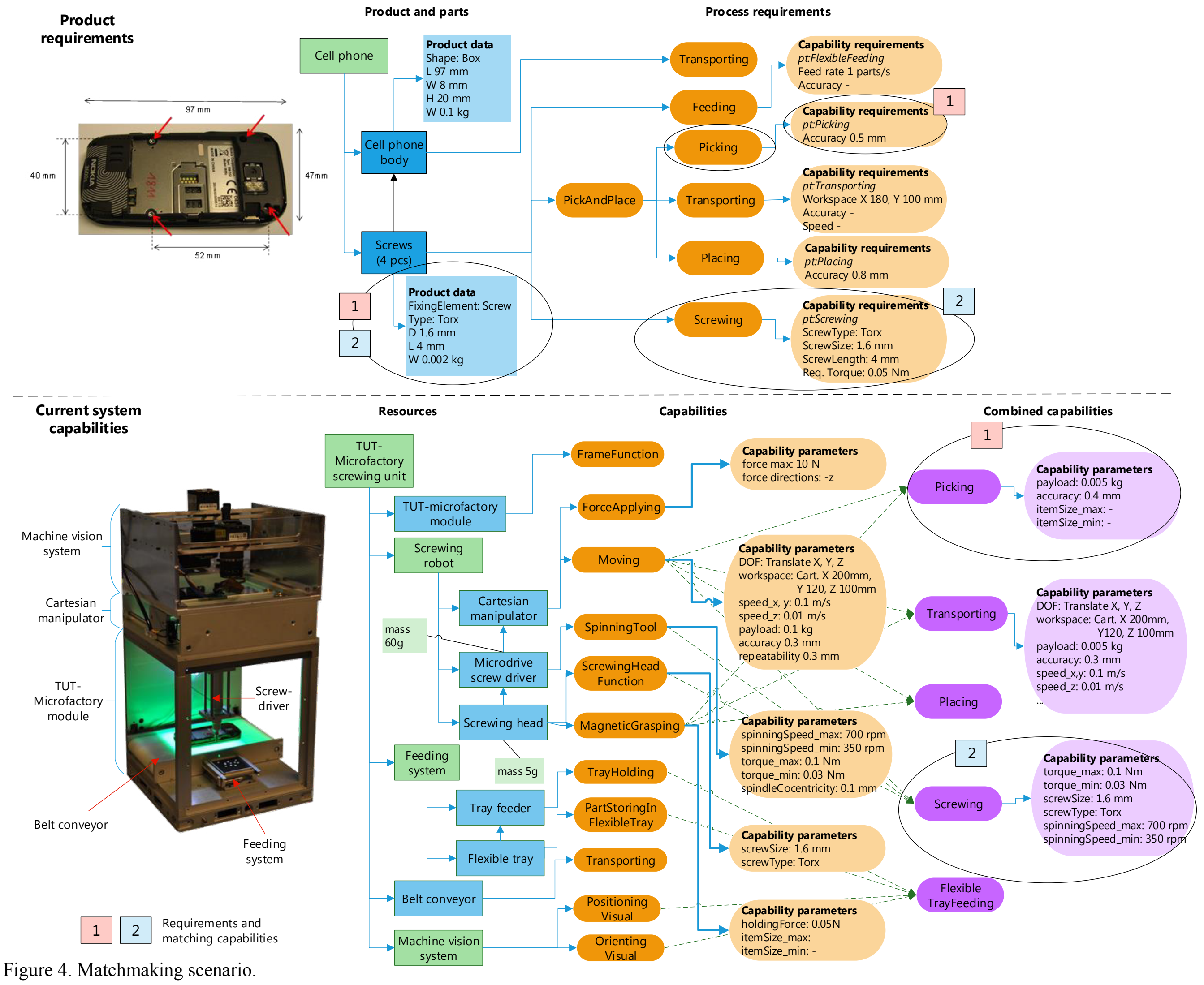

Figure 4. Matchmaking scenario.

6. Discussion and Conclusions

We presented a Product Model ontology that can be filled with instance information, and used as an input for automatic capability-based matchmaking of product requirements against resource capabilities. Such matchmaking can be exploited in greenfield and brownfield design, to partly automatize the search for suitable resources. The ontology is specifically tailored to the capability matchmaking purpose, neglecting other product information, e.g. related to product lifecycle.

Compared to other approaches, which target to similar matchmaking, our approach clearly separates the models used to describe the products and resources. The Process Taxonomy acts as a connector between these two. This approach allows the designer to describe the products even without any knowledge about the complex Capability Model. Furthermore, the Process Taxonomy allows the matchmaking procedure to suggest alternative resources as the same required process may be achieved by multiple different capabilities. This approach can facilitate the Design for Manufacturing and Assembly (DFMA) and support the reuse of the existing resources.

We identified that it may be difficult to define all the requirements affecting to the resource selection at once. This is because the selection of some resources may propagate further requirements for the rest of the system. For instance, the width of the TUT-microfactory module affects to the needed transportation distance from one station to another. This can not be pre-defined in the Product Model. The requirement description needs thus be iteratively appended after some resources have been selected and more information becomes available.

The capability matching actions discussed in this paper can be done automatically based on the defined SPIN rules. However, the results need to be validated by the human designer, because often the combined capability rules provide only crude estimations of the capability parameters of co-operating resources. This has been discussed in [14]. Despite the inaccuracies in combined capability calculations, we believe that the presented matchmaking approach provide a valuable aid for the system designer to find appropriate resource solutions from a large amount of input data. It is possible to explore large resource catalogues and rapidly filter out the unsuitable resources, leaving only the possible resources and resource combinations for the given requirement. Therefore, it will ease and speed up the system design and adaptation planning. Furthermore, it allows larger amount of different configurations to be analysed, compared to traditional manual design and matchmaking approach.

As our future work, we will implement a Capability Matchmaking Software that will take the required inputs from the external system design and reconfiguration planning software, apply the matchmaking rules in a coordinated manner, and provide the matchmaking results back to these external software. Later, we also plan to apply other types of rules that could append the matchmaking results with performance related information, such as evaluations of process times for each step.

Acknowledgements

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 680759 (www.recam-project.eu).

References

[1] Strzelczak, S. Towards Ontology-Aided Manufacturing and Supply Chain Management – A Literature Review. In Proceedings of Advances in Production Management Systems: Innovative Production Management Towards Sustainable Growth, 2015:467-475.

[2] Ameri F, Urbanovsky C, McArthur C. A Systematic Approach to Developing Ontologies for Manufacturing Service Modeling. In Proceedings of the Workshop on Ontology and Semantic Web for Manufacturing, 2012, 14 p.

[3] Järvenpää E, Lanz M, Siltala N. Formal Resource and Capability Models supporting Re-use of Manufacturing Resources. Procedings of 6th International Conference on Through-life Engineering Services, 2017, 8 p.

[4] Pfrommer J, Stogl D, Aleksandrov K, Schubert V, Hein B. Modelling and Orchestration of Service-based Manufacturing Systems via Skills. In Proceedings of the IEEE Emerging Technology and Factory Automation, 2014.

[5] Backhaus J, Reinhart G. Digital description of products, processes and resources for task-oriented programming of assembly systems. Journal of Intelligent Manufacturing, 2017;28(8):1787–1800.

[6] Bengel M. Model-based Configuration – A Workpiece-centred Approach. In Proceedings of ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots, 2009:689–695.

[7] Rauschecker U, Stöhr M. Using Manufacturing Service Descriptions for flexible Integration of Production Facilities to Manufacturing Clouds. In Proceedings of the 18th International Conference on Engineering Technology and Innovation, 2012.

[8] Shin J, Kulvatunyou B, Lee Y et al. Concept Analysis to Enrich Manufacturing Service Capability Models, Procedia Computer Science, 2013;16:648-657.

[9] Ameri F, Patil L. Digital manufacturing market: a semantic web-based framework for agile supply chain deployment. Journal of Intelligent Manufacturing, 2012;23(5):1817–1832.

[10] Järvenpää E, Siltala N, Hylli O, Lanz M. Capability Matchmaking Procedure to Support Rapid Configuration and Re-configuration of Production Systems. Procedia Manufacturing, 2017;11:1053-1060.

[11] Lohse N, Maraldo T, Barata J. EUPASS std-0007: Assembling Process Ontology Specification. 2008. 38 p.

[12] CO2PE!. CO2PE! - Taxonomy. 2010. Available at: https://www.co2pe.org/?Taxonomy [Accessed February 5, 2011].

[13] W3C. SPARQL Query Language for RDF. 2008. Available in: http://www.w3.org/TR/rdf-sparql-query/ [Accessed 10.3.2016].

[14] Järvenpää E, Hylli O, Siltala N, Lanz M. Utilizing SPIN rules to infer the parameters of combined capabilities of aggregated manufacturing resources. Submitted to 16th IFAC Symposium on Information Control Problems in Manufacturing (INCOM), 2018.

[15] SPIN working group. SPIN – SPARQL Inferencing Notation. 2017 Available in: http://spinrdf.org/. [Accessed 15.10.2017].

[16] Sure Y, Staab S, Studer R. Ontology Engineering Methodology. In: Staab S, Studer R (Eds.), Handbook on Ontologies, 2nd edition. 2009:135-152.

[17] Siltala N, Järvenpää E, Lanz M. Creating resource combinations based on formally described hardware interfaces. In Proceedings of 8th International Precision Assembly Seminar, 2018.

[18] Knublauch H. The TopBraid SPIN API. 2016. Available in: http://topbraid.org/spin/api/ [Accessed 1.4.2017]

[19] Siltala N, Järvenpää E, Lanz M. Formal Information Model for Representing Production Resources. In Proceedings of Advances in Production Management Systems: Initiatives for a Sustainable World, 2016:53-60.

[20] Järvenpää E, Lanz M, Tuokko R. Application of a capability-based adaptation methodology to a small-size production system. Int. J. Manufacturing Technology and Management, 2016;30(1/2):67-85.