Final author version of manuscript submitted to Journal of Manufacturing Systems

Implementation of capability matchmaking software facilitating faster production system design and reconfiguration planning

Eeva Järvenpää[1]
Faculty of Engineering and Natural Sciences, Tampere University
Korkeakoulunkatu 6, 33720 Tampere, Finland
eeva.jarvenpaa@tuni.fi
Tel. +358-40-8490869
ORCID 0000-0001-6513-135X

Niko Siltala
Faculty of Engineering and Natural Sciences, Tampere University
Korkeakoulunkatu 6, 33720 Tampere, Finland
niko.siltala@tuni.fi
ORCID 0000-0001-6456-1251

Otto Hylli
Faculty of Information Technology and Communication Sciences, Tampere University
Korkeakoulunkatu 6, 33720 Tampere, Finland
otto.hylli@tuni.fi

Minna Lanz
Faculty of Engineering and Natural Sciences, Tampere University
Korkeakoulunkatu 6, 33720 Tampere, Finland
minna.lanz@tuni.fi
ORCID 0000-0003-2182-4669

[1] Corresponding author

Abstract

Smart manufacturing calls for rapidly responding production systems which help the manufacturing companies to operate efficiently in a highly dynamic environment. Currently, the system design and reconfiguration planning are manual processes which rely heavily on the designers' expertise and tacit knowledge to find feasible system configuration solutions by comparing the characteristics of the product to the technical properties of the available resources. Rapid responsiveness requires new computer-aided intelligent design and planning solutions that would reduce the time and effort put into system design, both in brownfield and greenfield scenarios. This article describes the implementation of a capability matchmaking approach and software which automatizes the matchmaking between product requirements and resource capabilities. The interaction of the matchmaking system with external design and planning tools, through its web service interface, is explained and illustrated with a case example. The proposed matchmaking approach supports production system design and reconfiguration planning by providing automatic means for checking if the existing system already fulfils the new product requirements, and/or for finding alternative resources and resource combinations for specific product requirements from large search spaces, e.g. from global resource catalogues.

Keywords: Production system design, Production system reconfiguration, Resource representation, Capability modelling, Capability matchmaking, Matchmaking software, Matchmaking Web Service

1. Introduction

Future smart manufacturing calls for rapidly responding production systems that can adapt to the required changes in processing functions, production capacity, and the dispatching of orders. This is due to the ever-increasing requirements for highly flexible production of individualized products in small batches [1,2]. Currently, production system design and reconfiguration planning are manual processes and are heavily dependent on the designers' expertise and tacit knowledge to find feasible system solutions by comparing the characteristics of the product to the technical properties of the available resources. This slow process sets limitations on the number of potential system configuration alternatives that can be considered. Meeting the requirements of fast adaptation calls for new computer-aided intelligent planning and decision support solutions that would reduce the time and effort put into system design, both in brownfield (reconfiguration) and greenfield (new system design) scenarios.

A key enabler of smart manufacturing is the virtualization of the physical assets of the manufacturing, namely resources and products [2,3,4]. There is a need for formal, structured representation of resources and products that allow the resource vendors to describe the functionality of their offerings in a comparable manner, and system designers to make a match between the product requirements and resource capabilities. A similar matchmaking concept, but on a wider network scale, is Cloud Manufacturing. It is a service-oriented business model, with the idea being to share manufacturing capabilities and resources through a cloud platform and to form temporary, reconfigurable supply chains on demand [5,6,7]. An essential enabler for the Cloud Manufacturing concept is the virtualization of the resource capabilities and especially the services they can offer to the clients of the platform, as well as service searching and matching against the task requirements [7,8,9].

Formal engineering ontologies and other Semantic Web technologies have become popular solutions for addressing the semantic interoperability issue in heterogeneous distributed environments

[10,11,12,13,14]. In the context of distributed intelligent systems, such as agent-based or service-oriented systems, ontologies play a key role as they provide a shared, machine-understandable vocabulary for information representation and exchange among dispersed actors [10,14]. Several different semantic ontology-based descriptions have been proposed for the description of the services in the Cloud Manufacturing context (e.g. [15,16,17]). Lu and Xu [6] presented a service composition and mapping approach based on ontological description of the services and Jena rules to compare the service request with the offering. Manufacturing Service Description Language (MSDL) was developed as a formal domain ontology for representing the capabilities of manufacturing services, focusing on mechanical machining and metal casting services [10], and used for a matchmaking methodology which aims to connect buyers and sellers of manufacturing services in distributed digital manufacturing environments [18]. Ameri and McArthur [19] utilized SWRL (Semantic Web Rule Language) for intelligent supplier discovery based on the services they provide. In the ManuCloud project, an XML-based manufacturing service description was developed to enable the Manufacturing-as-a-Service operation principle in production networks [20].

Despite there being several approaches to service description and resource virtualization, the current research around service and capability matching has concentrated more on the theory and framework, rather than comprehensive descriptions of the resources and services, and practical implementations of the matchmaking. In addition, the previous research efforts to describe manufacturing resource capabilities have not considered the combined capabilities of multiple co-operating resources, and consequently have not included mechanisms to infer the aggregated parameters of combined capabilities. Thus, the existing approaches cannot provide a solution for capability matchmaking in the context of production system design and reconfiguration planning.

The European Union-funded project ReCaM [21] developed computerized support for the system design and reconfiguration planning process. The goal of our research was to develop a concept and technical implementation of a capability matchmaking system which should support production system designers and reconfiguration planners in finding feasible resources for specific product requirements from large search spaces. The matchmaking system should provide an easy interface to the other design and planning software utilized by the designers for the actual system design and optimization tasks. In our earlier works, we presented the formal models for representing the resources and products [22,23,24] which provide a foundation for the capability-based matchmaking of product requirements and resource offerings [25,26]. In this article, we will describe the implementation of the matchmaking software and how it can be utilized by external design and planning tools.

The article is organized as follows. In Section 2 we will introduce the capability matchmaking approach and its associated concepts and information models. In Section 3 we will explain the implementation of the matchmaking software and its architecture, and its interaction with other design and planning systems. In Section 4 we will illustrate, through a case example, how external design and planning tools can utilize the matchmaking software through its Web Service interface and show test results obtained from running different matchmaking scenarios. Finally, we will end with discussion and conclusions in Sections 5 and 6.

2. Capability matchmaking concept and information models involved

The matchmaking system intends to make the system design and reconfiguration planning procedure easier by automatically suggesting alternative resource combinations for specific product requirements. The matchmaking system utilizes formal representation of product requirements as well as resources and their capabilities and interfaces as an input, and tries to make a match between these by using rule-based reasoning. We will briefly explain these aspects in the following sub-sections.

2.1. Information models involved

The foundation of the capability matchmaking is the formal information models representing the product requirements and available manufacturing resources. We have developed OWL-based (Web Ontology Language) information models to represent the information that is needed, and introduced them in our earlier publications [22-27]. Figure 1 illustrates these models and their import structure. The models are available to download at [29]. The Ontology Engineering Methodology [28] was followed during the development of the models.

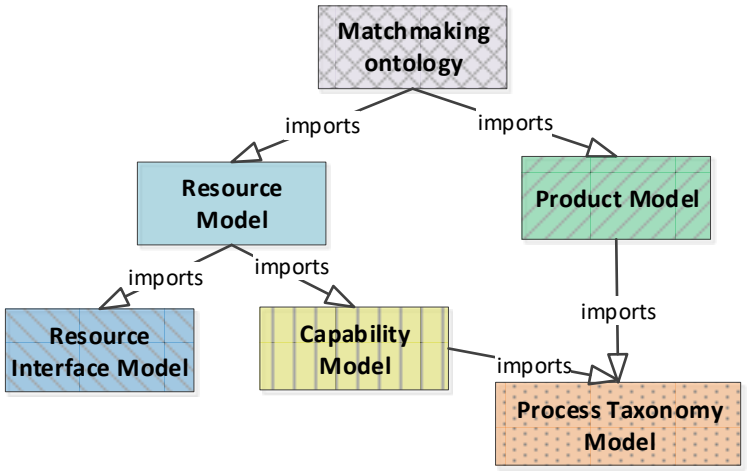

Figure 1. Information models used for the capability matchmaking.

The capability matchmaking is performed with the Matchmaking Ontology Model, which imports the Product Model and Resource Model, as shown in Figure 1. Figure 2 illustrates the main classes and relations of the Matchmaking Ontology, including the inherited classes and relations from the imported models, discussed in the following paragraphs. It also includes some additional classes and properties used by the matchmaking rules, discussed in the following section. Similar colours and patterns are used in the figures to indicate the model (Figure 1) to which the classes (boxes in Figure 2) belong. For instance, “Activity” is a class in the Product Model.

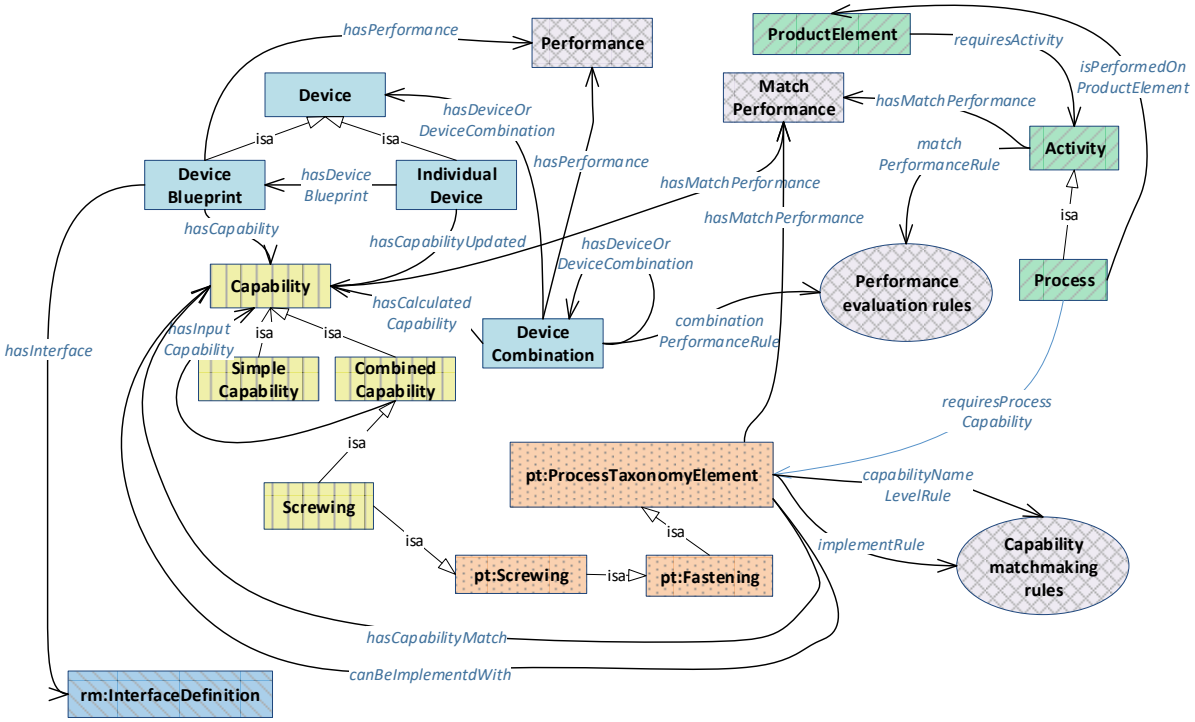

Figure 2. Simplified view of the Matchmaking Ontology (for readability does not include all classes and relations).

The Resource Model ontology [22] is used to describe the available manufacturing resources, including their capabilities, interfaces and other characteristics, as well as systems composed of multiple resources. Class "Device" (Figure 2) is used to model machine and tooling resources, while humans should be modelled with another class, not visible in the figure. The Resource Model imports two other ontologies, namely the Resource Interface Model and Capability Model. The Resource Interface Model [24] is used to give a formal description of the resource interfaces. Similarly to the design of modular products [30], considering interfaces plays an important role in enabling the interchangeability and independence of resource elements. This information helps to identify whether two or more resources can be connected together from their interface perspective.

The Capability Model [22] formalizes the functionalities of the resources and parameters related to these functionalities. It also defines the relations between simple (atomic) and combined capabilities through the *hasInputCapability* object property. For instance, a robot can have the simple capability "Moving" and a gripper can have the simple capabilities "Grasping" and "Releasing". These relations are modelled through the *hasCapability* object property between instances of the "Device" and "Capability" classes. Together, the robot and gripper can have the combined capabilities "Pick and Place" and "Transporting". On the basis of the formalized relations, the potential resource combinations that have a certain combined capability can be identified programmatically by utilizing the information provided by SPARQL queries. The parameters of the combined capabilities can be inferred on the basis of the rules discussed in the next section and linked to the specific device combination through the *hasCalculatedCapability* object property.

The Capability Model imports another ontology called the Process Taxonomy Model. This model categorizes different manufacturing and assembly processes in a hierarchical structure. The "Capability" classes are linked to the "ProcessTaxonomyElement" classes according to what kind of process they can provide. This linkage is implemented as a direct *is-a* (sub-class) relationship between

the “Capability” class and “ProcessTaxonomyElement” Class. For instance, “Screwing” is a sub-class of the “Fastening” class.

The Product Model ontology [23] can be used to describe the product requirements for the matchmaking. The Product Model describes the parts and their basic characteristics, sub-assemblies and the parts they contain, the processes related to the parts and sub-assemblies, the capability requirements related to the processes, and the sequence of the processes. The Product Model imports the same Process Taxonomy as the Capability Model. This allows a link to be built between the requirements and capabilities during the matchmaking. This link is established through the *requiresProcessCapability* object property between the instances of the “Process” class and the instances of the “ProcessTaxonomyElement” class. The parametric requirements related to a specific process capability, e.g. the required torque for screwing, are defined as properties of the “ProcessTaxonomyElement” sub-classes.

2.2. Matchmaking stages and rules

The overall matchmaking process [25] has three stages, all of which require specific algorithms and rules: 1) defining the combined capabilities and calculating their parameters when new resource combinations are formed; 2) checking the interface compatibility of the resources when new resource combinations are formed; 3) matching the product requirements against the capabilities of the combined resources. We have discussed the combined capabilities and their parameter calculation in [26], interface matchmaking in [24], and capability matchmaking in [23]. All three of these stages are included into the operational chain of the capability matchmaking software.

For the rule implementation we use SPIN (SPARQL Inferencing Notation). SPIN is a W3C Member Submission that has become the de facto industry standard to represent SPARQL rules and constraints on Semantic Web models [31]. SPIN can be used to link class definitions with SPARQL queries to capture constraints and rules that formalize the expected behaviour 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 [32]. We use SPIN in both the combined capability parameter inference [26] and capability matchmaking [23]. The capability matchmaking rules are attached to the sub-classes of the “ProcessTaxonomyElement” in the Matchmaking Ontology, as shown in Figure 2. The properties *hasCapabilityMatch* and *canBeImplementedWith* link the capability requirements (i.e. instances of “ProcessTaxonomyElement” sub-classes) with the “Capability” instances, as illustrated in Figure 2. The property *hasCapabilityMatch* indicates that the capability matches the requirement on the capability concept name level, while the property *canBeImplementedWith* indicates that the capability parameters also match the requirement. These linkages are established as part of the matchmaking process when the matchmaking rules find the matches. First, the resource combinations with required capabilities are formed. Secondly, the interface compatibility of the resources is checked and incompatible combinations are filtered out. Thirdly, the capability parameters for the remaining combinations are calculated with the combined capability rules. Fourthly, these capability parameters are compared against the parameters of the requirements with capability matchmaking rules to find detailed matches. For the detailed matches also estimated performance is calculated with performance evaluation rules. This contains the estimated duration of the specific process step with the suggested resource or resource combination.

3. Implementation of the capability matchmaking software

The capability matchmaking software follows the principles of client-server architecture. It is constructed from three main components: the capability matchmaking web service, the software packages for executing the capability matchmaking process, and the formal information models, discussed in the previous section. The web service and associated software modules are deployed and hosted on an Apache Tomcat server [33]. This section introduces the matchmaking system architecture and interactions with the external client systems.

3.1. Matchmaking system architecture and technologies used

The capability matchmaking system follows a layered architecture. Figure 3 outlines the various layers and the interactions between them. It also illustrates the technologies and languages used for the software implementation. The Data Model layer contains the ontology and other data models needed for the matchmaking. The Data Layer represents the actual data, i.e. instances, used during the matchmaking. The Business and Data Access layers run and execute the matchmaking procedures. The topmost layer represents different client systems which interact with the web service component of the system in order to trigger matchmaking or to obtain the matchmaking results.

The Web Service layer is implemented as a RESTful web service. This choice of interface allows easy and loose coupling for client applications to connect with the matchmaking system. It was developed with the help of JAX-RS API [34] and its open source reference implementation Jersey [35], both of which ease and harmonise the development of the RESTful application. The Web Service layer performs multiple tasks. It receives the various request messages from the client systems in the XML or JSON formats, validates the inputs in the messages received, and produces the response messages. After validation, the Web Service Layer gathers the input resources from different catalogue(s)/database(s) and invokes the matchmaking process in the Business layer.

The most important packages in the architecture from the matchmaking reasoning perspective are the Capability Query Library (CQL) and the Matchmaker. The Matchmaker is responsible for sequencing and managing the matchmaking process and performing the actual capability matchmaking for the incoming requests. It takes care of the execution of the various SPIN rules through the Java-based CQL API (Application Programming Interface). It creates resource combination possibilities, calculates the combined capability parameters for the resource combinations, checks the interface compatibility of the resources, executes matchmaking rules from the Matchmaking Ontology, and constructs the matchmaking result from the rule inferences. CQL uses the open source Jena semantic web framework [36] and Openllet reasoner [37] for working with the ontology models. Jena and Openllet themselves do not support SPIN, so another open source library that builds on top of Jena, called SPIN API [32], is used to execute the SPIN rules.

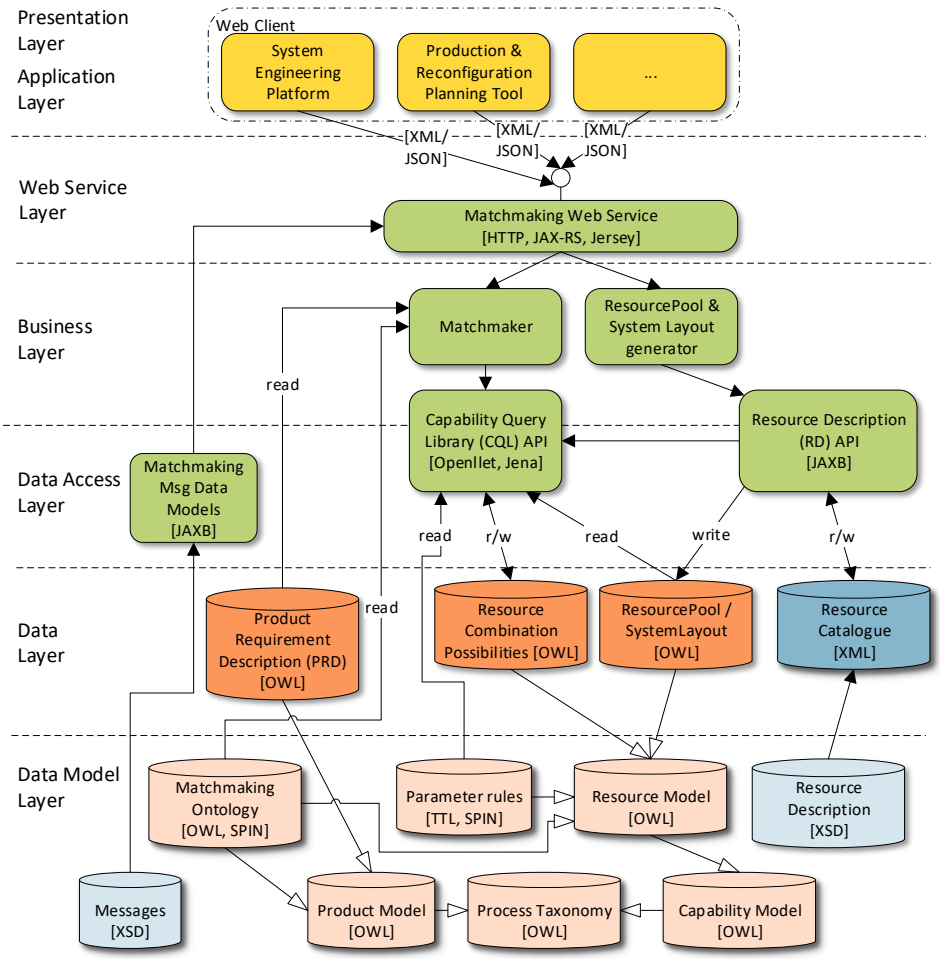

Figure 3. Overall software architecture of the Capability Matchmaking System.

3.2. Interaction with external software

The Capability Matchmaking service can be utilized by any design and planning system to trigger matchmaking requests and to receive the matchmaking results by following the specified message schemas (Figure 4). The structure of the messages that are exchanged is defined by XML Schemas (XSD), which are published for the use of the client application developers. The schemas are used internally for generating data objects with the help of JAXB (Java Architecture for XML Binding). These data objects enable the easy processing of input and output messages for the capability matchmaking service in both accepted formats – XML and JSON.

The matchmaking process involves searching through the matchmaking search space to identify the resources or resource combinations that match the required capabilities. As an input, the matchmaking system needs to receive the search space to be considered. This includes the Product Requirement Description (PRD) and the set of Resource Descriptions (resource pool) that ought to be considered during the matchmaking process. The matchmaking request can specify which process steps (if not all) in the PRD are included for the specific matchmaking run. Depending on the design scenario, the input is different. In the case of a brownfield scenario, there is an existing system that could be reused for the production of the new product. Thus, a description of the existing system layout should also be included into the input. This contains information about the resources in the current layout and how they are physically connected to each other. The inputs are provided to the matchmaking software by the client application in the form of PRD IDs and RD IDs. The search space is then retrieved and read into the

Matchmaking Ontology from various catalogues storing the actual information content. For instance, the resource information is collected from the Resource Catalogue, where resource providers have supplied descriptions of their offerings in the Resource Description format [27]. The PRD is represented with the Product Model ontology format [23].

The matchmaking system provides the matchmaking result as an output. It contains the matches found for each process step of the input PRD. A process step is marked with *No Match* if no matching resources are found. On the other hand, a *Match* means a solvable process step and it contains a reference to a resource or a resource combination which possesses capabilities matching the requirement. In addition to resource identification and linking information, some information relating to business properties, performance, and reliability is collected and delivered along with the resource record in the matchmaking result. The results are provided to the external design tool, which can then be used to make the decision about the resource selection and system configuration on the basis of the optimization criteria that are valued, such as availability, performance, the smallest number of reconfigurations, or costs.

The matchmaking is a time-consuming process, and therefore the interaction was implemented as asynchronous calls. Figure 4 shows the interaction between an external application (client) with the capability matchmaking system to create a matchmaking search space (sequences 1 and 2), to trigger the matchmaking process with a matchmaking request (sequence 3), and to request the matchmaking result (sequence 4).

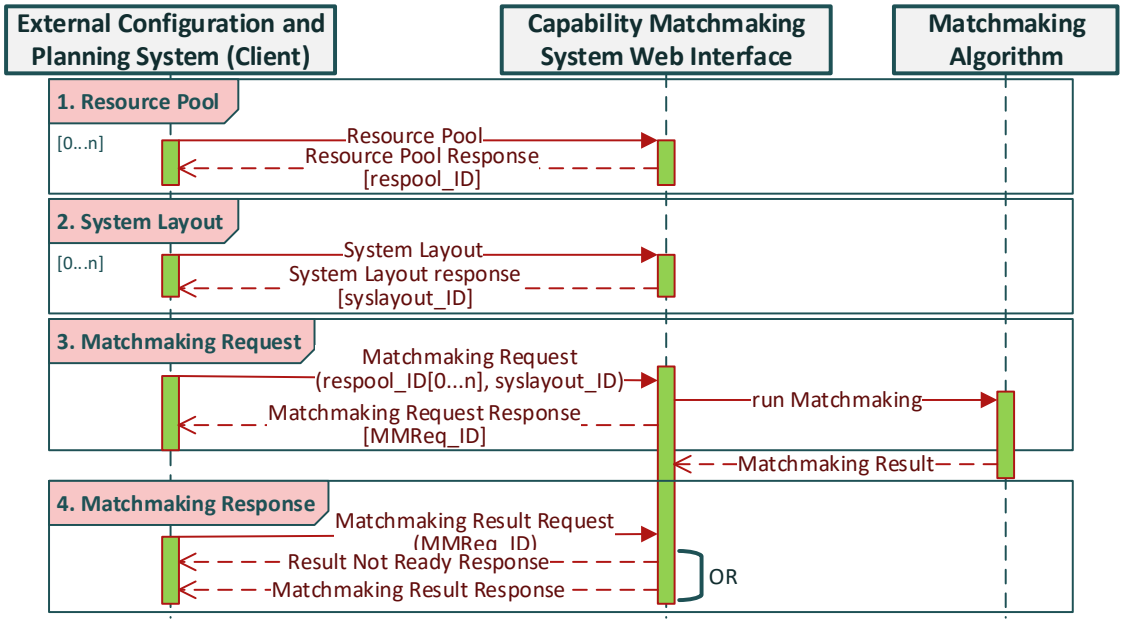

Figure 4. Matchmaking interaction scenario.

4. Testing of the matchmaking system

The matchmaking software is designed to be used as a background service by other design and planning applications. During the ReCaM project, interaction with two such client applications was tested. These were the Flexible System Engineering Platform [38], meant for greenfield system design, and the Integrated Production and Reconfiguration Planning tool [39], intended for brownfield system design. In the following sections, we will first demonstrate the utilization of the matchmaking service by the latter as part of larger integrated system design software chain. The second section focuses on testing the functionality of the matchmaking software itself.

4.1. Case study – interaction of the matchmaking system with an external reconfiguration planning tool

The interaction of the matchmaking system with the Integrated Production and Reconfiguration Planning tool (IPRP) follows the schema defined in Figure 4. The upper right part of Figure 5 shows an HTML view of the partial matchmaking result for a specific case product (named "Bosch_DRE"), including the resources and resource combinations found for each process step, and the estimated process duration with the specific resource or resource combination. The other parts of Figure 5 illustrate the user interface of the IRPP tool and how it utilizes these matchmaking results.

The search space of the matchmaking scenario shown in Figure 5 corresponds to the test #1 from Table 3 discussed in the next section. It includes the PRD for the product "Bosch_DRE" and the full resource catalogue with 67 resources. The suggested resource combinations column in the HTML view shows the already matching resources and resource combinations, such as "rd.Bosch.Press.Anvil.01.01", and new resource combinations created dynamically by the matchmaking software, such as "Screwing_possibility_14". The number 14 indicates that the matchmaking system has created multiple different resource combinations with the capability "Screwing", and the 14th of them matches the parametric requirements of the specific process step "productModel_Bosch_DRE_step11". Furthermore, the result shows that there are two other resource combination possibilities, "Screwing_possibility_21" and "Screwing_possibility_29", that can also satisfy the requirements of the same process step. In other words, they are alternative combinations from which the designer can choose. Figure 5 shows that other process steps, e.g. the fixturing step "productModel_Bosch_DRE_step22", also have multiple different resource options which match the capability requirements.

The matchmaking result information is shown to the designer through the client application's user interface (bottom right of the figure, dashed arrows). The matchmaking result contains a reference to the matching resources and the estimated processing time. The designer can use the data provided to evaluate the feasibility of the suggested resource for a specific production task and system and to allocate the resources to the tasks and workstations, as shown in the bottom right of the figure. The tree view (Layouts) on the left side of the figure shows how the designer has allocated the proposed resources or resource combinations into a specific layout (double arrows).

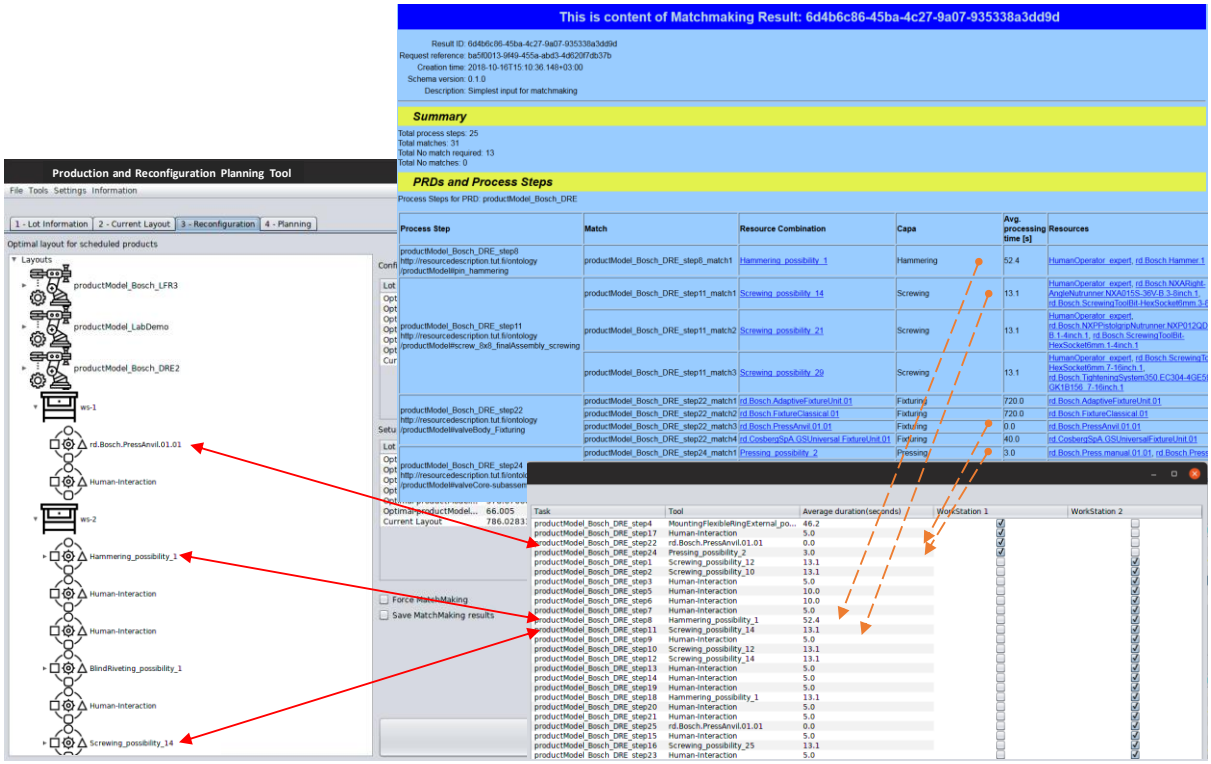

Figure 5. Matchmaking results (upper right part) read in and utilized by the Integrated Production and Reconfiguration Planning Tool developed in Politecnico di Milano (POLIMI). Modified from [39].

The client application's interaction with the matchmaking service is an iterative process. The client application sends matchmaking requests, gets matchmaking results, and shows the results to the system designer, who then makes the resource selections, builds layouts, and decides how the interaction with the service should continue. The search space can differ between the matchmaking rounds. During a typical reconfiguration scenario, the designer using the client application first builds the search space by sending the ID of the new product (or products) and a description of the current system layout to the matchmaking service (Figure 4/Sequence 2). After this, the designer creates and submits the first matchmaking request (Figure 4/Sequence 3) to find out if the current production system can produce the new product.

If not all the tasks in the PRD can be fulfilled by the current system, the designer can submit a second request to the matchmaking service. In this case, the arguments are the PRD and the same system layout as earlier, but now it is allowed for the layout to be broken down into individual resources, and those resources can be re-organized into new combinations by the matchmaking service. If some production tasks still cannot be solved after the re-organization, new production resources need to be added to the system. The third matchmaking request could thus include the selected tasks from the PRD (which have not yet been solved by any resource or resource combination), the system layout in a deconstructed way, and an additional resource pool (Figure 4/Sequence 1). The resource pool may be extended throughout the matchmaking rounds, e.g. starting from spares and free resources at the production site's warehouse, and ending with global resource catalogues containing resources from multiple resource providers. The matchmaking process can be repeated, with different inputs, until all the tasks have a match or other optimization criteria for resource selection are met, e.g. the smallest number of reconfigurations.

4.2. Testing the system functionality and validating the results

We ran several matchmaking scenarios with different search spaces to test the functionality of the software and validity of the results obtained. Six different case products and their assembly processes were used: three different valve products from the process and agriculture industries, a manifold and pitch trimmer from the aviation industry, and a simple laboratory product with a disc stacking process. Table 1 lists those products, how many process steps are needed for their assembly, for how many process steps a capability matchmaking requirement has been defined in the Product Requirement Description (PRD), and what kind of capabilities are required. In some cases the resource to be used (e.g. a human operator) has already been pre-defined by the designer. In such cases, there is no need to model the requirement for those process steps in the PRD and no need to impose a load on the matchmaking system.

Table 1. Products used for testing the matchmaking.

Product (PRD)	**Total number of process steps**	**Number of process steps which require matchmaking**	**Required capabilities**
DRE (valve)	25	12	Six different capabilities: Fixturing, Pick&Place, Hammering, Screwing, Pressing, Mounting O-ring
2-2_SW (valve)	28	14	Seven different capabilities: Fixturing, Pick&Place, Hammering, BlindRiveting, Screwing, Pressing, MountingO-ring
LFR (valve)	8	4	Five different capabilities: Fixturing, Pick&Place, Hammering, BlindRiveting, Screwing
Pitchtrimmer	29	29	Four different capabilities: Fixturing, Pick&Place, Pressing, ThreadTightening
Manifold	18	18	Five different capabilities: Transporting, Fixturing, Pick&Place, Pressing, ThreadTightening
LabDemo	4	4	Two different capabilities: Pick&Place, Feeding

The resources and resource pools used as an input for the matchmaking are presented in Table 2. Altogether, 67 different catalogue resources (“Device Blueprints”) and one existing resource (“Individual Device”) were used. The resources included different robots, grippers, fixtures, screwdrivers, drills and their associated bits, presses, O-ring mounting devices, and hammers, as well as human operators. ResPool 1 contains all the catalogue resources, and SysLayout 1 is a description of the existing production system layout for the product “DRE”. It includes the actual physically existing resource instances (“Individual Devices”), their catalogue representation (“Device Blueprints”) and combinations of these individual instances (“Device Combinations”) according to the production system layout. The last column “Capabilities” indicate how many capabilities, either “Simple” or “Combined” are modelled for these resources in their resource description.

Table 2. Resource pools used for testing the matchmaking.

	Resources		Device Combinations		Capabilities	
Resource Pool	**Device Blueprint**	**Individual Device**	**Existing**	**Test**	**Simple**	**Combined**
ResPool 1	67	1	0	0	109	18
SysLayout 1	25	26	9	0	40	15

Table 3. Inputs (search space) and results of the matchmaking tests.

	Inputs for matchmaking test (search space)			Matchmaking test results				
#	**PRD**	**Resource Pool**	**Layout Fixed?**	**Created test resource combinations**	**Combined capabilities**	**Found matches**	**Process steps without found matches**	**Processing time (mm:ss)**
1	DRE	ResPool 1	N/A	61	93	31	0	17:54
2	2-2_SW	ResPool 1	N/A	44	76	24	1	12:33
3	LFR	ResPool 1	N/A	40	72	7	0	11:22
4	Pitchtrim	ResPool 1	N/A	8	46	77	3	11:18
5	Manifold	ResPool 1	N/A	9	47	49	0	11:28
6	LabDemo	ResPool 1	N/A	11	50	11	0	02:42
7	DRE	SysLayout 1	T	0	26	17	0	00:58
8	DRE	SysLayout 1	F	19	50	34	0	01:55
9	2-2_SW	SysLayout 1	T	0	26	13	5	00:58
10	2-2_SW	SysLayout 1	F	18	49	27	1	01:58
11	2-2_SW	SysLayout 1 + ResPool 1	F	44	87	31	1	15:12

Table 3 shows the test results from a few matchmaking rounds. Different search spaces were used as inputs during different test rounds: 1) all catalogue resources (#1-#6); 2) only the existing layout and its resource combinations (#7, #9); 3) individual resources in the existing layout (#8, #10); 4) existing resources in the current layout supplemented with all catalogue resources (#11). The column "Layout fixed?" indicates whether the designer has allowed the current layout to be broken down into individual resources (true/false), which means that new combinations can be built from the same resources.

The first six rows (#1-#6) of Table 3 show the results obtained for each case product by using all the available catalogue resources as an input. What can be observed is that matchmaking creates temporal test resource combinations with required capabilities, and calculates combined capability parameters for all these combinations. These calculated capabilities are then matched against the parametric requirements of the process steps. The same resource combination or a single resource can provide a solution for many different process steps when the required process parameters are close and/or the resource is flexible (e.g. a offers a wide operating range). This characterizes especially the results of tests #4 and #5, in which a small number of resource combinations provides a large number of matches. A few process steps cannot be fulfilled with available resources, e.g. in tests #2 and #4. This is either because none of the resources or resource combinations can provide the requested capability, or the

parameter range that is offered (e.g. gripper opening, payload, force applied) does not meet the product requirement.

The rows from #7 to #11 show the matchmaking results against the existing System Layout 1. In the case of tests #7 and #9, the layout is fixed, i.e. it is not allowed to be broken down into individual resources. Thus, the matchmaking system does not create any new test resource combinations. Tests #8 and #10 use the same input, but in these cases the layout is not fixed and the matchmaking can split the resource combinations into individual resources and re-organize them freely. New resource combinations are created and more matches are found. For instance, the matchmaking does not find matches for five process steps in case #9, because the system layout was originally built for product "DRE". After allowing re-organization in test #10 (fixed layout = false), four more process steps can be solved with the available resources.

The last column of Table 3 indicates the duration of the matchmaking process with the given search space. This is affected by the number of process steps which require matchmaking, the number of different capabilities required, and the amount of resources in the search space. For instance, test #6 is completed significantly faster than the others with the same resource pool, because the LabDemo case product has only a few process steps, which require only two different capabilities (pick and place and feeding). The resource combination creation has an impact on the matchmaking processing time, as the creation of combinations and calculation of their combined capabilities is a computationally demanding process. For instance, in the case of tests #7 and #9, which use a fixed layout and in which no new combinations are created, the processing takes much less time compared to the non-fixed layout in tests #8 and #10. In the last matchmaking test, #11, two pools – SysLayout 1 and ResPool 1 – are given as the input search space. An increased amount of resources leads to more possible combinations, which also increases the matchmaking processing time.

We analyzed each matchmaking result manually to check that the suggested resource combinations for each process step were valid. The validity was evaluated on the basis of the expected result: if the rules and the matchmaking system behaved as expected and produced the expected result, the result was considered valid. This does not yet mean that the resource combination found would necessarily be feasible in real life. The rules and information representations are simplifications of the real world. The matchmaking does not take into account the different constraints imposed by the other factory facilities and so on. For instance, if a process step was requesting the "Fixturing" capability, the matchmaking system could suggest a press with an integrated fixture, as it has both "Pressing" and "Fixturing" capabilities. However, this may not be a desirable choice for the application and needs to be evaluated by the designer.

In the event of an invalid result, the rules, input data, and/or software code were checked to detect the root cause of the problem. Two different types of invalid results were: 1) a found match which should not be a match; 2) a known match not found. Type 1 errors were easy to detect, while in the case of type 2 errors, some unexpected combinations may have been neglected as a result of being overlooked during the analysis. The root causes were fixed iteratively until there were no invalid results with the given search spaces.

5. **Discussion**

The test results we obtained show that the matchmaking concept and the system we implemented work in practice. The matchmaking service is able to receive the inputs – product requirements and resource pool – and to reason out the matches (if possible) for each process step. The software is able to analyse the requirements of the process steps, look for the necessary capabilities, combine various

resources as device combinations, and infer the combined capabilities and their parameters for these combinations. It is able to reason whether the proposed resources can be physically connected together. Finally, it is able to analyse of which resources or resource combinations match the requirements of each process step and supply this information back to the client who submitted the request. The test results and their analysis show that the software that was developed is able to solve this work chain and it provides valid matchmaking results.

Overall, the matchmaking is a time-consuming operation. In the test cases presented here the processing time varied from 1.5 minutes to 18 minutes, but we have even observed processing times of up to 135 minutes with the same server hardware. However, the project focused on proof of concept, and the performance optimization was not a high priority. The performance could be improved by optimizing the internal algorithms of the matchmaking procedure, for instance the order in which different activities are performed.

The resource representation and capability matchmaking approach that we developed contributes to the existing resource virtualization and matchmaking research by providing means for modelling and reasoning about the combined capabilities of multiple cooperating resources. The resource combinations can be dynamically created for certain requirements on the basis of resource descriptions of single resources. In addition, the existing resource combinations can be decomposed into individual resources to allow automatic reasoning methods to provide suggestions for reconfiguration measures. Furthermore, our approach also describes the parameters relating to the capabilities and utilizes SPIN rules to infer the parameters of combined capabilities from the parameters of the simple capabilities and to insert them into the model. It is a unique approach and implementation that has not been presented by other researchers. Similar conceptual ideas for capability matching have been presented, e.g. in [18]. The main difference, however, lies in the ability to manage combined capabilities automatically. First of all, it allows the resources to be described on a lower level of granularity, and second, it eliminates the need to describe the combined capabilities manually for each possible resource combination.

The capability matching reasoning actions discussed in this article can be performed automatically on the basis of the defined SPIN rules. However, the results should be validated by a human designer, as the information models and rules are always a simplified representation of the real world. For example, the combined capability rules can provide only estimations of the capability parameters of cooperating resources. In many cases, the properties of the combined capabilities emerge as a behaviour of the machine or station as a whole in a certain context and environment, and they cannot be decomposed into the properties of the various components (i.e. simple capabilities). Furthermore, some of the capabilities depend on the physical locations of the combined resources. This information is not currently handled with the Resource Model, and thus cannot be taken into consideration during the matchmaking.

The matchmaking result delivered to the client does not consider the allocation of resources to the process steps and workstations. It handles each process step individually, and thus it is the duty of the client application (or the designer) to consider the optimum resource allocation and layout of the production system. For example, if the designer allocates a specific resource instance to a specific process step and a physical location (e.g. workstation), he/she cannot use the same resource for some later process step in a different location without having more resource instances of the same kind. In this sense, the matchmaking result delivers only potential possibilities from the perspectives of capability parameters and physical interface connectivity, and the client application (or designer) needs to ensure that the production system can actually be built from the available resources.

Currently, the matchmaking rules have been defined and especially tested only for the most common process steps appearing within the case products. The testing of the matchmaking was limited to the six case products and 67 resources mentioned above. Thus, more testing is needed with different products with different process steps, and resources with different capabilities.

6. Conclusions

The goal of our research was to develop a capability matchmaking concept and software system that can partly automatize the search for suitable resources and resource combinations to specific product requirements. The aim was to support both greenfield and brownfield system design processes and their associated design systems. In our previous works, we have presented the underlying concepts and information models, while in this article, we presented our implementation of the capability matchmaking software and its interaction with external design and planning tools through its web service interface. Furthermore, we explained the information inputs and outputs that are expected from/to the client systems utilizing the matchmaking service. We also presented matchmaking test cases which validate the intended functionality of the system.

The matchmaking system utilizes formal OWL-based information representations of both products and resources, and SPIN rules to infer new knowledge from those representations. These SPIN rules are used to calculate the combined capabilities of combined resources and to compare the requirements of the product with the capabilities of the resources in order to find matches and to save that information back to the ontology. The matchmaking service and software takes inputs from different client systems, executes the various rules needed during the matchmaking process, and delivers the results back to the clients. The implementation of the service follows the RESTful architecture. Thus, it can couple easily with external design and planning systems, and does not require any specific technical ability on the client side to process the XML or JSON messages. Use of the service does not require any major software development on the client side, which is expected to facilitate quick and easy adoption of the service.

Information models and rules can never represent the real world perfectly, but are always a simplification. Thus, a human designer should check the suggestions provided by the matchmaking system and make the final decision about the resource selection. However, we believe that the approach developed here can act as a valuable aid for the system designer and reconfiguration planner. It creates the possibility of exploring large resource catalogues automatically and rapidly filtering out the unsuitable resources, leaving only the possible resources and resource combinations for the given requirement. Consequently, less manual time-consuming search and filtering effort is needed to find and evaluate different alternative solutions, and more alternative configurations can be considered, compared to a traditional design approach. It can also be used to check if the existing system already fulfils the new product requirements. Thus, we expect the capability matchmaking approach and software presented here to make the system design and reconfiguration planning process easier and speed it up. The matchmaking can also find surprising configuration solutions which may have been overlooked by a manual design approach, leading to potentially innovative system solutions. Reaching the envisioned impact requires there to be large catalogues of resources described with the Resource Description format [27] and those catalogues to be accessible to the matchmaking system. This means that the resource providers should provide the descriptions of their resources in that format and publish these through the resource catalogues.

In the future, new industrial projects should be established to test the matchmaking approach and software in wider industrial settings covering larger numbers of different process capabilities. Consequently, new capability classes and their associated properties should be implemented into the Capability Model to increase the capability catalogue when needed. Also, the rule base needs to be extended to cover the new capabilities. Furthermore, the reliability and information content of the matchmaking result could be increased by extending the current approach with automatic layout generation for feasibility checks and processing time estimations.

Acknowledgements

The authors would like to thank the team of Prof. Marcello Colledani from Politecnico di Milano for providing screenshots of the Integrated Production and Reconfiguration Planning Tool for Figure 5.

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 680759 and project title ReCaM (Rapid Reconfiguration of Flexible Production Systems through Capability-based Adaptation, Autoconfiguration and Integrated Tools for Production Planning). (www.recam-project.eu).

References

1. M. Bortolini, F.G. Galizia, C. Mora, Reconfigurable manufacturing systems: Literature review and research trend, J. Manuf. Syst. 49 (2018) 93–106. doi:10.1016/j.jmsy.2018.09.005.
2. Y. Lu, X. Xu, Resource virtualization: A core technology for developing cyber-physical production systems, J. Manuf. Syst. 47 (2018) 128–140. doi:10.1016/j.jmsy.2018.05.003.
3. F. Tao, Q. Qi, A. Liu, A. Kusiak, Data-driven smart manufacturing, J. Manuf. Syst. 48 (2018) 157–169. doi:https://doi.org/10.1016/j.jmsy.2018.01.006.
4. K.-D. Thoben, S. Wiesner, T. Wuest, "Industrie 4.0" and Smart Manufacturing – A Review of Research Issues and Application Examples, Int. J. Autom. Technol. 11 (2017) 4–16. doi:10.20965/ijat.2017.p0004.
5. O. Fisher, N. Watson, L. Porcu, D. Bacon, M. Rigley, R.L. Gomes, Cloud manufacturing as a sustainable process manufacturing route, J. Manuf. Syst. 47 (2018) 53–68. doi:10.1016/j.jmsy.2018.03.005.
6. Y. Lu, X. Xu, A semantic web-based framework for service composition in a cloud manufacturing environment, J. Manuf. Syst. 42 (2017) 69–81. doi:10.1016/j.jmsy.2016.11.004.
7. F. Tao, L. Zhang, Y. Liu, Y. Cheng, L. Wang, X. Xu, Manufacturing Service Management in Cloud Manufacturing: Overview and Future Research Directions, J. Manuf. Sci. Eng. 137 (2015) 040912. doi:10.1115/1.4030510.
8. F. Tao, J. Cheng, Y. Cheng, S. Gu, T. Zheng, H. Yang, SDMSim: A manufacturing service supply-demand matching simulator under cloud environment. Robot Comput Integr Manuf 2017;45:34–46. doi:10.1016/j.rcim.2016.07.001.
9. Y. Cheng, F. Tao, D. Zhao, L. Zhang, Modeling of manufacturing service supply-demand matching hypernetwork in service-oriented manufacturing systems. Robot Comput Integr Manuf 2017;45:59–72. doi:10.1016/j.rcim.2016.05.007.
10. F. Ameri, C. Urbanovsky, C. McArthur, A Systematic Approach to Developing Ontologies for Manufacturing Service Modeling, Proc. Work. Ontol. Semant. Web Manuf. (2012) 1–14.
11. S. Borgo, P. Leitão, Foundations for a Core Ontology of Manufacturing, in: R. Sharman, R. Kishore, R. Ramesh (Eds.), Ontologies, Springer US, 2007: pp. 751–775. doi:10.1007/978-0-387-37022-4_27.
12. R. Jardim-Goncalves, A. Grilo, K. Popplewell, Novel strategies for global manufacturing systems interoperability, J. Intell. Manuf. 27 (2016) 1–9. doi:10.1007/s10845-014-0948-x.
13. S. Strzelczak, Towards Ontology-Aided Manufacturing and Supply Chain Management – A Literature Review, in: Adv. Prod. Manag. Syst. Innov. Prod. Manag. Towar. Sustain. Growth, Springer, 2015: pp. 467–475. doi:10.1007/978-3-319-22759-7.

14. P. Leitão, A.W. Colombo, S. Karnouskos, Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges, Comput. Ind. 81 (2016) 11–25. doi:10.1016/j.compind.2015.08.004.
15. Y. Luo, L. Zhang, F. Tao, L. Ren, Y. Liu, Z. Zhang, A modeling and description method of multidimensional information for manufacturing capability in cloud manufacturing system, Int. J. Adv. Manuf. Technol. 69 (2013) 961–975. doi:10.1007/s00170-013-5076-9.
16. M. Yuan, K. Deng, W.A. Chaovalitwongse, Manufacturing Resource Modeling for Cloud Manufacturing, Int. J. Intell. Syst. 32 (2017) 414–436. doi:10.1002/int.21867.
17. Y. Lu, H. Wang, X. Xu, ManuService ontology: a product data model for service-oriented business interactions in a cloud manufacturing environment, J. Intell. Manuf. (2016) 1–18. doi:10.1007/s10845-016-1250-x.
18. F. Ameri, L. Patil, Digital manufacturing market: a semantic web-based framework for agile supply chain deployment, J. Intell. Manuf. 23 (2012) 1817–1832. doi:10.1007/s10845-010-0495-z.
19. F. Ameri, C. McArthur, Semantic rule modelling for intelligent supplier discovery, Int. J. Comput. Integr. Manuf. 27 (2014) 570–590. doi:10.1080/0951192x.2013.834467.
20. U. Rauschecker, M. Stohr, Using manufacturing service descriptions for flexible integration of production facilities to manufacturing clouds, 2012 18th Int. ICE Conf. Eng. Technol. Innov. (2012) 1–10. doi:10.1109/ICE.2012.6297693.
21. ReCaM consortium, ReCaM project web page, http://www.recam-project.eu. [Accessed 20.2.2019].
22. E. Järvenpää, N. Siltala, O. Hylli, M. Lanz, The development of an ontology for describing the capabilities of manufacturing resources, J. Intell. Manuf. (2018) 1–20. doi:10.1007/s10845-018-1427-6.
23. E. Järvenpää, N. Siltala, O. Hylli, M. Lanz, Product Model ontology and its use in capability-based matchmaking, in: Procedia CIRP, 2018. doi:10.1016/j.procir.2018.03.211.
24. N. Siltala, E. Järvenpää, M. Lanz, Creating Resource Combinations Based on Formally Described Hardware Interfaces, in: S. Ratchev (ed.) Precision Assembly in the Digital Age. IPAS 2018. IFIP Advances in Information and Communication Technology, vol. 530, 29–39, Springer, Cham. doi:10.1007/978-3-030-05931-6_3.
25. E. Järvenpää, N. Siltala, O. Hylli, M. Lanz, Capability Matchmaking Procedure to Support Rapid Configuration and Re-configuration of Production Systems, Procedia Manuf. 11 (2017) 1053–1060. doi:10.1016/j.promfg.2017.07.216.
26. E. Järvenpää, O. Hylli, N. Siltala, M. Lanz, Utilizing SPIN Rules to Infer the Parameters for Combined Capabilities of Aggregated Manufacturing Resources, IFAC-PapersOnLine. 51 (2018) 84–89. doi:10.1016/j.ifacol.2018.08.239.
27. N. Siltala, E. Järvenpää, M. Lanz, Value Proposition of a Resource Description Concept in a Production Automation Domain, in: Procedia CIRP, 2018. doi:10.1016/j.procir.2018.03.154.
28. Y. Sure, S. Staab, R. Studer, Ontology Engineering Methodology. In: S. Staab, R. Studer (Eds.), Handbook on Ontologies, 2nd edition. (2009), 135–152.
29. E. Järvenpää, N. Siltala, O. Hylli, Product, Manufacturing Resource and Capability Ontologies. (2019) Available at: http://urn.fi/urn:nbn:fi:csc-kata20190225154330611362
30. J. Pakkanen, T. Juuti, T. Lehtonen, Brownfield Process: A method for modular product family development aiming for product configuration, DESIGN STUDIES, 2016, vol 45B, pp. 210–241. DOI: 10.1016/j.destud.2016.04.004.
31. SPIN working group, SPIN – SPARQL Inferencing Notation. (2017). Available at: http://spinrdf.org/. [Accessed 15.10.2017].
32. H. Knublauch, The TopBraid SPIN API. (2016). Available at: http://topbraid.org/spin/api/ [Accessed 1.4.2017].

33. Apache Tomcat. Available at: http://tomcat.apache.org/. [Accessed 26.2.2019]
34. Java EE 6 Tutorial. Available at: https://docs.oracle.com/javaee/6/tutorial/doc/ (2013). [Accessed 26.2.2019]
35. Jersey – reference implementation of JAX-RS. Available at: https://jersey.github.io/. [Accessed 26.2.2019].
36. Apache Software Foundation, Apache Jena – A free and open source Java frawework for building Semantic Web and Linked Data applications. (2017) Available at: https://jena.apache.org/ [Accessed 10.8.2017].
37. Openllet API. Available at: https://github.com/Galigator/openllet. [Accessed 26.2.2019]
38. M. Colledani, A. Yemane, G. Lugaresi, G. Borzi, D. Callegaro, A software platform for supporting the design and reconfiguration of versatile assembly systems, in: Procedia CIRP. 72 (2018) 808–813. doi:10.1016/j.procir.2018.03.082.
39. A. Angius, A. Yemane, M. Colledani, F. Micchetti, G. Borzi, D4.4: Reconfiguration Management Platform: implementation and testing, ReCaM project deliverable (2018)